Wasserstein Weight Estimation for Stochastic Petri Nets

Tobias Brockhoff*, Merih Seran Uysal† and Wil M.P. van der Aalst‡
Process and Data Science Group (PADS), RWTH Aachen University, Aachen, Germany
Email: *brockhoff@pads.rwth-aachen.de, †uysal@pads.rwth-aachen.de, ‡wvdaalst@pads.rwth-aachen.de

***Abstract*—Traditional process models like Petri nets effectively describe the control flow of processes but fail to capture stochastic information such as choice likelihoods. To address this, Stochastic Labeled Petri Nets (SPNs) have recently gained attention, extending Petri nets with transition weights that allow to associate executions with probabilities. The language of an SPN thereby becomes a probability distribution over traces (i.e., sequences of activities). To assess an SPN's quality, Earth Mover's Stochastic Conformance (EMSC) emerged as a natural metric that measures the similarity of the SPN's trace distribution to the observed real-world distribution. In this paper, we propose a locally optimal approach for fine-tuning (or finding) transitions weights to maximize an SPN's EMSC. Leveraging the relationship between EMSC and the Wasserstein distance, which recently gained attention as a loss function in machine learning, we compute subgradients for EMSC to optimize transition weights via subgradient descent. Besides, we propose a straightforward solution to handle models that allow for infinitely many traces. Our optimization approach is broadly applicable for EMSC—that is, for EMSC using arbitrary trace-to-trace distances—unlike existing works that either to not explicitly consider EMSC or only special variants. We demonstrate the applicability of our approach on several real-life event logs and discovery algorithms, comparing it to state-of-the-art stochastic process discovery methods and a recent full automated simulation approach.**

***Index Terms*—Stochastic Process Mining, Stochastic Petri Nets, Earth Mover's Distance, Optimal Transport, Wasserstein Loss**

I. Introduction

Processes implemented in companies usually exhibit significant variability where process executions can differ in choices made or in the order of activities executed. Therefore, when modeling processes for analysis or simulation purposes, probabilities must be considered.

To assess how well a process model describes a real-life process, one typically compares its language—that is, the traces it can generate—to real-life process executions. Such real-life process executions are usually extracted from a company's information system and collected in so-called event logs. If the model includes stochastic information, not only the traces it can generate determine its language but also their respective likelihoods. Thereby, the language naturally becomes a probability distribution over (possibly infinitely many) traces where each trace is again associated with possibly infinitely many model executions. Consequently, a *good* stochastic process model should induce a trace distribution similar to the observed one. The Earth Mover's Stochastic Conformance (EMSC) metric [1] measures this similarity by assessing the effort required to transform one trace distribution into the other or, equivalently, to transport probability mass from one distribution to the other. A ground distance measures the costs of transforming individual traces. This concept is rooted in Optimum Transport (OT) [2], a subfield of probability theory that not only considers the cost but also the shape of such transportation plans. Conceptually, OT is also the foundation for the Wasserstein distance in statistics where the ground distance must be metric. Due to its intuitive notion of dissimilarity [3], the Wasserstein distance has recently gained attention as a loss function in machine learning [4], [3].

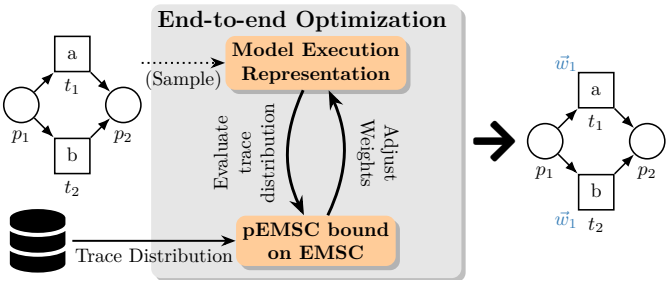


Fig. 1: Wasserstein Weight Estimation maximizing EMSC. Based on a path-based representation of the Stochastic Labeled Petri Net (SPN), we iteratively adjust the weights maximizing the similarity between the SPN's and event log's language.

In this paper, we explore the use of OT-based losses for stochastic process discovery. Specifically, we focus on *weight estimation* [5]—the discovery of Stochastic Labeled Petri Nets (SPNs), where each transition is annotated with a weight [6], from a Petri net. While we consider SPNs due to their tractable semantics and for comparability with existing works [5], [7], [1], [6], the idea might be more broadly applicable. Employing OT-based losses for stochastic generative process models only requires a tractable relation between the model's parameters and the desired output distribution.

Figure 1 gives an overview of the proposed approach. As input, we consider an event log and a Petri net (or SPN). Next, we represent the model by a finite set of model

We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy — EXC-2023 Internet of Production — 390621612

executions and the event log by a finite trace distribution. Using *TensorFlow* [8], we build a computational graph that links model weights via the model executions to activity traces and their probabilities. For a given weight assignment, we then approximate (due to the sampling) the model's trace distribution by a forward pass through this graph. To evaluate the model's quality, we use an OT-based loss function derived from a modified EMSC metric. This loss function allows us to compute *subgradients*—a generalization of gradients for non-differentiable functions—with respect to the model's trace probabilities. The subgradients are then backpropagated the graph to adjust the weights. Ultimately, we return locally optimal weights that maximize our EMSC proxy metric.

We sample model executions to address two challenges: first, infinitely many executions can correspond to the same trace of visible activities; second, there can be infinitely many visible traces. While the authors in [9] propose a method to compute the probability of any trace in an SPN solving the first challenge, we prefer sampling for its simplicity and computational tractability. Sampling avoids building a large portion of the Petri net's state space; yet both methods can be combined. In contrast, the second challenge remains unsolved for general EMSC (i.e., for general trace-level distances such as edit distance). While Leemans' et al. recently proposed an approach that also directly optimizes SPN weights with respect to EMSC [10], they only consider a special case where unequal traces have unit distance. This reduces EMSC to the total variation distance of the trace distributions, allowing them to focus on the finitely many traces in the event log. However, this sacrifices one of EMSC's main advantages—namely, the ability to account for similarity between unequal traces.

Our contributions are thus as follows:

(i) We propose an—aside from sampling—end-to-end method for optimizing SPNs with transition weights with respect to general EMSC.
(ii) We relate the EMSC metric to an OT formulation that is better suited for optimization.
(iii) We compare the proposed approach to a recent simulation approach [11], stochastic discovery approaches [7], [12], and various weight estimation methods [5] on different discovery algorithms and real-life event logs.

The remainder of this paper is structured as follows: we discuss related work and introduce EMSC in Sections II and III. In Section IV, we present our OT-based weight estimation method for SPNs. We evaluate the approach in Section V and conclude our paper in Section VI.

II. Related Work

In machine learning, OT-based/Wasserstein loss functions were investigated for training Generative Adversarial Networks [4] and in the general realm of learning (e.g. classification) [3]. Rather than solving the linear program that is commonly encountered in discrete OT, they consider a regularized, more scalable solvable form. However, for this work, it is still feasible to compute an exact solution.

In the process mining domain, our work is related to *stochastic process model discovery* and *evaluation* as well as to *process simulation*. Rogge-Solti et al. [12] proposed an early approach with an available implementation for discovering SPNs. The approach discovers a Petri net using Inductive Miner Infrequent [13], potentially repairs it, aligns the net with the event log, and solves an optimization problem to determine the weights. Unlike or end-to-end EMSC optimization approach, they to find weights such that in each state of the net (i.e., in each marking), each enabled transition fires with the probability observed in the aligned data Besides, Burke et al. propose a broader framework and various techniques for locally estimating weights for a given Petri net [5]. They also introduce the Toothpaste Miner which is an algorithm that directly discovers SPNs by reducing an initially created trace net [7]. Finally, the automated simulation approach in [11] combines hyperparameter tuning during a process model discovery phase with a weight estimation method.

To measure how well an SPN models observed event data, Leemans et al. proposed EMSC [1]. Moreover, in [9], [14], a method was introduced to compute the probability of an arbitrary trace in an SPN, even if induced by infinitely many model executions. Yet, computing EMSC remains challenging for models with loops involving visible activities, leading to infinitely many visible activity traces. When calculating distances from event log traces to not yet unfolded model traces is complex (e.g., edit distance), current EMSC implementations still rely on sampling. Merely for uEMSC, where unequal traces have unit distance, this can be avoided as it suffices to consider traces contained in both the event log *and* the model. To handle infinitely many model traces, stochastic quality measures based on entropy were proposed [15], [6]. However, these measures are less intuitive than EMSC.

Recently, two approaches were proposed which also directly optimize an SPN's weights with respect to EMSC. The approach in [10] avoids sampling but is restricted to uEMSC. In [16], the authors use a general, gradient-free function minimization method. Unlike this method, we explicitly utilize information from EMSC's solution by computing subgradients. Additionally, their method only considers model traces present in the event log, whereas we also account for potentially deviating behavior. Finally, they normalize the model sample, while we propose a modification of EMSC to handle residual probability mass. Preliminary results in Section V indicate that normalization combined with our method performs worse.

III. Preliminaries

This section introduces SPNs and EMSC [1].

We denote sets by capital letters. Let S be a set. A multiset over S is a function $M\colon S \to \mathbb{N}$ that assigns a quantity to each element. The number of elements in M is denoted as $|M|$, and the support of M, $\mathrm{supp}(M) = \{s \in S \mid M(s) > 0\}$, is the set of elements that occur. The union $(+)$, difference $(-)$, and comparison $(\leq, \geq)$ of M and another (multi)set are defined element-wise. In doing so, we assume that sets are implicitly lifted to multisets and that negative differences do

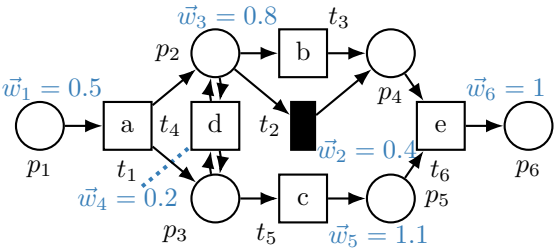


Fig. 2: Example SPN N^{ex} using a vector notation to denote transitions weights (i.e., weight $\vec{w}_i$ refers to transition t_i).

not occur. The set of all multisets over S is denoted as $\mathcal{B}(S)$, and S^* is the set of all finite sequences of elements of S. The concatenation of two sequences $\sigma, \nu \in S^*$ is denoted as $\sigma \cdot \nu$.

We write $\mathbf{f} \in (\mathbb{R}^{\geq 0})^{m \times n}$ to denote a non-negative flow between m sources and n targets and $\vec{v} \in \mathbb{R}^n$ to denote an n-dimensional vector. The flow between the ith source and jth target is $\mathbf{f}_{ij}$, and $\vec{v}_i$ is the ith entry of $\vec{v}$.

In this paper, we focus on the control flow of processes. Therefore, we model process executions as finite sequences of activities from the universe of activities $\mathcal{A}$. An *event log* collects multiple processes executions.

Definition 1 (Event Log)**.** Given a finite alphabet $\Sigma \subseteq \mathcal{A}$, an *event log* $L \in \mathcal{B}(\Sigma^*)$ is a finite multiset of traces over Σ.

Stochastic languages assign probabilities to process executions, while truncation reduces computation times later.

Definition 2 (Stochastic Language)**.** A (truncated) stochastic language over a finite alphabet $\Sigma \subseteq \mathcal{A}$ is a function $h \colon \Sigma^* \to [0, 1]$ such that $\sum_{\sigma \in \Sigma^*} h(\sigma) \overset{(\leq)}{=} 1$.

Under a frequentist interpretation of probability, each event log $L \in \mathcal{B}(\Sigma^*)$ can be translated into a stochastic language $h \colon \Sigma^* \to [0, 1]$ with $h(\sigma) = \frac{L(\sigma)}{|L|}$.

Stochastic Labeled Petri Nets (SPNs) explicitly model the stochasticity of processes with respect to the control flow. They extend Petri nets by transition weights which determine the likelihood of a transition to *fire* in a given state of the net.

Definition 3 (Stochastic Labeled Petri Net)**.** Let $\Sigma \subseteq \mathcal{A}$ be a finite alphabet, and let $\tau \notin \mathcal{A}$ denote the silent label. A Stochastic Labeled Petri Net (SPN) is a quintuple (P, T, A, l, w) of disjoint sets of places P and transitions T, an edge relation $A \subseteq (P \times T) \cup (T \times P)$, a labeling function $l \colon T \to \Sigma \cup \{\tau\}$, and a weight function $w \colon T \to \mathbb{R}^{\geq 0}$.

Figure 2 shows an SPN with five visible and one silent transition (t_2). Assuming that transitions are implicitly ordered, we represent the weight function by means of a weight vector.

Let $N = (P, T, A, l, w)$ be an SPN. For an element $x \in P \cup T$, we denote the set of elements connected via an ingoing (outgoing) edge as ${}^\bullet x$ ($x^\bullet$). A *marking* $m \in \mathcal{B}(P)$ is a multiset of places. The marking m *enables* a transition $t \in T$ if it marks all places that have an incoming edge to t—that is, ${}^\bullet t \leq m$ (e.g., in Figure 2, t_1 is enabled in $[p_1]$). Firing the enabled transition t, denoted as $m[t\rangle m'$, consumes a token from each input place and produces a token in each output place (i.e., $m' = m - {}^\bullet t + t^\bullet$). For example, in Figure 2, firing t_1 in $[p_1]$ results in the marking $[p_2, p_3]$. Given a marking $m \in \mathcal{B}(P)$, we denote the *set of enabled transitions* as $\text{enabled}_N(m)$—for instance, $\text{enabled}_N([p_2, p_3]) = \{t_2, t_3, t_4, t_5\}$. Firing an enabled transition t in a marking $m \in \mathcal{B}(P)$ of N is associated with the probability $\frac{w(t)}{\sum_{t' \in \text{enabled}_N(m)} w(t')}$. For example, the probability of firing t_3 in $[p_2, p_3]$ is $\frac{0.8}{0.2+0.8+0.4+1.1}$.

Let $m_I \in \mathcal{B}(P)$ be an initial marking of N. A *path* (c.f. [9]) through N is a sequence $\lambda = \langle t_1, \ldots, t_n \rangle \in T^*$ of transitions associated with a sequence of intermediate markings $\langle m_0, \ldots, m_n \rangle$ such that (i) λ starts in the initial marking (i.e., $m_0 = m_I$), (ii) each step is a valid transition firing (i.e., $m_{i-1}[t_i\rangle m_i$, $i = 1, \ldots, n$), and (iii) in the final marking m_n, no transition is enabled (i.e., $\text{enabled}_N(m_n) = \emptyset$). For example, the path $\lambda_1 = \langle t_1, t_4, t_4, t_5, t_2, t_6 \rangle$ starts in the initial marking $[p_1]$ and ends in the marking $[p_6]$, which is a deadlock. By multiplying the individual transition firing probabilities along the path, we obtain a path's probability (e.g., the path λ_1 has probability $\frac{0.5}{0.5} \cdot \frac{0.2}{2.5} \cdot \frac{0.2}{2.5} \cdot \frac{1.1}{2.5} \cdot \frac{0.4}{1.2} \cdot \frac{1}{1} \approx 0.005$). Finally, stochastic path languages describe the behavior of SPNs.

Definition 4 ((Truncated) Stochastic Path Language)**.** Let $N = (P, T, A, l, w)$ be an SPN with initial marking $m_I \in \mathcal{B}(P)$. Let $\Lambda \subseteq T^*$ be a set of *paths* for N and m_I. The stochastic path language of Λ given N and m_I is given as $g^p_{N,m_I,\Lambda} \colon T^* \to [0, 1]$ with (i) $g^p_{N,m_I,\Lambda}(\lambda) = 0$ if $\lambda \notin \Lambda$ and (ii) $g^p_{N,m_I,\Lambda}(\lambda) = \Pi_{i=1}^n \frac{w(t_i)}{\sum_{t' \in \text{enabled}_N(m_{i-1})} w(t')}$ if $\lambda = \langle t_1, \ldots, t_n \rangle \in \Lambda$ with markings $\langle m_0, \ldots, m_n \rangle$ along the path. We call $g^p_{N,m_I,\Lambda}$ truncated if Λ does not contain all possible paths of N and, therefore, $\sum_{\lambda \in \Lambda} g^p_{N,m_I,\Lambda}(\lambda) < 1$.

Earth Mover's Stochastic Conformance [1] is a similarity measure for stochastic languages and (truncated) stochastic path languages. It is based on the minimal effort required to transform one language into the other. The cost of transforming a trace into a path is measured by a *trace-path distance*.

Definition 5 (Earth Mover's Stochastic Conformance [1])**.** Let $h \colon \Sigma^* \to [0, 1]$ be a finite stochastic language over a finite set of activities $\Sigma \subseteq \mathcal{A}$ with $\text{supp}(h) = \{\sigma_1, \ldots, \sigma_m\}$. Let $N = (P, T, A, l, w)$ be an SPN with initial marking $m_I \in \mathcal{B}(P)$. Let $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ be a finite set of paths for N and m_I and $\delta^{tp} \colon \Sigma^* \times T^* \to [0, 1]$ be a trace-path distance. Earth Mover's Stochastic Conformance (EMSC) between h and $g_{\lambda,N}$ is defined by the following optimization problem:

$$\text{EMSC}_{\delta^{tp}}(h, g^p_{N,m_I,\Lambda}) = 1 - \min_{\mathbf{f} \in (\mathbb{R}^{\geq 0})^{m \times n}} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{f}_{ij} \delta^{tp}(\sigma_i, \lambda_j) \tag{1a}$$

$$\text{s.t.} \sum_{j=1}^{n} \mathbf{f}_{ij} = h(\sigma_i) \qquad i = 1, \ldots, m \tag{1b}$$

$$\sum_{i=1}^{m} \mathbf{f}_{ij} \geq g^p_{N,m_I,\lambda}(\lambda_j) \quad j = 1, \ldots, n \tag{1c}$$

EMSC can be computed using linear programming, frequently utilizing post-normalized edit distance as a trace-path distance based on the visible activities along a path (i.e., the trace). This approach allows for aggregating paths sharing the same trace prior to solving EMSC. Using techniques from [9], one can therefore still compute EMSC exactly if the number of paths is infinite but the number of traces is finite. However, for infinitely many traces, the size of EMSC's optimization problem becomes infinite making naive solving using linear programming infeasible. The special case of uEMSC [1], [10] is an exception as it allows for a finite formulation.

IV. Wasserstein Weight Estimation

We propose a *weight estimation* method for SPNs that uses subgradient ascent to optimize the weights with respect to EMSC. For Petri nets, we randomly initialize the weights.

Figure 3 gives an overview of our method. Initially, we need a finite representation of the SPN to later compute EMSC for any ground distance using linear programming techniques (cf. Section III). While one could sample a *finite* number of *traces* and express their probability based on the transition weights using techniques from [9], we opt to sample a *finite* set of *paths* and leave a combination with [9] for future work. Sampling paths is simpler, yields good results, and requires fewer computations than the method in [9].

In Figure 3, we show a path sample containing four paths from N^{ex} (Figure 2). The main approach consists of three steps each in the forward and backward passes. In the forward pass, we first compute a path's probability based on the SPN's weights. Second, we aggregate paths to obtain trace probabilities. Third, we compare the stochastic language of the path sample with that of the event log using a modified version of EMSC. In the backward pass, we first compute a subgradient of the loss function with respect to the model's stochastic language. Using automatic differentiation, this subgradient is backpropagated to adjust the SPN's weights.

While Definition 4 covers the first, differentiable step of the forward pass, this section discusses the remaining steps. In the following, let $N = (P, T, A, l, w)$ be an SPN with initial marking $m_I \in \mathcal{B}(P)$ and Λ be a set of paths for N and m_I.

a) SPN Trace Probabilities: As discussed in [1], the relation between *paths* of N and *traces* is straightforward. First, we project a path on the sequence of visible labels:

$$\pi_N^{vis}(\langle\rangle) = \langle\rangle \tag{2a}$$

$$\pi_N^{vis}(\lambda \cdot \langle t \rangle) = \begin{cases} \pi_N^{vis}(\lambda) & \text{if } l(t) = \tau \\ \pi_N^{vis}(\lambda) \cdot \langle l(t) \rangle & \text{else} \end{cases} \tag{2b}$$

Second, a trace's probability is the sum of those paths' probabilities whose sequence of activities is equal to the trace:

$$g^{tr}_{N,m_I,\Lambda}(\nu) = \sum_{\substack{\lambda \in \Lambda \\ \pi_N^{vis}(\lambda) = \nu}} g^{p}_{N,m_I,\Lambda}(\lambda), \ \nu \in \Sigma^* \tag{3}$$

This aggregation step is trivially differentiable. In Figure 3, the paths λ_1 and λ_2 both correspond to the trace $\langle a, c, e \rangle$.

b) Loss Function: Maximizing N's EMSC score, Equation (1c) can become problematic. Assume that N allows for infinitely many paths, and let $\Lambda' \subseteq T^*, \Lambda' \cap \Lambda = \emptyset$ denote the non-empty set of remaining paths. If we adjust the weights such that the probability of Λ' increases, the probability of Λ decreases. Consequently, Equation (1c) allows that the probability of each trace in the event log can be reallocated to its *nearest* trace in the model. In fact, for a fixed set of model traces, the reallocation in which each event log trace reallocates its full probability to its *nearest* model trace would achieve the best EMSC score. Thus, there are two ways for an optimization algorithm to improve EMSC: improve the similarity of the stochastic languages *or* reduce the total probability of Λ. This can also happen in case of infinitely many *traces* (rather than paths).

This can either be alleviated by normalizing probabilities or by a modified formulation that punishes residual probability. Conceptually, we capture residual probability by introducing an auxiliary model trace. In a slight abuse of notation, we denote this trace with the label $\perp$ in Figure 3. From this trace, probability can be reallocated to log traces at configurable costs. Treating sampling on the log side symmetrically, we obtain the following variant of EMSC.

Definition 6 (Penalized Earth Mover's Stochastic Conformance (pEMSC))**.** Let $\Sigma \subseteq \mathcal{A}$ be a finite set of activities, $h, g \colon \Sigma^* \to [0, 1]$ denote (truncated) finite stochastic languages with $\text{supp}(h) = \{\sigma_1, \ldots, \sigma_m\}$ and $\text{supp}(g) = \{\nu_1, \ldots, \nu_n\}$, and $\delta^{tr} \colon \Sigma^* \times \Sigma^* \to [0, 1]$ be a trace distance. Penalized Earth Mover's Stochastic Conformance (pEMSC) between h and g is defined by the optimization problem:

$$\text{pEMSC}_{\delta^{tr},\vec{\xi}^{\text{m}},\vec{\xi}^{\text{l}}}(h, g) = 1 - \min_{\mathbf{f} \in (\mathbb{R}^{\geq 0})^{(m+1)\times(n+1)}}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{f}_{ij}\delta^{tr}(\sigma_i, \nu_j) + \sum_{i=1}^{m} \vec{\xi}^{\text{m}}_i \mathbf{f}_{i(n+1)} + \sum_{j=1}^{n} \vec{\xi}^{\text{l}}_j \mathbf{f}_{(m+1)j} \tag{4a}$$

$$\text{s.t. } \mathbf{f}_{i(n+1)} + \sum_{j=1}^{n} \mathbf{f}_{ij} = h(\sigma_i) \qquad i = 1, \ldots, m \tag{4b}$$

$$\mathbf{f}_{(m+1)j} + \sum_{i=1}^{m} \mathbf{f}_{ij} = g(\nu_j) \qquad j = 1, \ldots, n \tag{4c}$$

$$\sum_{j=1}^{n+1} \mathbf{f}_{(m+1)j} = 1 - \sum_{i=1}^{m} h(\sigma_i) \tag{4d}$$

$$\sum_{i=1}^{m+1} \mathbf{f}_{i(n+1)} = 1 - \sum_{j=1}^{n} g(\nu_j) \tag{4e}$$

where $\vec{\xi}^{\text{m}}, \vec{\xi}^{\text{l}} \geq 0$ penalize residual probability mass.

Equations (4b) and (4c) extend Equations (1b) and (1c) for EMSC by the possibility to reallocate probability mass to the auxiliary model and log traces. Moreover, Equations (4d) and (4e) complement Equations (4b) and (4c) limiting the flows to the auxiliary traces. Finally, we extend the objective function by costs for flows adjacent to the auxiliary traces. For example, the penalty vector $\vec{\xi}^{\text{m}}$ penalizes flows between

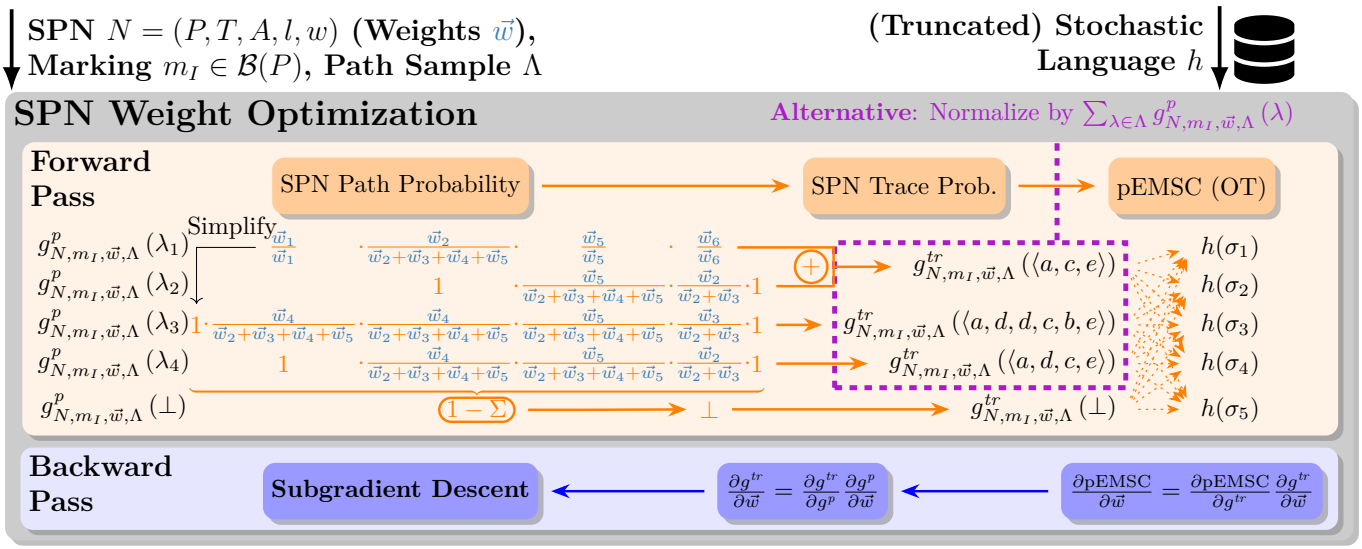


Fig. 3: Overview of the computation showing the forward and backward pass of our optimization method.

the auxiliary model trace and the event log. Note that any flow between the auxiliary traces is free. Using the analogy of the auxiliary model and log trace, one can also merge Equations (4b) and (4d) and Equations (4c) and (4e). This leads to the typical linear programming formulation of OT.

Assume that the complete log is used. We show that EMSC is recovered from pEMSC using a best-case auxiliary trace distance. Moreover, unit distance gives a lower bound.

Theorem 1 (pEMSC bounds on EMSC)**.** *Let $\Sigma \subseteq \mathcal{A}$ be a finite set of activities and $h\colon \Sigma^* \to [0,1]$ be a finite stochastic language with $\mathrm{supp}(h) = \{\sigma_1, \dots, \sigma_m\}$. Let $N = (P, T, A, l, w)$ be an SPN with initial marking $m_I \in \mathcal{B}(P)$ and Λ be a finite set of paths for N and m_I with $\mathrm{supp}(g^{tr}_{N,m_I,\Lambda}) = \{\nu_1, \dots \nu_n\}$ being the set of visible traces. Let $\delta^{tr}\colon \Sigma^* \times \Sigma^* \to [0,1]$ be a trace distance. For the trace-path distance $\delta^{tp}\colon \Sigma^* \times T^* \to [0,1]$, with $\delta^{tp}(\sigma, \lambda) = \delta^{tr}(\sigma, \pi^{vis}_N(\lambda))$ for all $\sigma \in \mathrm{supp}(h)$ and $\lambda \in \Lambda$, penalty vectors $\vec{\xi}^{\mathrm{m-min}}, \vec{\xi}^{\mathrm{m-min}} \in \mathbb{R}^m$ with*

$$\vec{\xi}^{\mathrm{m-min}}_i = \min_{j=1,\dots,n} \delta^{tr}(\sigma_i, \nu_j),\ \vec{\xi}^{\mathrm{m-max}}_i = 1,\ i = 1, \dots, m \tag{5}$$

and arbitrary log penalty vectors $\vec{\xi}^{\mathrm{l}} \in \mathbb{R}^n$, it holds

$$\begin{aligned} &\mathrm{pEMSC}_{\delta^{tr},\vec{\xi}^{\mathrm{m-max}},\vec{\xi}^{\mathrm{l}}}(h, g^{tr}_{N,m_I,\Lambda}) \\ &\leq \mathrm{pEMSC}_{\delta^{tr},\vec{\xi}^{\mathrm{m-min}},\vec{\xi}^{\mathrm{l}}}(h, g^{tr}_{N,m_I,\Lambda}) \\ &= \mathrm{pEMSC}_{\delta^{tp}}(h, g^{p}_{N,m_I,\Lambda}) \end{aligned} \tag{6}$$

Proof Sketch. The penalty vectors do not occur in the constraints. Consequently, each solution for $\vec{\xi}^{\mathrm{m-max}}$ is feasible for $\vec{\xi}^{\mathrm{m-min}}$ with lower pEMSC. Thus, the inequality relation holds. In an optimal EMSC solution, any overfilled path receives flow from log traces for which it is the nearest path. More precisely, the flow from these traces must at least match the probability mass that exceeds this path's capacity. Otherwise, we could reduce costs by re-routing flow from a trace for which this path is not its nearest path to the trace's nearest path. This would not violate any constraint and thus contradicts optimality proving equality. □

PEMSC (and EMSC) are non-differentiable rendering classical gradient-based optimization infeasible. However, we can obtain a subgradient using duality theory. For the sake of readability, we add auxiliary traces σ_{m+1} and ν_{n+1} to Definition 6. Additionally, we extend h and g by the respective residual probabilities and incorporate the penalty vectors $\vec{\xi}^{\mathrm{l}}$ and $\vec{\xi}^{\mathrm{m}}$ into δ^{tr}. Then, the dual problem of the minimization problem in pEMSC's is:

$$\max_{\vec{u} \in \mathbb{R}^{m+1}, \vec{v} \in \mathbb{R}^{n+1}} \sum_{i=1}^{m+1} u_i h(\sigma_i) + \sum_{j=1}^{n+1} u_j g^p_{N,m_I,\Lambda}(\nu_j) \tag{7a}$$

$$\text{s.t. } \delta^{tr}(\sigma_i, \nu_j) \geq v_j - u_i, \quad \text{for all } i, j \tag{7b}$$

Using duality theory, for an optimal solution $(\vec{u}^*, \vec{v}^*)$, the dual variable $\vec{u}^*_i$ captures how changing the probability of σ_i affects the optimal objective function value. In fact, the vector $\vec{u}^*$ is a subgradient of pEMSC at its optimal solution $\mathbf{f}^*$ with respect to h [17, p. 250].

By decreasing the minimization term in pEMSC (using $\vec{\xi}^{\mathrm{m-max}}$), we increase pEMSC and, consequently, EMSC since pEMSC is a lower bound. As shown in Figure 3, we use subgradient descent to adjust the weights. In doing so, we exploit that, given a path sample, there is a differentiable relation between the SPN weights and the probabilities of the associated model traces. This allows us to backpropagate the subgradient through the model trace probabilities to update the weights. After adjusting the weights, we re-evaluate the model probabilities with another forward pass and repeat the process. This results in an iterative optimization method.

c) Sampling: Penalization encourages the optimization algorithm to reduce residual probability, even if this increases the likelihood of model traces that are far from all log traces. For instance, for edit distance only traces with distinct sets

of activities have a distance of $1 = \vec{\xi}^{\mathrm{m-max}}$. Therefore, it is desirable for initial model sample to include traces similar to those in the event log. While further investigation is left for future work, we simply assign unit weights to all transitions and unfold the most likely paths (as in [1]). This approach favors short paths that enable few transitions and is based on the idea that short traces are more likely to occur in reality.

d) Two-phase Approach: As shown in Section V, our approach is relatively fast for samples that contain several hundred paths and traces from the event log. To incorporate more information despite the sampling, we investigate a two-phase approach. First, we train on a fixed sample of paths and the most likely log traces. Second, we create a large sample of paths that are most likely under the current weights. From this sample, we create multiple subsamples by weighted sampling. Likewise, we perform a weighted subsampling of the event log. This approach tries to strike a balance between incorporating new information and avoiding large residuals. Finally, we continue to optimize the weights cycling through different input pairs of path and event log samples.

A. Implementation

We implemented our approach using *TensorFlow* [8], the Adam [18] optimizer, the *Python Optimal Transport* [19] package, and post-normalized edit distance for pEMSC[1]. Besides, there are three main considerations: possibly negative transition weights, the large product in the path probability computation, and the stopping criterion. In practice, transition weights can become negative during optimization for models with unlikely behavior. While we tried approaches like clipping, logarithmic barrier functions, or optimizing in the log domain, simply using absolute weights yielded good results. We allow weights to become negative but use their absolute value in computations. For a path $\lambda \in \Lambda$, we compute its probability as $\exp\left(\log(g^p_{N,m_I,\Lambda}(\lambda))\right)$. The logarithm converts the product into a summation, leading to faster computations. We stop the optimization after 5000 iterations or if the sum of pEMSC losses over the last 25 iterations does not differ significantly from that of their preceding 25 iterations.

V. Evaluation

We evaluate our method Wasserstein Weight Estimation (WaWE) using various real-life event logs and process discovery algorithms. We assess WaWE's computational feasibility and compare it to related work on SPNs. Additionally, we consider the simulation model discovery method Simod [11]. The experiments were conducted on an AMD Ryzen Threadripper 1920X 12-Core CPU with 90 GB main memory.

a) Setup: Like in [5], we consider three real-life event logs—namely, Road Traffic Fine Management (RTFM) [20], Sepsis [21], and BPI Challenge 18 Reference Log (BPIC18-ref) [22] (projected "reference" documents). For these event logs, we discover process models using (i) Inductive Miner Infrequent noise threshold 0.2 (IMf02) [13], (ii) a Heuristic Miner (HM) variant [23], (iii) Directly Follows Miner (DFM) [24], and (iv) Split Miner (SM) [25]. We only deviated from the default parameters for HM and SM. The model discovered by HM on RTFM allows for a single trace, so we reduced the thresholds. Simod uses SM and hyperparameter tuning but requires start lifecycle events and resource information. Former were created by duplicating the complete events. Moreover, we introduced artificial resources, each handling a single activity. This should not affect the stochastic process model as the discovery does not consider resource information. For SM, we re-discovered the model using the hyperparameters returned by Simod, as Simod's models have separate transitions for the start and completion of activities.

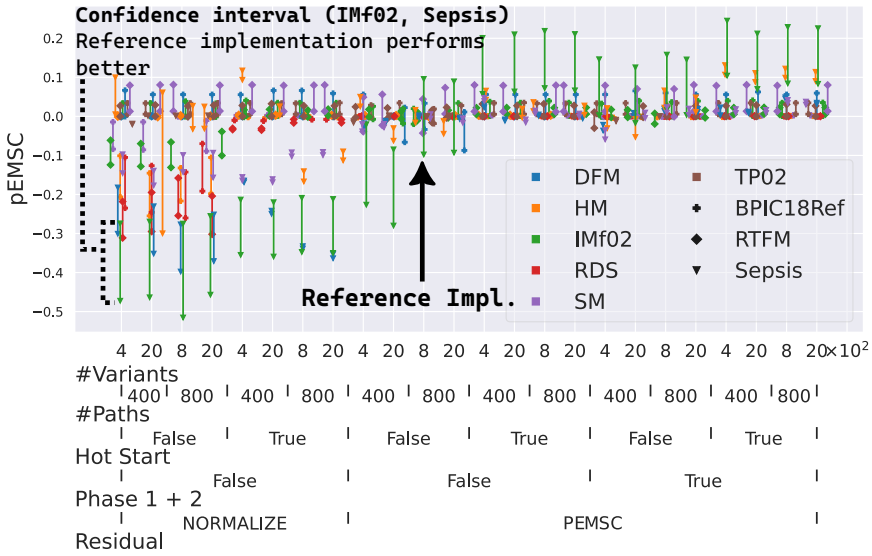
For each Petri net, we discovered three SPNs using the weight estimators MSAPE, FDE, and ABE as introduced in [5]. Thereby, we ensure that we take into account the best—or at least a well—performing estimator for each scenario considered in [5]. Moreover, we consider two stochastic process discovery approaches: the Toothpaste Miner with noise threshold 0.2 (TP02) [7] and the Rogge-Solite Stochastic Discovery (RDS) [12] algorithm.

Except for Simod, which does not return an SPN, we evaluate the SPNs with respect to EMSC [1], Entropy-based Recall (Ent. Rec.), and Entropy-based Precision (Ent. Prec.) [6]. For Simod, we generate 100k cases to approximate the stochastic language of the simulation model. Thereby, we can still compute the log-log variant of EMSC [26].

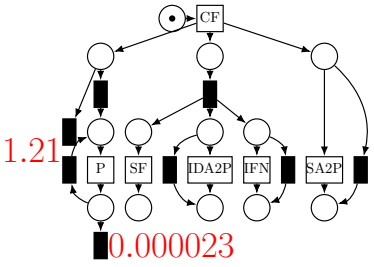
A. Parameterization WaWE

The setup of WaWE involves five primary hyperparameters: the maximum number of (i) paths and (ii) log variants, (iii) whether we use an SPN's weights for a warm start, (iv) whether we execute the second optimization phase, and (v) whether we normalize residuals or employ pEMSC using $\vec{\xi}^{\mathrm{m-max}}$. We therefore tested various parameters to evaluate WaWE's sensitivity. When considering the size of the pEMSC problem, restricting it to 400 paths and traces typically leads to fast optimization, whereas a size of 800 by 800 already notably increases the runtime. Besides, a maximum number of 2000 traces in the log's language ensures that the entire event log is considered in all test event logs. The x-axis in Figure 4a depicts the parameter combinations where we excluded residual normalization with a second optimization phase for readability reasons. We repeated the experiment 30 times. Given that raw metric values vary across event logs and process discovery algorithms, we show differences relative to a reference parameterization. As reference implementation, we use pEMSC with at most 800 paths and log variants, disable warm-starting, and disable phase II. Like the Wilcoxon signed-rank test, we examine differences in mean pEMSC (using $\vec{\xi}^{\mathrm{m-max}}$). To limit computational complexity, we sample the 15k most probable paths in the model. Besides, we consider pEMSC rather than EMSC to penalize models that cannot be easily sampled. Finally, in Figure 4a, we display bootstrapped confidence intervals on the differences in mean pEMSC for

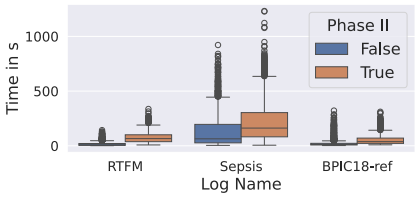
[1]https://github.com/tbr-git/wasserstein-spn-weight-estimation



(a) Bootstrapped 95% confidence intervals on differences in mean pEMSC



(b) Snippet of a model obtained by optimizing EMSC



(c) Computation times

Fig. 4: Effect of WaWE's hyperparameters on (a) the pEMSC score (with unit penalization) of the resulting SPN compared to a reference implementation, (b) the resulting model using EMSC as optimization criterion, and (c) the computation times.

various discovery algorithms (color) and event logs (shape). A negative value indicates that, on average, the reference parameterization performs better.

a) Results: For *most* log-model combinations, WaWE using pEMSC is robust. The confidence intervals are mostly narrow and centered around zero. In contrast, normalizing residual probability often performs notably worse. Therefore, one can expect good results by repeating the optimization for a few times using the reference parameterization. Yet, for some event logs and models, the reference parameterization exhibits notable variance in pEMSC. In general, for the Sepsis log, which contains many unique and long traces, and the IMf02 model, which allows for a lot of behavior, pEMSC exhibits large variance. Finally, warm-starting or adding a second optimization phase can have a small positive effect.

Figure 4c shows the times measured for *all* hyperparameter settings using pEMSC distinguishing different event logs and whether we execute phase II. Usually, WaWE terminates in less than 5 minutes, and, for simpler event logs, in less than two minutes. As expected, adding a second phase considerably increases computation times. While there are some outliers, we conclude that WaWE is computationally feasible.

Finally, we also tried to directly optimize EMSC. Figure 4b shows a snippet of an SPN where this lead to minimizing the total probability of the path sample. Repeating the *Payment* transition has a high weight (≈ 1.2) compared to exiting the loop ($\approx 10^{-5}$). Being further embedded in a concurrent part of the model, even finding the most probable path using the classical unfolding of likely states became infeasible. Thus, we could not compute EMSC for this model.

B. Comparison State-of-the-Art

Table I shows a comparison between WaWE's reference parameterization, the simulation method Simod, and other state-of-the-art stochastic process discovery approaches. Despite warm-starting the optimization can lead to better results, the reference implementation is applicable to SPNs *and* Petri nets. Like before, we consider at most 15k paths for EMSC. Therefore, since EMSC might favor path samples that have low total probability, we do not report a score if the sample covers less than 0.85 probability. We report the *median* score of the 30 repetitions for WaWE. Finally, for a simple assessment of an SPN's complexity, we also report the number graph elements (i.e., sum of nodes and edges). Detailed results for each weight estimator by Burke et al. [5] as well as preliminary results for the method by Leemans et al. [10] can be found online[2]. Interestingly, incorporating latter method does not really change the results shown in Table I. Usually, the considered weight estimator perform better.

a) Results: In all but one case, WaWE matches or improves over existing approaches in terms of the models' EMSC score. In particular for IMf02, the improvement is quite large. The process models that were most suitable for normal weight estimation were discovered using SM. For the RTFM and BPIC18-ref logs, both approaches achieve almost perfect score, while, for Sepsis, WaWE yields a slightly lower EMSC score. However, there is still some room for improvement in the latter case as the best score achieved by WaWE over all iterations is 0.647 (not shown in Table I). Besides, on the same event log but different discovery algorithms, only WaWE discovered models for which we could cover the required probability mass with the limited number of paths.

[2]https://doi.org/10.6084/m9.figshare.26819437

TABLE I: Comparing WaWE's reference implementation (right) to related work (left). For non-stochastic discovery algorithms (underlined), we report the best score achieved by the estimators by Burke et al. [5]. On the right-hand side of each cell, we report the median score achieved by WaWE. Finally, we report the size of the SPNs summing the number of transitions, places, and arcs.

Log Name	Alg.	EMSC	pEMSC	Ent. Rec.	Ent. Prec.	Size
BPIC18-ref	DFM	**0.98** — **0.98**	**0.98** — **0.98**	**0.73** — **0.73**	**1.00** — **1.00**	21
	HM	0.98 — **0.99**	**0.98** — **0.98**	**1.00** — 0.88	**1.00** — 0.99	41
	IMf02	0.96 — **0.99**	0.95 — **0.99**	**0.96** — **0.96**	**1.00** — **1.00**	32
	RDS	0.96 — **0.98**	0.95 — **0.98**	**0.93** — 0.83	0.84 — **1.00**	62
	SM	**0.99** — **0.99**	**0.98** — **0.98**	**0.85** — **0.85**	**1.00** — 0.99	51
	Simod	0.85 — **0.99**				51
	TP02	0.90 — **0.98**	0.90 — **0.98**	**0.73** — **0.73**	**1.00** — **1.00**	18
RTFM	DFM	**0.95** — **0.95**	**0.95** — **0.95**	**0.82** — **0.82**	**1.00** — **1.00**	27
	HM	0.95 — **0.97**	0.95 — **0.97**	**0.92** — **0.92**	**1.00** — **1.00**	40
	IMf02	0.76 — **0.86**	0.76 — **0.86**	**0.93** — **0.93**	0.91 — **0.99**	74
	RDS	0.58 — **0.61**	0.45 — **0.61**	0.68 — **1.00**	**0.27** — 0.23	131
	SM	**0.98** — **0.98**	**0.98** — **0.98**	**0.94** — **0.94**	**0.99** — **0.99**	60
	Simod	0.97 — **0.98**				60
	TP02	0.88 — **0.90**	0.88 — **0.90**	**0.87** — **0.87**	**1.00** — **1.00**	27
Sepsis	DFM	- — **0.60**	0.45 — **0.60**	**0.58** — **0.58**	0.25 — **0.63**	90
	HM	- — **0.47**	0.33 — **0.46**	**0.49** — **0.49**	0.08 — **0.29**	82
	IMf02	- — **0.43**	0.16 — **0.40**	**0.79** — **0.79**	**0.21** — 0.17	86
	SM	**0.62** — 0.59	**0.61** — 0.59	**0.32** — **0.32**	0.29 — **0.30**	144
	Simod	0.57 — **0.59**				144
	TP02	0.72 — **0.75**	0.72 — **0.75**			1424

While the Ent. Rec. and Ent. Prec. scores are quite similar as well, WaWE achieves higher precision for Sepsis. In contrast, on BPIC18-ref, WaWE achieves slightly lower recall in two cases. Compared to Simod, WaWE seems to be a promising approach that can lead to simulation models that better capture probabilities in the control flow.

VI. Conclusion and Future Work

In this paper, we propose an end-to-end approach for optimizing the weights of a given Stochastic Labeled Petri Net (SPN) with respect to its Earth Mover's Stochastic Conformance (EMSC) score. To this end, we introduce an Optimum Transport (or Wasserstein) formulation of EMSC which lower-bounds EMSC but is better suited for optimization. Using duality theory, we obtain a subgradient with respect to this formulation which we backpropagate to the SPN's weights. Intuitively, we thereby optimize the similarity between the distribution of traces generated by the SPN and the distribution of traces observed in the event data. We show that the proposed method performs well compared to existing stochastic process discovery and process simulation approaches.

For future work, we plan to consider more event logs and to compare our approach to the simultaneously proposed weight optimization approach by Leemans et al. [10]. Besides, as the idea to use a Wasserstein loss is more generally applicable for generative models, we want to investigate applications in process simulation. In particular, we aim to include the time dimension into an end-to-end optimization approach.

References

[1] S. J. J. Leemans, A. F. Syring, and W. M. P. van der Aalst, "Earth movers' stochastic conformance checking," in *BPM Forum*, 2019.

[2] C. Villani, *Optimal Transport*, 1st ed., 2009.

[3] C. Frogner, C. Zhang, H. Mobahi, M. Araya-Polo, and T. Poggio, "Learning with a Wasserstein loss," in *NeurIPS*, 2015, p. 2053–2061.

[4] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, vol. 70, 2017, pp. 214–223.

[5] A. Burke, S. J. J. Leemans, and M. T. Wynn, "Stochastic process discovery by weight estimation," in *ICPM Workshops*, 2021, pp. 260–272.

[6] S. J. J. Leemans and A. Polyvyanyy, "Stochastic-aware conformance checking: An entropy-based approach," in *CAiSE*, 2020, pp. 217–233.

[7] A. Burke, S. J. J. Leemans, and M. T. Wynn, "Discovering stochastic process models by reduction and abstraction," in *PETRI NETS*, 2021, pp. 312–336.

[8] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.

[9] S. J. J. Leemans, F. M. Maggi, and M. Montali, "Reasoning on labelled petri nets and their dynamics in a stochastic setting," in *BPM*, 2022, pp. 324–342.

[10] S. J. J. Leemans, T. Li, M. Montali, and A. Polyvyanyy, "Stochastic process discovery: Can it be done optimally?" in *CAiSE*, 2024, pp. 36–52.

[11] M. Camargo, M. Dumas, and O. González-Rojas, "Automated discovery of business process simulation models from event logs," *Decis. Support Syst.*, vol. 134, p. 113284, 2020.

[12] A. Rogge-Solti, W. M. P. van der Aalst, and M. Weske, "Discovering stochastic petri nets with arbitrary delay distributions from event logs," in *BPM Workshops*, 2014.

[13] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *BPM Workshops*, 2014, pp. 66–78.

[14] S. J. Leemans, F. M. Maggi, and M. Montali, "Enjoy the silence: Analysis of stochastic petri nets with silent transitions," *Inf. Sys.*, vol. 124, p. 102383, 2024.

[15] A. Polyvyanyy, A. Moffat, and L. García-Bañuelos, "An entropic relevance measure for stochastic conformance checking in process mining," in *ICPM*, 2020, pp. 97–104.

[16] P. Cry, A. Horváth, P. Ballarini, and P. L. Gall, "A framework for optimisation based stochastic process discovery," 2024.

[17] S. Boyd and L. Vandenberghe, *Convex Optimization*, 7th ed., 2004.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[19] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer, "Pot: Python optimal transport," *J. Mach. Learn. Res.*, vol. 22, no. 78, pp. 1–8, 2021.

[20] M. de Leoni and F. Mannhardt, "Road traffic fine management process," 2015.

[21] F. Mannhardt, "Sepsis cases - event log," 2016.

[22] B. F. van Dongen and F. Borchert, "Bpi challenge 2018," 2018.

[23] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. P. van der Aalst, "Data-driven process discovery - revealing conditional infrequent behavior from event logs," in *CAiSE*, 2017, pp. 545–560.

[24] S. J. Leemans, E. Poppe, and M. T. Wynn, "Directly follows-based process mining: Exploration & a case study," in *ICPM*, 2019, pp. 25–32.

[25] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and A. Polyvyanyy, "Split miner: automated discovery of accurate and simple business process models from event logs," *Knowl. Inf. Syst.*, vol. 59, pp. 251–284, 2019.

[26] S. J. Leemans, W. M. van der Aalst, T. Brockhoff, and A. Polyvyanyy, "Stochastic process mining: Earth movers' stochastic conformance," *Inf. Syst.*, vol. 102, p. 101724, 2021.