# Computing the Expected Execution Time of Probabilistic Workflow Nets


Philipp J. Meyer(✉), Javier Esparza, and Philip Offtermatt

Technical University of Munich, Munich, Germany
{meyerphi,esparza,offtermp}@in.tum.de

**Abstract.** Free-Choice Workflow Petri nets, also known as Workflow Graphs, are a popular model in Business Process Modeling.

In this paper we introduce Timed Probabilistic Workflow Nets (TPWNs), and give them a Markov Decision Process (MDP) semantics. Since the time needed to execute two parallel tasks is the maximum of the times, and not their sum, the expected time cannot be directly computed using the theory of MDPs with rewards. In our first contribution, we overcome this obstacle with the help of "earliest-first" schedulers, and give a single exponential-time algorithm for computing the expected time.

In our second contribution, we show that computing the expected time is #P-hard, and so polynomial algorithms are very unlikely to exist. Further, #P-hardness holds even for workflows with a very simple structure in which all transitions times are 1 or 0, and all probabilities are 1 or 0.5.

Our third and final contribution is an experimental investigation of the runtime of our algorithm on a set of industrial benchmarks. Despite the negative theoretical results, the results are very encouraging. In particular, the expected time of every workflow in a popular benchmark suite with 642 workflow nets can be computed in milliseconds. Data or code related to this paper is available at: [24].

## 1 Introduction

Workflow Petri Nets are a popular model for the representation and analysis of business processes [1,3,7]. They are used as back-end for different notations like BPMN (Business Process Modeling Notation), EPC (Event-driven Process Chain), and UML Activity Diagrams.

The project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 787367 (PaVeS). Further it is partially supported by the DFG Project No. 273811150 (Negotiations: A Model for Tractable Concurrency).



T. Vojnar and L. Zhang (Eds.): TACAS 2019, Part II, LNCS 11428, pp. 154–171, 2019.
https://doi.org/10.1007/978-3-030-17465-1_9

There is recent interest in extending these notations with quantitative information, like probabilities, costs, and time. The final goal is the development of tool support for computing performance metrics, like the average cost or the average runtime of a business process.

In a former paper we introduced Probabilistic Workflow Nets (PWN), a foundation for the extension of Petri nets with probabilities and rewards [11]. We presented a polynomial time algorithm for the computation of the expected cost of free-choice workflow nets, a subclass of PWN of particular interest for the workflow process community (see e.g. [1,10,13,14]). For example, 1386 of the 1958 nets in the most popular benchmark suite in the literature are free-choice Workflow Nets [12].

In this paper we introduce Timed PWNs (TPWNs), an extension of PWNs with time. Following [11], we define a semantics in terms of Markov Decision Processes (MDPs), where, loosely speaking, the nondeterminism of the MDP models absence of information about the order in which concurrent transitions are executed. For every scheduler, the semantics assigns to the TPWN an expected time to termination. Using results of [11], we prove that this expected time is actually independent of the scheduler, and so that the notion "expected time of a TPWN" is well defined.

We then proceed to study the problem of computing the expected time of a sound TPWN (loosely speaking, of a TPWN that terminates successfully with probability 1). The expected cost and the expected time have a different interplay with concurrency. The cost of executing two tasks in parallel is the sum of the costs (cost models e.g. salaries of power consumption), while the execution time of two parallel tasks is the maximum of their individual execution times. For this reason, standard reward-based algorithms for MDPs, which assume additivity of the reward along a path, cannot be applied.

Our solution to this problem uses the fact that the expected time of a TPWN is independent of the scheduler. We define an "earliest-first" scheduler which, loosely speaking, resolves the nondeterminism of the MDP by picking transitions with earliest possible firing time. Since at first sight the scheduler needs infinite memory, its corresponding Markov chain is infinite-state, and so of no help. However, we show how to construct another finite-state Markov chain with additive rewards, whose expected reward is equal to the expected time of the infinite-state chain. This finite-state Markov chain can be exponentially larger than the TPWN, and so our algorithm has exponential complexity. We prove that computing the expected time is #P-hard, even for free-choice TPWNs in which all transitions times are either 1 or 0, and all probabilities are 1 or $1/2$. So, in particular, the existence of a polynomial algorithm implies $\mathsf{P} = \mathsf{NP}$.

In the rest of the paper we show that, despite these negative results, our algorithm behaves well in practice. For all 642 sound free-choice nets of the benchmark suite of [12], computing the expected time never takes longer than a few milliseconds. Looking for a more complicated set of examples, we study a TPWN computed from a set of logs by process mining. We observe that the computation of the expected time is sensitive to the distribution of the execution

time of a task. Still, our experiments show that even for complicated distributions leading to TPWNs with hundreds of transitions and times spanning two orders of magnitude the expected time can be computed in minutes.

All missing proofs can be found in the Appendix of the full version [19].

## 2 Preliminaries

We introduce some preliminary definitions. The full version [19] gives more details.

**Workflow Nets.** A *workflow net* is a tuple $\mathbf{N} = (P, T, F, i, o)$ where $P$ and $T$ are disjoint finite sets of *places* and *transitions*; $F \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs*; $i, o \in P$ are distinguished *initial* and *final* places such that $i$ has no incoming arcs, $o$ has no outgoing arcs, and the graph $(P \cup T, F \cup \{(o, i)\})$ is strongly connected. For $x \in P \cup T$, we write ${}^\bullet x$ for the set $\{y \mid (y, x) \in F\}$ and $x^\bullet$ for $\{y \mid (x, y) \in F\}$. We call ${}^\bullet x$ (resp. $x^\bullet$) the *preset* (resp. *postset*) of $x$. We extend this notion to sets $X \subseteq P \cup T$ by ${}^\bullet X \stackrel{\text{def}}{=} \cup_{x \in X} {}^\bullet x$ resp. $X^\bullet \stackrel{\text{def}}{=} \cup_{x \in X} x^\bullet$. The notions of marking, enabled transitions, transition firing, firing sequence, and reachable marking are defined as usual. The *initial marking* (resp. *final marking*) of a workflow net, denoted by $\boldsymbol{i}$ (resp. $\boldsymbol{o}$), has one token on place $i$ (resp. $o$), and no tokens elsewhere. A firing sequence $\sigma$ is a *run* if $\boldsymbol{i} \xrightarrow{\sigma} \boldsymbol{o}$, i.e. if it leads to the final marking. $Run_{\mathbf{N}}$ denotes the set of all runs of $\mathbf{N}$.

**Soundness and 1-safeness.** Well designed workflows should be free of deadlocks and livelocks. This idea is captured by the notion of soundness [1,2]: A workflow net is *sound* if the final marking is reachable from any reachable marking.[1] Further, in this paper we restrict ourselves to 1-safe workflows: A marking $M$ of a workflow net $\mathcal{W}$ is *1-safe* if $M(p) \leq 1$ for every place $p$, and $\mathcal{W}$ itself is *1-safe* if every reachable marking is 1-safe. We identify 1-safe markings $M$ with the set $\{p \in P \mid M(p) = 1\}$.

**Independence, concurrency, conflict** [22]**.** Two transitions $t_1$, $t_2$ of a workflow net are *independent* if ${}^\bullet t_1 \cap {}^\bullet t_2 = \emptyset$, and *dependent* otherwise. Given a 1-safe marking $M$, two transitions are *concurrent at* $M$ if $M$ enables both of them, and they are independent, and *in conflict at* $M$ if $M$ enables both of them, and they are dependent. Finally, we recall the definition of Mazurkiewicz equivalence. Let $\mathbf{N} = (P, T, F, i, o)$ be a 1-safe workflow net. The relation $\equiv_1 \subseteq T^* \times T^*$ is defined as follows: $\sigma \equiv_1 \tau$ if there are independent transitions $t_1, t_2$ and sequences $\sigma', \sigma'' \in T^*$ such that $\sigma = \sigma' t_1 t_2 \sigma''$ and $\tau = \sigma' t_2 t_1 \sigma''$. Two sequences $\sigma, \tau \in T^*$ are *Mazurkiewicz equivalent* if $\sigma \equiv \tau$, where $\equiv$ is the reflexive and transitive closure of $\equiv_1$. Observe that $\sigma \in T^*$ is a firing sequence iff every sequence $\tau \equiv \sigma$ is a firing sequence.

**Confusion-freeness, free-choice workflows.** Let $t$ be a transition of a workflow net, and let $M$ be a 1-safe marking that enables $t$. The *conflict set of* $t$

[1] In [2], which examines many different notions of soundness, this is called *e*asy soundness.

*at* $M$, denoted $C(t, M)$, is the set of transitions in conflict with $t$ at $M$. A set $U$ of transitions is a *conflict set* of $M$ if there is a transition $t$ such that $U = C(t, M)$. The conflict sets of $M$ are given by $\mathcal{C}(M) \stackrel{\text{def}}{=} \cup_{t \in T} C(t, M)$. A 1-safe workflow net is *confusion-free* if for every reachable marking $M$ and every transition $t$ enabled at $M$, every transition $u$ concurrent with $t$ at $M$ satisfies $C(u, M) = C(u, M \setminus {}^\bullet t) = C(u, (M \setminus {}^\bullet t) \cup t^\bullet)$. The following result follows easily from the definitions (see also [11]):

**Lemma 1** [11]. *Let* $\mathbf{N}$ *be a 1-safe workflow net. If* $\mathbf{N}$ *is confusion-free then for every reachable marking* $M$ *the conflict sets* $\mathcal{C}(M)$ *are a partition of the set of transitions enabled at* $M$*.*

A workflow net is *free-choice* if for every two places $p_1, p_2$, if $p_1^\bullet \cap p_2^\bullet \neq \emptyset$, then $p_1^\bullet = p_2^\bullet$. Any free-choice net is confusion-free, and the conflict set of a transition $t$ enabled at a marking $M$ is given by $C(t, M) = ({}^\bullet t)^\bullet$ (see e.g. [11]).

## 3 Timed Probabilistic Workflow Nets

In [11] we introduced a probabilistic semantics for confusion-free workflow nets. Intuitively, at every reachable marking a choice between two concurrent transitions is resolved nondeterministically by a scheduler, while a choice between two transitions in conflict is resolved probabilistically; the probability of choosing each transition is proportional to its *weight*. For example, in the net in Fig. 1a, at the marking $\{p_1, p_3\}$, the scheduler can choose between the conflict sets $\{t_2, t_3\}$ and $\{t_4\}$, and if $\{t_2, t_3\}$ is chosen, then $t_2$ is chosen with probability $1/5$ and $t_3$ with probability $4/5$. We extend Probabilistic Workflow Nets by assigning to each transition $t$ a natural number $\tau(t)$ modeling the time it takes for the transition to fire, once it has been selected.[2]

**Definition 1 (Timed Probabilistic Workflow Nets).** *A* Timed Probabilistic Workflow Net *(TPWN) is a tuple* $\mathcal{W} = (\mathbf{N}, w, \tau)$ *where* $\mathbf{N} = (P, T, F, i, o)$ *is a 1-safe confusion-free workflow net,* $w \colon T \to \mathbb{Q}_{>0}$ *is a* weight function*, and* $\tau \colon T \to \mathbb{N}$ *is a* time function *that assigns to every transition a duration.*

**Timed sequences.** We assign to each transition sequence $\sigma$ of $\mathcal{W}$ and each place $p$ a *timestamp* $\mu(\sigma)_p$ through a *timestamp function* $\mu : T^* \to \mathbb{N}_\perp^P$. The set $\mathbb{N}_\perp$ is defined by $\mathbb{N}_\perp \stackrel{\text{def}}{=} \{\perp\} \cup \mathbb{N}$ with $\perp \leq x$ and $\perp + x = \perp$ for all $x \in \mathbb{N}_\perp$. Intuitively, if a place $p$ is marked after $\sigma$, then $\mu(\sigma)_p$ records the "arrival time" of the token in $p$, and if $p$ is unmarked, then $\mu(\sigma)_p = \perp$. When a transition occurs, it removes all tokens in its preset, and $\tau(t)$ time units later, puts tokens into its postset.

[2] The semantics of the model can be defined in the same way for both discrete and continuous time, but, since our results only concern discrete time, we only consider this case.

Formally, we define $\mu(\epsilon)_i \stackrel{\text{def}}{=} 0$, $\mu(\epsilon)_p \stackrel{\text{def}}{=} \bot$ for $p \neq i$, and $\mu(\sigma t) \stackrel{\text{def}}{=} upd(\mu(\sigma), t)$, where the update function $upd : \mathbb{N}_\bot^P \times T \to \mathbb{N}_\bot^P$ is given by:

$$upd(\boldsymbol{x}, t)_p \stackrel{\text{def}}{=} \begin{cases} \max_{q \in {}^\bullet t} \boldsymbol{x}_q + \tau(t) & \text{if } p \in t^\bullet \\ \bot & \text{if } p \in {}^\bullet t \setminus t^\bullet \\ \boldsymbol{x}_p & \text{if } p \notin {}^\bullet t \cup t^\bullet \end{cases}$$

We then define $tm(\sigma) \stackrel{\text{def}}{=} \max_{p \in P} \mu(\sigma)_p$ as the time needed to fire $\sigma$. Further $[\![\boldsymbol{x}]\!] \stackrel{\text{def}}{=} \{p \in P \mid \boldsymbol{x}_p \neq \bot\}$ is the marking represented by a timestamp $\boldsymbol{x} \in \mathbb{N}_\bot^P$.

*Example 1.* The net in Fig. 1a is a TPWN. Weights are shown in red next to transitions, and times are written in blue into the transitions. For the sequence $\sigma_1 = t_1 t_3 t_4 t_5$, we have $tm(\sigma_1) = 9$, and for $\sigma_2 = t_1 t_2 t_3 t_4 t_5$, we have $tm(\sigma_2) = 10$. Observe that the time taken by the sequences is *not* equal to the sum of the durations of the transitions.

**Markov Decision Process semantics.** A *Markov Decision Process* (MDP) is a tuple $\mathcal{M} = (Q, q_0, Steps)$ where $Q$ is a finite set of states, $q_0 \in Q$ is the initial state, and $Steps \colon Q \to 2^{dist(Q)}$ is the probability transition function. Paths of an MDP, schedulers, and the probability measure of paths compatible with a scheduler are defined as usual (see the Appendix of the full version [19]).

The semantics of a TPWN $\mathcal{W}$ is a Markov Decision Process $MDP_\mathcal{W}$. The states of $MDP_\mathcal{W}$ are either markings $M$ or pairs $(M, t)$, where $t$ is a transition enabled at $M$. The intended meanings of $M$ and $(M, t)$ are "the current marking is $M$", and "the current marking is $M$, and $t$ has been selected to fire next." Intuitively, $t$ is chosen in two steps: first, a conflict set enabled at $M$ is chosen nondeterministically, and then a transition of this set is chosen at random, with probability proportional to its weight.

Formally, let $\mathcal{W} = (\mathbf{N}, w, \tau)$ be a TPWN where $\mathbf{N} = (P, T, F, i, o)$, let $M$ be a reachable marking of $\mathcal{W}$ enabling at least one transition, and let $C$ be a conflict set of $M$. Let $w(C)$ be the sum of the weights of the transitions in $C$. The *probability distribution* $P_{M,C}$ *over* $T$ is given by $P_{M,C}(t) = \frac{w(t)}{w(C)}$ if $t \in C$ and $P_{M,C}(t) = 0$ otherwise. Now, let $\mathcal{M}$ be the set of 1-safe markings of $\mathcal{W}$, and let $\mathcal{E}$ be the set of pairs $(M, t)$ such that $M \in \mathcal{M}$ and $M$ enables $t$. We define the Markov decision process $MDP_\mathcal{W} = (Q, q_0, Steps)$, where $Q = \mathcal{M} \cup \mathcal{E}$, $q_0 = \mathbf{i}$, the initial marking of $\mathcal{W}$, and $Steps(M)$ is defined for markings of $\mathcal{M}$ and $\mathcal{E}$ as follows. For every $M \in \mathcal{M}$,

- if $M$ enables no transitions, then $Steps(M)$ contains exactly one distribution, which assigns probability 1 to $M$, and 0 to all other states.
- if $M$ enables at least one transition, then $Steps(M)$ contains a distribution $\lambda$ for each conflict set $C$ of $M$. The distribution is defined by: $\lambda(M, t) = P_{M,C}(t)$ for every $t \in C$, and $\lambda(s) = 0$ for every other state $s$.

For every $(M, t) \in \mathcal{E}$, $Steps(M, t)$ contains one single distribution that assigns probability 1 to the marking $M'$ such that $M \xrightarrow{t} M'$, and probability 0 to every other state.

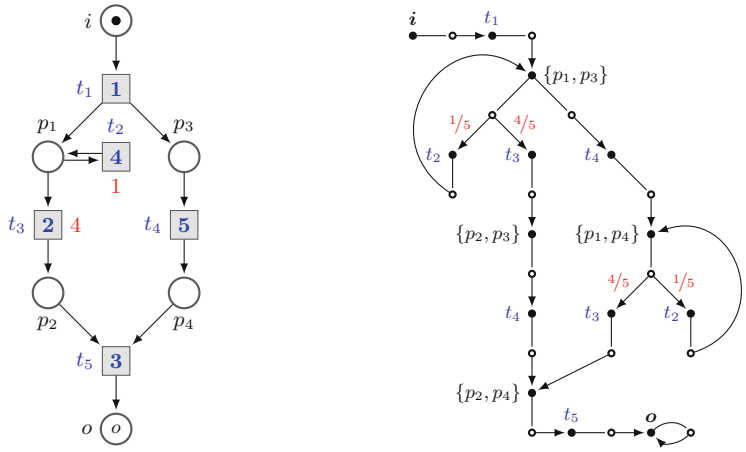



**Fig. 1.** A TPWN and its associated MDP. (Color figure online)

*Example 2.* Figure 1b shows a graphical representation of the MDP of the TPWN in Fig. 1a. Black nodes represent states, white nodes probability distributions. A black node $q$ has a white successor for each probability distribution in $Steps(q)$. A white node $\lambda$ has a black successor for each node $q$ such that $\lambda(q) > 0$; the arrow leading to this black successor is labeled with $\lambda(q)$, unless $\lambda(q) = 1$, in which case there is no label. States $(M, t)$ are abbreviated to $t$.

**Schedulers.** Given a TPWN $\mathcal{W}$, a scheduler of $MDP_W$ is a function $\gamma : T^* \to 2^T$ assigning to each firing sequence $\boldsymbol{i} \xrightarrow{\sigma} M$ with $\mathcal{C}(M) \neq \emptyset$ a conflict set $\gamma(\sigma) \in \mathcal{C}(M)$. A firing sequence $\boldsymbol{i} \xrightarrow{\sigma} M$ is *compatible* with a scheduler $\gamma$ if for all partitions $\sigma = \sigma_1 t \sigma_2$ for some transition $t$, we have $t \in \gamma(\sigma_1)$.

*Example 3.* In the TPWN of Fig. 1a, after firing $t_1$ two conflict sets become concurrently enabled: $\{t_2, t_3\}$ and $\{t_4\}$. A scheduler picks one of the two. If the scheduler picks $\{t_2, t_3\}$ then $t_2$ may occur, and in this case, since firing $t_2$ does not change the marking, the scheduler chooses again one of $\{t_2, t_3\}$ and $\{t_4\}$. So there are infinitely many possible schedulers, differing only in how many times they pick $\{t_2, t_3\}$ before picking $t_4$.

**Definition 2 ((Expected) Time until a state is reached).** *Let $\pi$ be an infinite path of $MDP_{\mathcal{W}}$, and let $M$ be a reachable marking of $\mathcal{W}$. Observe that $M$ is a state of $MDP_{\mathcal{W}}$. The* time needed to reach $M$ along $\pi$, *denoted $tm(M, \pi)$, is defined as follows: If $\pi$ does not visit $M$, then $tm(M, \pi) \stackrel{def}{=} \infty$; otherwise, $tm(M, \pi) \stackrel{def}{=} tm(\Sigma(\pi'))$, where $\Sigma(\pi')$ is the transition sequence corresponding to the shortest prefix $\pi'$ of $\pi$ ending at $M$. Given a scheduler $S$, the expected time until reaching $M$ is defined as*

$$ET^S_{\mathcal{W}}(M) \stackrel{def}{=} \sum_{\pi \in Paths^S} tm(M,\pi) \cdot Prob^S(\pi).$$

*and the expected time* $ET^S_{\mathcal{W}}$ *is defined as* $ET^S_{\mathcal{W}} \stackrel{def}{=} ET^S_{\mathcal{W}}(\boldsymbol{o})$*, i.e. the expected time until reaching the final marking.*

In [11] we proved a result for Probabilistic Workflow Nets (PWNs) with rewards, showing that the expected reward of a PWN is independent of the scheduler (intuitively, this is the case because in a confusion-free Petri net the scheduler only determines the logical order in which transitions occur, but not which transitions occur). Despite the fact that, contrary to rewards, the execution time of a firing sequence is not the sum of the execution times of its transitions, the proof carries over to the expected time with only minor modifications.

**Theorem 1.** *Let* $\mathcal{W}$ *be a TPWN.*

*(1) There exists a value* $ET_{\mathcal{W}}$ *such that for every scheduler* $S$ *of* $\mathcal{W}$*, the expected time* $ET^S_{\mathcal{W}}$ *of* $\mathcal{W}$ *under* $S$ *is equal to* $ET_{\mathcal{W}}$*.*
*(2)* $ET_{\mathcal{W}}$ *is finite iff* $\mathcal{W}$ *is sound.*

By this theorem, the expected time $ET_{\mathcal{W}}$ can be computed by choosing a suitable scheduler $S$, and computing $ET^S_{\mathcal{W}}$.

## 4 Computation of the Expected Time

We show how to compute the expected time of a TPWN. We fix an appropriate scheduler, show that it induces a finite-state Markov chain, define an appropriate reward function for the chain, and prove that the expected time is equal to the expected reward.

### 4.1 Earliest-First Scheduler

Consider a firing sequence $\boldsymbol{i} \xrightarrow{\sigma} M$. We define the *starting time* of a conflict set $C \in \mathcal{C}(M)$ as the earliest time at which the transitions of $C$ become enabled. This occurs after *all* tokens of ${}^\bullet C$ arrive[3], and so the starting time of $C$ is the maximum of $\mu(\sigma)_p$ for $p \in {}^\bullet C$ (recall that $\mu(\sigma)_p$ is the latest time at which a token arrives at $p$ while firing $\sigma$).

Intuitively, the "earliest-first" scheduler always chooses the conflict set with the earliest starting time (if there are multiple such conflict sets, the scheduler chooses any one of them). Formally, recall that a scheduler is a mapping $\gamma\colon T^* \to 2^T$ such that for every firing sequence $\boldsymbol{i} \xrightarrow{\sigma} M$, the set $\gamma(\sigma)$ is a conflict set of $M$. We define the *earliest-first scheduler* $\gamma$ by:

$$\gamma(\sigma) \stackrel{\text{def}}{=} \arg\min_{C \in \mathcal{C}(M)} \max_{p \in {}^\bullet C} \mu(\sigma)_p \qquad \text{where } M \text{ is given by } \boldsymbol{i} \xrightarrow{\sigma} M.$$

[3] This is proved in Lemma 7 in the Appendix of the full version [19].

*Example 4.* Figure 2a shows the Markov chain induced by the "earliest-first" scheduler defined above in the MDP of Fig. 1b. Initially we have a token at $\boldsymbol{i}$ with arrival time 0. After firing $t_1$, which takes time 1, we obtain tokens in $p_1$ and $p_3$ with arrival time 1. In particular, the conflict sets $\{t_2, t_3\}$ and $\{t_4\}$ become enabled at time 1. The scheduler can choose any of them, because they have the same starting time. Assume it chooses $\{t_2, t_3\}$. The Markov chain now branches into two transitions, corresponding to firing $t_2$ and $t_3$ with probabilities $1/5$ and $4/5$, respectively. Consider the branch in which $t_2$ fires. Since $t_2$ starts at time 1 and takes 4 time units, it removes the token from $p_1$ at time 1, and adds a new token to $p_1$ with arrival time 5; the token at $p_3$ is not affected, and it keeps its arrival time of 1. So we have $\mu(t_1t_2) = \left\{ {}^{p_1}_{5}, {}^{p_3}_{1} \right\}$ (meaning $\mu(t_1t_2)_{p_1} = 5$, $\mu(t_1t_2)_{p_3} = 1$, and $\mu(t_1t_2)_p = \bot$ otherwise). Now the conflict sets $\{t_2, t_3\}$ and $\{t_4\}$ are enabled again, but with a difference: while $\{t_4\}$ has been enabled since time 1, the set $\{t_2, t_3\}$ is now enabled since time $\mu(t_1t_2)_{p_1} = 5$. The scheduler must now choose $\{t_4\}$, leading to the marking that puts tokens on $p_1$ and $p_4$ with arrival times $\mu(t_1t_2t_4)_{p_1} = 5$ and $\mu(t_1t_2t_4)_{p_4} = 6$. In the next steps the scheduler always chooses $\{t_2, t_3\}$ until $t_5$ becomes enabled. The final marking $\boldsymbol{o}$ can be reached after time 9, through $t_1t_3t_4t_5$ with probability $4/5$, or with times $10 + 4k$ for $k \in \mathbb{N}$, through $t_1t_2t_4t_2^kt_3t_5$ with probability $(1/5)^{k+1} \cdot 4/5$ (the times at which the final marking can be reached are written in blue inside the final states).

Theorem 2 below shows that the earliest-first scheduler only needs finite memory, which is not clear from the definition. The construction is similar to those of [6,15,16]. However, our proof crucially depends on TPWNs being confusion-free.

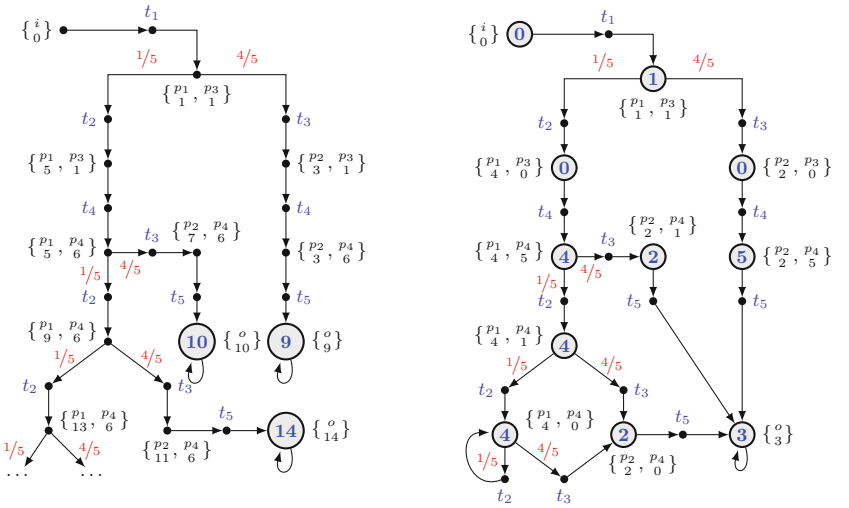
**Theorem 2.** *Let* $H \stackrel{def}{=} \max_{t \in T} \tau(t)$ *be the maximum duration of the transitions of* $T$*, and let* $[H]_\bot \stackrel{def}{=} \{\bot, 0, 1, \ldots, H\} \subseteq \mathbb{N}_\bot$*. There are functions* $\nu \colon T^* \to [H]_\bot^P$ *(compare with* $\mu \colon T^* \to \mathbb{N}_\bot^P$*),* $f \colon [H]_\bot^P \times T \to [H]_\bot^P$ *and* $r \colon [H]_\bot^P \to \mathbb{N}$ *such that for every* $\sigma = t_1 \ldots t_n \in T^*$ *compatible with* $\gamma$ *and for every* $t \in T$ *enabled by* $\sigma$*:*

$$\gamma(\sigma) = \operatorname*{arg\,min}_{C \in \mathcal{C}(\llbracket \nu(\sigma) \rrbracket)} \max_{p \in {}^\bullet C} \nu(\sigma)_p \tag{1}$$

$$\nu(\sigma t) = f(\nu(\sigma), t) \tag{2}$$

$$tm(\sigma) = \max_{p \in P} \nu(\sigma)_p + \sum_{k=0}^{n-1} r(\nu(t_1 \ldots t_k)) \tag{3}$$

Observe that, unlike $\mu$, the range of $\nu$ is finite. We call it the *finite abstraction* of $\mu$. Equation 1 states that $\gamma$ can be computed directly from the finite abstraction $\nu$. Equation 2 shows that $\nu(\sigma t)$ can be computed from $\nu(\sigma)$ and $t$. So $\gamma$ only needs to remember an element of $[H]_\bot^P$, which implies that it only requires finite memory. Finally, observe that the function $r$ of Eq. 3 has a finite domain, and so it allows us to use $\nu$ to compute the time needed by $\sigma$.



(a) Infinite MC for scheduler using $\mu(\sigma)$, with final states labeled by $tm(\sigma)$.

(b) Finite MC for scheduler using $\nu(\sigma)$, with states labeled by rewards $r(\nu(\sigma))$.

**Fig. 2.** Two Markov chains for the "earliest-first" scheduler. (Color figure online)

The formal definition of the functions $\nu$, $f$, and $r$ is given below, together with the definition of the auxiliary operator $\ominus\colon \mathbb{N}_\perp^P \times \mathbb{N} \to \mathbb{N}_\perp^P$:

$$(\boldsymbol{x} \ominus n)_p \stackrel{\text{def}}{=} \begin{cases} \max(\boldsymbol{x}_p - n, 0) & \text{if } \boldsymbol{x}_p \neq \perp \\ \perp & \text{if } \boldsymbol{x}_p = \perp \end{cases} \qquad f(\boldsymbol{x}, t) \stackrel{\text{def}}{=} upd(\boldsymbol{x}, t) \ominus \max_{p \in {}^\bullet t} \boldsymbol{x}_p$$

$$\nu(\epsilon) \stackrel{\text{def}}{=} \mu(\epsilon) \text{ and } \nu(\sigma t) \stackrel{\text{def}}{=} \mu(\sigma t) \ominus \max_{p \in {}^\bullet t} \mu(\sigma)_p \qquad r(\boldsymbol{x}) \stackrel{\text{def}}{=} \min_{C \in \mathcal{C}(\llbracket \boldsymbol{x} \rrbracket)} \max_{p \in {}^\bullet C} \boldsymbol{x}_p$$

*Example 5.* Figure 2b shows the finite-state Markov chain induced by the "earliest-first" scheduler computed using the abstraction $\nu$. Consider the firing sequence $t_1 t_3$. We have $\mu(t_1 t_3) = \left\{ \begin{smallmatrix} p_2 \\ 3 \end{smallmatrix}, \begin{smallmatrix} p_3 \\ 1 \end{smallmatrix} \right\}$, i.e. the tokens in $p_2$ and $p_3$ arrive at times 3 and 1, respectively. Now we compute $\nu(t_1 t_3)$, which corresponds to the *local* arrival times of the tokens, i.e. the time elapsed *since the last transition starts to fire until the token arrives*. Transition $t_3$ starts to fire at time 1, and so the local arrival times of the tokens in $p_2$ and $p_3$ are 2 and 0, respectively, i.e. we have $\nu(t_1 t_3) = \left\{ \begin{smallmatrix} p_2 \\ 2 \end{smallmatrix}, \begin{smallmatrix} p_3 \\ 0 \end{smallmatrix} \right\}$. Using these local times we compute the local starting time of the conflict sets enabled at $\{p_2, p_3\}$. The scheduler always chooses the conflict set with earliest local starting time. In Fig. 2b the earliest local starting time of the state reached by firing $\sigma$, which is denoted $r(\nu(\sigma))$, is written in blue inside the state. The theorem above shows that this scheduler always chooses the same conflict sets as the one which uses the function $\mu$, and that the time of a sequence can be obtained by adding the local starting times. This allows us to consider the earliest local starting time of a state as a *reward*

associated to the state; then, the time taken by a sequence is equal to the sum of the rewards along the corresponding path of the chain. For example, we have $tm(t_1t_2t_4t_3t_5) = 0 + 1 + 0 + 4 + 2 + 3 = 10$.

Finally, let us see how $\nu(\sigma t)$ is computed from $\nu(\sigma)$ for $\sigma = t_1t_2t_4$ and $t = t_2$. We have $\nu(\sigma) = \left\{ \begin{smallmatrix} p_1 \\ 4 \end{smallmatrix}, \begin{smallmatrix} p_4 \\ 5 \end{smallmatrix} \right\}$, i.e. the local arrival times for the tokens in $p_1$ and $p_4$ are 4 and 5, respectively. Now $\{t_2, t_3\}$ is scheduled next, with local starting time $r(\nu(\sigma)) = \nu(\sigma)_{p_1} = 4$. If $t_2$ fires, then, since $\tau(t_2) = 4$, we first add 4 to the time of $p_1$, obtaining $\left\{ \begin{smallmatrix} p_1 \\ 8 \end{smallmatrix}, \begin{smallmatrix} p_4 \\ 5 \end{smallmatrix} \right\}$. Second, we subtract 4 from *all* times, to obtain the time elapsed since $t_2$ started to fire (for local times the origin of time changes every time a transition fires), yielding the final result $\nu(\sigma t_2) = \left\{ \begin{smallmatrix} p_1 \\ 4 \end{smallmatrix}, \begin{smallmatrix} p_4 \\ 1 \end{smallmatrix} \right\}$.

### 4.2 Computation in the Probabilistic Case

Given a TPWN and its corresponding MDP, in the previous section we have defined a finite-state earliest-first scheduler and a reward function of its induced Markov chain. The reward function has the following property: the execution time of a firing sequence compatible with the scheduler is equal to the sum of the rewards of the states visited along it. From the theory of Markov chains with rewards, it follows that the expected accumulated reward until reaching a certain state, provided that this state is reached with probability 1, can be computed by solving a linear equation system. We use this result to compute the expected time $ET_W$.

Let $\mathcal{W}$ be a sound TPWN. For every firing sequence $\sigma$ compatible with the earliest-first scheduler $\gamma$, the finite-state Markov chain induced by $\gamma$ contains a state $\boldsymbol{x} = \nu(\sigma) \in [H]^P_\perp$. Let $C_{\boldsymbol{x}}$ be the conflict set scheduled by $\gamma$ at $\boldsymbol{x}$. We define a system of linear equations with variables $X_{\boldsymbol{x}}$, one for each state $\boldsymbol{x}$:

$$\begin{aligned} X_{\boldsymbol{x}} &= r(\boldsymbol{x}) + \sum_{t \in C_{\boldsymbol{x}}} \frac{w(t)}{w(C_{\boldsymbol{x}})} \cdot X_{f(\boldsymbol{x},t)} && \text{if } [\![\boldsymbol{x}]\!] \neq \boldsymbol{o} \\ X_{\boldsymbol{x}} &= \max_{p \in P} \boldsymbol{x}_p && \text{if } [\![\boldsymbol{x}]\!] = \boldsymbol{o} \end{aligned} \tag{4}$$

The solution of the system is the expected reward of a path leading from $\boldsymbol{i}$ to $\boldsymbol{o}$. By the theory of Markov chains with rewards/costs ([4], Chap. 10.5), we have:

**Lemma 2.** *Let $\mathcal{W}$ be a sound TPWN. Then the system of linear equations (4) has a unique solution $\boldsymbol{X}$, and $ET_{\mathcal{W}} = \boldsymbol{X}_{\nu(\epsilon)}$.*

**Theorem 3.** *Let $\mathcal{W}$ be a TPWN. Then $ET_{\mathcal{W}}$ is either $\infty$ or a rational number and can be computed in single exponential time.*

*Proof.* We assume that the input has size $n$ and all times and weights are given in binary notation. Testing whether $\mathcal{W}$ is sound can be done by exploration of the state space of reachable markings in time $\mathcal{O}(2^n)$. If $\mathcal{W}$ is unsound, we have $ET_{\mathcal{W}} = \infty$.

Now assume that $\mathcal{W}$ is sound. By Lemma 2, $ET_{\mathcal{W}}$ is the solution to the linear equation system (4), which is finite and has rational coefficients, so it is a

rational number. The number of variables $|\boldsymbol{X}|$ of (4) is bounded by the size of $[H]_{\perp}^{P}$, and as $H = \max_{t \in T} \tau(t)$ we have $|\boldsymbol{X}| \leq (1+H)^{|P|} \leq (1+2^n)^n \leq 2^{n^2+n}$. The linear equation system can be solved in time $\mathcal{O}\left(n^2 \cdot |\boldsymbol{X}|^3\right)$ and therefore in time $\mathcal{O}(2^{p(n)})$ for some polynomial $p$.

## 5 Lower Bounds for the Expected Time

We analyze the complexity of computing the expected time of a TPWN. Botezano *et al.* show in [5] that deciding if the expected time exceeds a given bound is NP-hard. However, their reduction produces TPWNs with weights and times of arbitrary size. An open question is if the expected time can be computed in polynomial time when the times (and weights) must be taken from a finite set. We prove that this is not the case unless P = NP, even if all times are 0 or 1, all weights are 1, the workflow net is sound, acyclic and free-choice, and the size of each conflict set is at most 2 (resulting only in probabilities 1 or $1/2$). Further, we show that even computing an $\epsilon$-approximation is equally hard. These two results above are a consequence of the main theorem of this section: computing the expected time is #P-hard [23]. For example, counting the number of satisfying assignments for a boolean formula (#SAT) is a #P-complete problem. Therefore a polynomial-time algorithm for a #P-hard problem would imply P = NP.

The problem used for the reduction is defined on PERT networks [9], in the specialized form of *two-state stochastic PERT networks* [17], described below.

**Definition 3.** *A* two-state stochastic PERT network *is a tuple* $\mathbf{PN} = (G, s, t, \boldsymbol{p})$*, where* $G = (V, E)$ *is a directed acyclic graph with vertices* $V$*, representing events, and edges* $E$*, representing tasks, with a single source vertex* $s$ *and sink vertex* $t$*, and where the vector* $\boldsymbol{p} \in \mathbb{Q}^E$ *assigns to each edge* $e \in E$ *a rational probability* $p_e \in [0, 1]$*. We assume that all* $p_e$ *are written in binary.*

*Each edge* $e \in E$ *of* **PN** *defines a random variable* $X_e$ *with distribution* $\Pr(X_e = 1) = p_e$ *and* $\Pr(X_e = 0) = 1 - p_e$*. All* $X_e$ *are assumed to be independent. The* project duration $PD$ *of* **PN** *is the length of the longest path in the network*

$$PD(\mathbf{PN}) \stackrel{def}{=} \max_{\pi \in \Pi} \sum_{e \in \pi} X_e$$

*where* $\Pi$ *is the set of paths from vertex* $s$ *to vertex* $t$*. As this defines a random variable, the* expected project duration *of* **PN** *is then given by* $\mathbb{E}(PD(\mathbf{PN}))$*.*

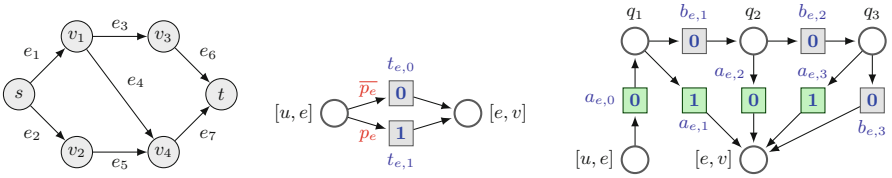
*Example 6.* Figure 3a shows a small PERT network (without $\boldsymbol{p}$), where the project duration depends on the paths $\Pi = \{e_1e_3e_6, e_1e_4e_7, e_2e_5e_7\}$.

The following problem is #P-hard (from [17], using the results from [20]):

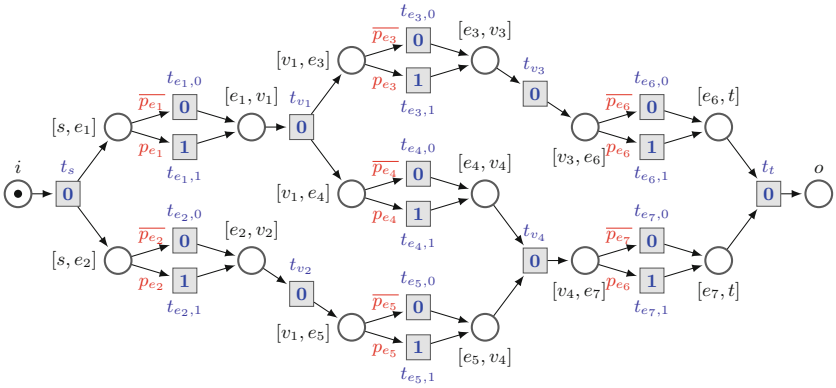
**Given:** A two-state stochastic PERT network **PN**.
**Compute:** The expected project duration $\mathbb{E}(PD(\mathbf{PN}))$.

**First reduction: 0/1 times, arbitrary weights.** We reduce the problem above to computing the expected time of an acyclic TPWN with 0/1 times but arbitrary weights. Given a two-state stochastic PERT network **PN**, we construct a timed probabilistic workflow net $\mathcal{W}_{\mathbf{PN}}$ as follows:

- For each edge $e = (u, v) \in E$, add the "gadget net" shown in Fig. 3b. Assign $w(t_{e,0}) = 1 - p_e$, $w(t_{e,1}) = p_e$, $\tau(t_{e,0}) = 0$, and $\tau(t_{e,1}) = 1$.
- For each vertex $v \in V$, add a transition $t_v$ with arcs from each $[e, v]$ such that $e = (u, v) \in E$ for some $u$ and arcs to each $[v, e]$ such that $e = (v, w) \in E$ for some $w$. Assign $w(t_v) = 1$ and $\tau(t_v) = 0$.
- Add the place $i$ with an arc to $t_s$ and the place $o$ with an arc from $t_t$.




(a) PERT network **PN**. (b) Gadget for $e = (u, v)$ with rational weights $p_e, \overline{p_e}$. (c) Equivalent gadget for $e$ with weights 1 for $p_e = 5/8 = (0.101)_2$.




(d) Timed probabilistic workflow net $\mathcal{W}_{\mathbf{PN}}$.

**Fig. 3.** A PERT network and its corresponding timed probabilistic workflow net. The weight $\overline{p}$ is short for $1 - p$. Transitions without annotations have weight 1.

The result of applying this construction to the PERT network from Fig. 3a is shown in Fig. 3d. It is easy to see that this workflow net is sound, as from any reachable marking, we can fire enabled transitions corresponding to the edges and vertices of the PERT network in the topological order of the graph, eventually firing $t_t$ and reaching $\boldsymbol{o}$. The net is also acyclic and free-choice.

**Lemma 3.** *Let* **PN** *be a two-state stochastic PERT network and let* $\mathcal{W}_{\mathbf{PN}}$ *be its corresponding TPWN by the construction above. Then* $ET_{\mathcal{W}_{\mathbf{PN}}} = \mathbb{E}(PD(\mathbf{PN}))$.

**Second reduction: 0/1 times, 0/1 weights.** The network constructed this way already uses times 0 and 1, however the weights still use arbitrary rational numbers. We now replace the gadget nets from Fig. 3b by equivalent nets where all transitions have weight 1. The idea is to use the binary encoding of the probabilities $p_e$, deciding if the time is 0 or 1 by a sequence of coin flips. We assume that $p_e = \sum_{i=0}^{k} 2^{-i} p_i$ for some $k \in \mathbb{N}$ and $p_i \in \{0, 1\}$ for $0 \leq i \leq k$. The replacement is shown in Fig. 3c for $p_e = 5/8 = (0.101)_2$.

**Approximating the expected time is #P-hard.** We show that computing an $\epsilon$-approximation for $ET_{\mathcal{W}}$ is #P-hard [17,20].

**Theorem 4.** *The following problem is* #P*-hard:*

> **Given:** *A sound, acyclic and free-choice TPWN* $\mathcal{W}$ *where all transitions* $t$ *satisfy* $w(t) = 1$, $\tau(t) \in \{0, 1\}$ *and* $|({}^\bullet t)^\bullet| \leq 2$, *and an* $\epsilon > 0$.
> **Compute:** *A rational* $r$ *such that* $r - \epsilon < ET_W < r + \epsilon$.

## 6 Experimental Evaluation

We have implemented our algorithm to compute the expected time of a TPWN as a package of the tool `ProM`[4]. It is available via the package manager of the latest nightly build under the package name `WorkflowNetAnalyzer`.

We evaluated the algorithm on two different benchmarks. All experiments in this section were run on the same machine equipped with an Intel Core i7-6700K CPU and 32 GB of RAM. We measure the actual runtime of the algorithm, split into construction of the Markov chain and solving the linear equation system, and exclude the time overhead due to starting `ProM` and loading the plugin.

### 6.1 IBM Benchmark

We evaluated the tool on a set of 1386 workflow nets extracted from a collection of five libraries of industrial business processes modeled in the IBM WebSphere Business Modeler [12]. All of the 1386 nets in the benchmark libraries are free-choice and therefore confusion-free. We selected the sound and 1-safe nets among them, which are 642 nets. Out of these, 409 are marked graphs, i.e. the size of any conflict set is 1. Out of the remaining 233 nets, 193 are acyclic and 40 cyclic.

As these nets do not come with probabilistic or time information, we annotated transitions with integer weights and times chosen uniformly from different intervals: (1) $w(t) = \tau(t) = 1$, (2) $w(t), \tau(t) \in [1, 10^3]$ and (3) $w(t), \tau(t) \in [1, 10^6]$. For each interval, we annotated the transitions of each net with random weights and times, and computed the expected time of all 642 nets.

For all intervals, we computed the expected time for any net in less than 50 ms. The analysis time did not differ much for different intervals. The solving time for the linear equation system is on average 5% of the total analysis time,

[4] http://www.promtools.org/.

and at most 68%. The results for the nets with the longest analysis times are given in Table 1. They show that even for nets with a huge state space, thanks to the earliest-first scheduler, only a small number of reachable markings is explored.

**Table 1.** Analysis times and size of the state space $|\boldsymbol{X}|$ for the 4 nets with the highest analysis times, given for each of the three intervals $[1], [10^3], [10^6]$ of possible times. Here, $|\mathcal{R}^N|$ denotes the number of reachable markings of the net.

| Net | Net info & size | | | | Analysis time (ms) | | | $\lvert\boldsymbol{X}\rvert$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | cyclic | $\lvert P\rvert$ | $\lvert T\rvert$ | $\lvert\mathcal{R}^N\rvert$ | $[1]$ | $[10^3]$ | $[10^6]$ | $[1]$ | $[10^3]$ | $[10^6]$ |
| m1.s30_s703 | no | 264 | 286 | 6117 | 40.3 | 44.6 | 43.8 | 304 | 347 | 347 |
| m1.s30_s596 | yes | 214 | 230 | 623 | 21.6 | 24.4 | 23.6 | 208 | 232 | 234 |
| b3.s371_s1986 | no | 235 | 101 | $2 \cdot 10^{17}$ | 16.8 | 16.4 | 16.5 | 101 | 102 | 102 |
| b2.s275_s2417 | no | 103 | 68 | 237626 | 14.2 | 17.8 | 15.9 | 355 | 460 | 431 |

## 6.2 Process Mining Case Study

As a second benchmark, we evaluated the algorithm on a model of a loan application process. We used the data from the BPI Challenge 2017 [8], an event log containing 31509 cases provided by a financial institute, and took as a model of the process the final net from the report of the winner of the academic category [21], a simple model with high fitness and precision w.r.t. the event log.

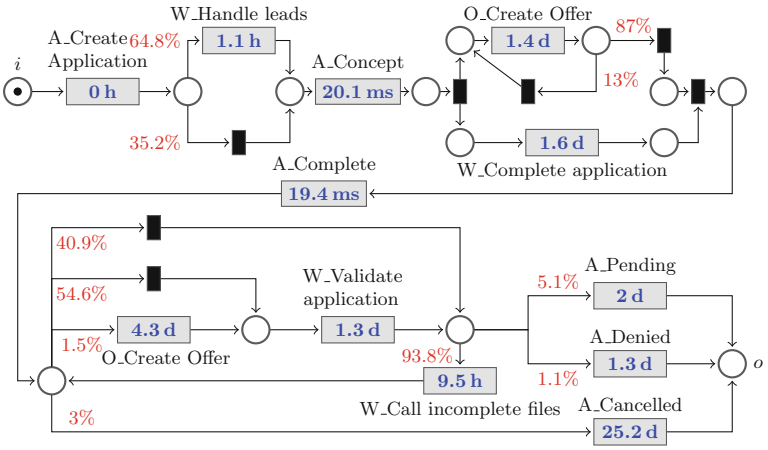



**Fig. 4.** Net from [21] of process for personal loan applications in a financial institute, annotated with mean waiting times and local trace weights. Black transitions are invisible transitions not appearing in the event log with time 0.

**Table 2.** Expected time, analysis time and state space size for the net in Fig. 4 for various distributions, where `memout` denotes reaching the memory limit.

| Distribution | $\lvert T\rvert$ | $ET_W$ | $\lvert \boldsymbol{X}\rvert$ | Analysis time | | |
|---|---|---|---|---|---|---|
| | | | | Total | Construction | Solving |
| Deterministic | 19 | 24 d 1 h | 33 | 40 ms | 18 ms | 22 ms |
| Histogram/12 h | 141 | 24 d 18 h | 4054 | 244 ms | 232 ms | 12 ms |
| Histogram/6 h | 261 | 24 d 21 h | 15522 | 2.1 s | 1.8 s | 0.3 s |
| Histogram/4 h | 375 | 24 d 22 h | 34063 | 10 s | 6 s | 4 s |
| Histogram/2 h | 666 | 24 d 23 h | 122785 | 346 s | 52 s | 294 s |
| Histogram/1 h | 1117 | — | 422614 | — | 12.7 min | `memout` |

Using the `ProM` plugin "Multi-perspective Process Explorer" [18] we annotated each transition with waiting times and each transition in a conflict set with a local percentage of traces choosing this transition when this conflict set is enabled. The net with mean times and weights as percentages is displayed in Fig. 4.

For a first analysis, we simply set the execution time of each transition deterministically to its mean waiting time. However, note that the two transitions "O_Create Offer" and "W_Complete application" are executed in parallel, and therefore the distribution of their execution times influences the total expected time. Therefore we also annotated these two transitions with a histogram of possible execution times from each case. Then we split them up into multiple transitions by grouping the times into buckets of a given interval size, where each bucket creates a transition with an execution time equal to the beginning of the interval, and a weight equal to the number of cases with a waiting time contained in the interval. The times for these transitions range from 6 ms to 31 days. As bucket sizes we chose $12, 6, 4, 2$ and 1 hour(s). The net always has 14 places and 15 reachable markings, but a varying number of transitions depending on the chosen bucket size. For the net with the mean as the deterministic time and for the nets with histograms for each bucket size, we then analyzed the expected execution time using our algorithm.

The results are given in Table 2. They show that using the complete distribution of times instead of only the mean can lead to much more precise results. When the linear equation system becomes very large, the solver time dominates the construction time of the system. This may be because we chose to use an exact solver for sparse linear equation systems. In the future, this could possibly be improved by using an approximative iterative solver.

## 7 Conclusion

We have shown that computing the expected time to termination of a probabilistic workflow net in which transition firings have deterministic durations is

#P-hard. This is the case even if the net is free-choice, and both probabilities and times can be written down with a constant number of bits. So, surprisingly, computing the expected time is much harder than computing the expected cost, for which there is a polynomial algorithm [11].

We have also presented an exponential algorithm for computing the expected time based on earliest-first schedulers. Its performance depends crucially on the maximal size of conflict sets that can be concurrently enabled. In the most popular suite of industrial benchmarks this number turns out to be small. So, very satisfactorily, the expected time of any of these benchmarks, some of which have hundreds of transitions, can still be computed in milliseconds.

**Acknowledgements.** We thank Hagen Völzer for input on the implementation and choice of benchmarks.

## References

1. van der Aalst, W.M.P.: The application of Petri nets to workflow management. J. Circ. Syst. Comput. **8**(1), 21–66 (1998). https://doi.org/10.1142/S0218126698000043
2. van der Aalst, W.M.P., et al.: Soundness of workflow nets: classification, decidability, and analysis. Formal Asp. Comput. **23**(3), 333–363 (2011). https://doi.org/10.1007/s00165-010-0161-4
3. van der Aalst, W., van Hee, K.M.: Workflow Management: Models, Methods, and Systems. MIT Press, Cambridge (2004)
4. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press, Cambridge (2008)
5. Botezatu, M., Völzer, H., Thiele, L.: The complexity of deadline analysis for workflow graphs with multiple resources. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 252–268. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_15
6. Carlier, J., Chrétienne, P.: Timed Petri net schedules. In: Rozenberg, G. (ed.) APN 1987. LNCS, vol. 340, pp. 62–84. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-50580-6_24
7. Desel, J., Erwin, T.: Modeling, simulation and analysis of business processes. In: van der Aalst, W., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 129–141. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45594-9_9
8. van Dongen, B.F.: BPI Challenge 2017 (2017). https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
9. Elmaghraby, S.E.: Activity Networks: Project Planning and Control by Network Models. Wiley, Hoboken (1977)
10. Esparza, J., Hoffmann, P.: Reduction rules for colored workflow nets. In: Stevens, P., Wąsowski, A. (eds.) FASE 2016. LNCS, vol. 9633, pp. 342–358. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49665-7_20
11. Esparza, J., Hoffmann, P., Saha, R.: Polynomial analysis algorithms for free choice probabilistic workflow nets. Perform. Eval. **117**, 104–129 (2017). https://doi.org/10.1016/j.peva.2017.09.006

12. Fahland, D., et al.: Instantaneous soundness checking of industrial business process models. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 278–293. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03848-8_19
13. Favre, C., Fahland, D., Völzer, H.: The relationship between workflow graphs and free-choice workflow nets. Inf. Syst. **47**, 197–219 (2015). https://doi.org/10.1016/j.is.2013.12.004
14. Favre, C., Völzer, H., Müller, P.: Diagnostic information for control-flow analysis of workflow graphs (a.k.a. free-choice workflow nets). In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 463–479. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_27
15. Gaubert, S., Mairesse, J.: Asymptotic analysis of heaps of pieces and application to timed Petri nets. In: Proceedings of the 8th International Workshop on Petri Nets and Performance Models, PNPM 1999, Zaragoza, Spain, 8–10 September 1999, pp. 158–169 (1999). https://doi.org/10.1109/PNPM.1999.796562
16. Gaubert, S., Mairesse, J.: Modeling and analysis of timed Petri nets using heaps of pieces. IEEE Trans. Autom. Control **44**(4), 683–697 (1999). https://doi.org/10.1109/9.754807
17. Hagstrom, J.N.: Computational complexity of PERT problems. Networks **18**(2), 139–147 (1988). https://doi.org/10.1002/net.3230180206
18. Mannhardt, F., de Leoni, M., Reijers, H.A.: The multi-perspective process explorer. In: Proceedings of the BPM Demo Session 2015 Co-located with the 13th International Conference on Business Process Management, BPM 2015, Innsbruck, Austria, 2 September 2015, pp. 130–134 (2015)
19. Meyer, P.J., Esparza, J., Offtermatt, P.: Computing the expected execution time of probabilistic workflow nets. arXiv:1811.06961 [cs.LO] (2018)
20. Provan, J.S., Ball, M.O.: The complexity of counting cuts and of computing the probability that a graph is connected. SIAM J. Comput. **12**(4), 777–788 (1983). https://doi.org/10.1137/0212053
21. Rodrigues, A., et al.: Stairway to value: mining a loan application process (2017). https://www.win.tue.nl/bpi/lib/exe/fetch.php?media=2017:bpi2017_winner_academic.pdf
22. Rozenberg, G., Thiagarajan, P.S.: Petri nets: basic notions, structure, behaviour. In: de Bakker, J.W., de Roever, W.-P., Rozenberg, G. (eds.) Current Trends in Concurrency. LNCS, vol. 224, pp. 585–668. Springer, Heidelberg (1986). https://doi.org/10.1007/BFb0027048
23. Valiant, L.G.: The complexity of computing the permanent. Theoret. Comput. Sci. **8**, 189–201 (1979). https://doi.org/10.1016/0304-3975(79)90044-6
24. Meyer, P.J., Esparza, J., Offtermatt, P.: Artifact and instructions to generate experimental results for TACAS 2019 paper: Computing the Expected Execution Time of Probabilistic Workflow Nets (artifact). Figshare (2019). https://doi.org/10.6084/m9.figshare.7831781.v1