Computing the Expected Execution Time of Probabilistic Workflow Nets

---

Philipp Meyer

TACAS 2019

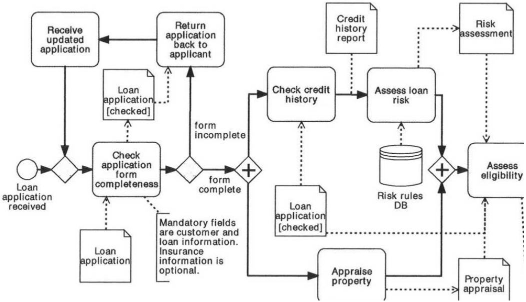
Technical University of Munich

Joint work with Javier Esparza and Philip Offtermatt

Introduction

Workflow nets

- Represent cases, i.e. life-cycles of process instances.
  Used for business processes or healthcare processes.
- Back-end for BPMN, EPC or UML Activity Diagrams.
- Describe tasks of the case and their causal order.
  May have information about task execution costs and times.



Introduction

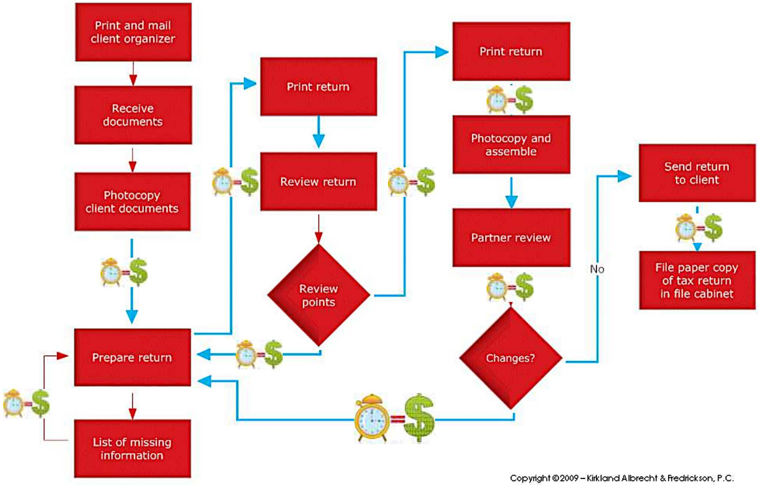
Workflow nets

- Represent cases, i.e. life-cycles of process instances.
  Used for business processes or healthcare processes.
- Back-end for BPMN, EPC or UML Activity Diagrams.
- Describe tasks of the case and their causal order.
  May have information about task execution costs and times.

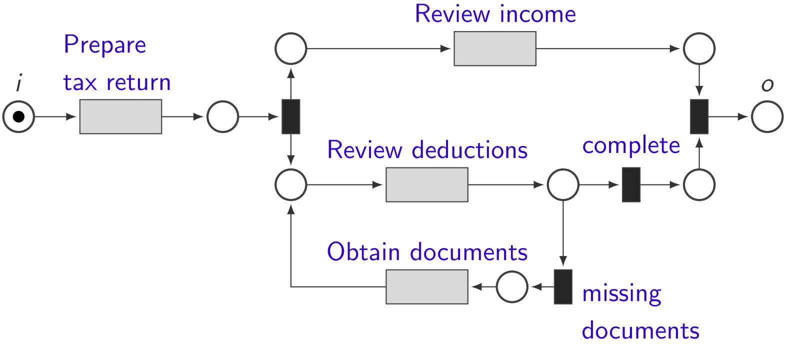
Analysis questions (for time)

- What is the expected time for completion of one case?
- What is the probability meeting a given deadline?

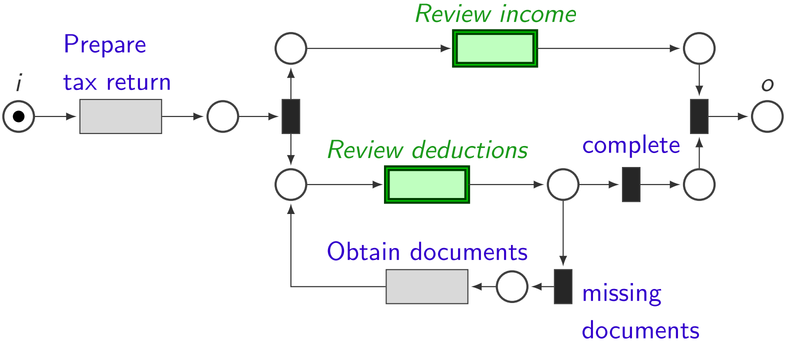
Example: Workflow of a tax return



Example: (Partial) workflow net of a tax return

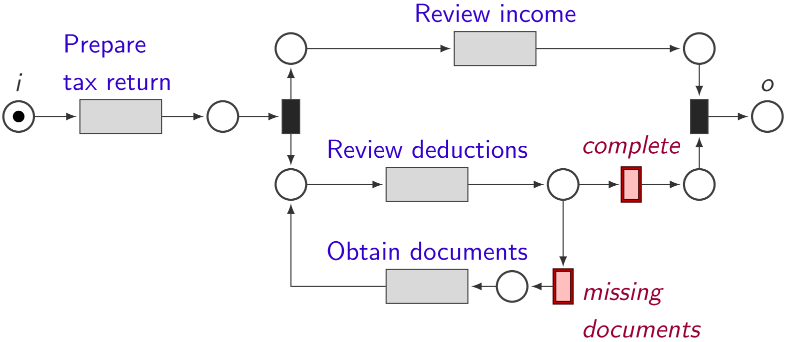


Example: (Partial) workflow net of a tax return



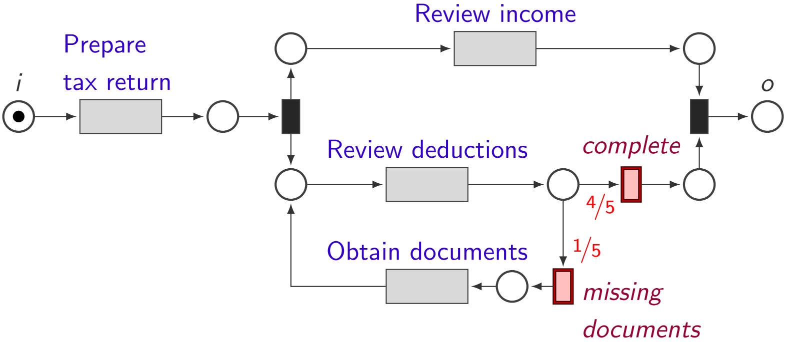
- Review of income and deductions is done *concurrently*.

Example: (Partial) workflow net of a tax return



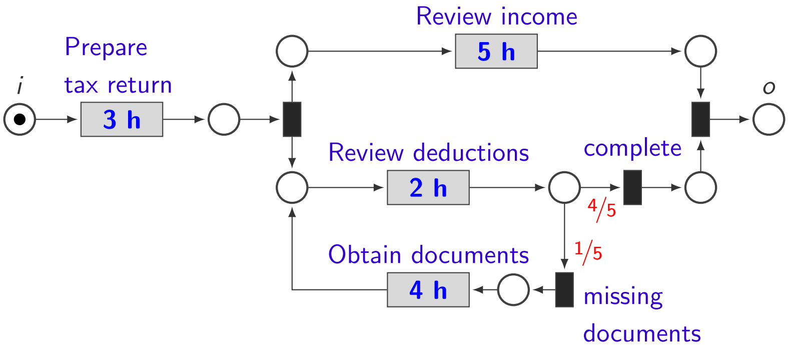
- Review of income and deductions is done *concurrently*.
- After reviewing deductions, there is a *choice*.

Example: (Partial) workflow net of a tax return



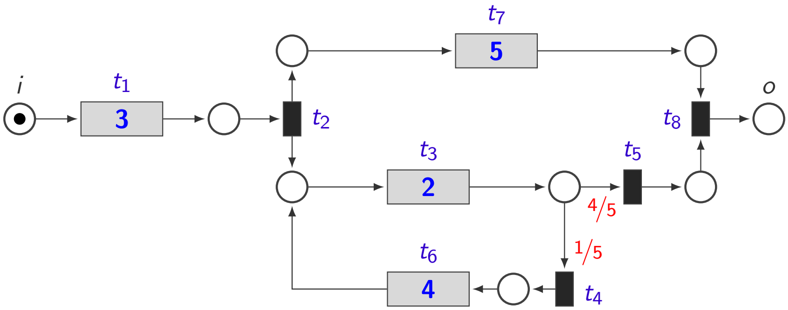
- Review of income and deductions is done *concurrently*.
- After reviewing deductions, there is a *choice*.
- Choice is weighted by *probabilities*.

Example: (Partial) workflow net of a tax return



- Review of income and deductions is done *concurrently*.
- After reviewing deductions, there is a *choice*.
- Choice is weighted by *probabilities*.
- Task transitions have *execution times*.

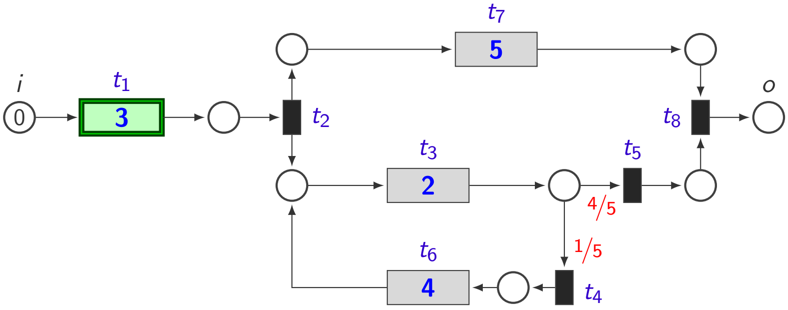
Example: Abstract workflow net of a tax return



Timed Probabilistic Workflow Net (TPWN)

- A *run* of the net is an execution starting in i and ending in o.
- The net is *sound* if every execution eventually ends in o.
- We assume *1-safe* nets, i.e. each place has at most one token.

Example: Run of the workflow net with time

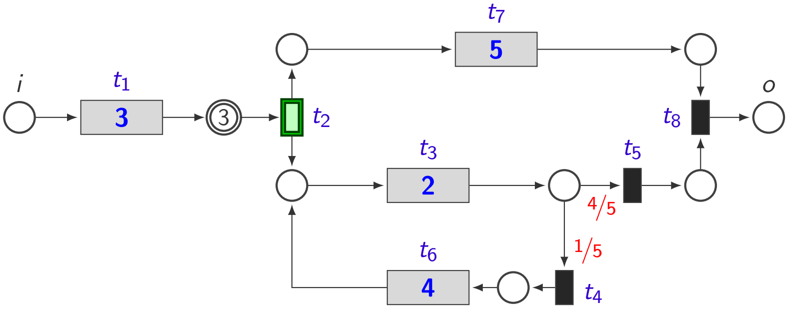


Run:

Probability: 1

Time: **0** 1 2 3 4 5 6 7 8 9 10 11

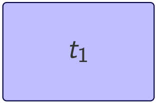
Example: Run of the workflow net with time



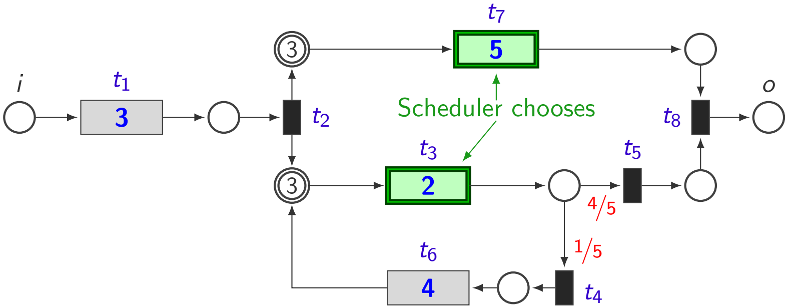
Run: t_1

Probability: 1

Time: 0 1 2 **3** 4 5 6 7 8 9 10 11



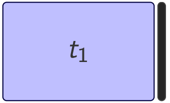
Example: Run of the workflow net with time



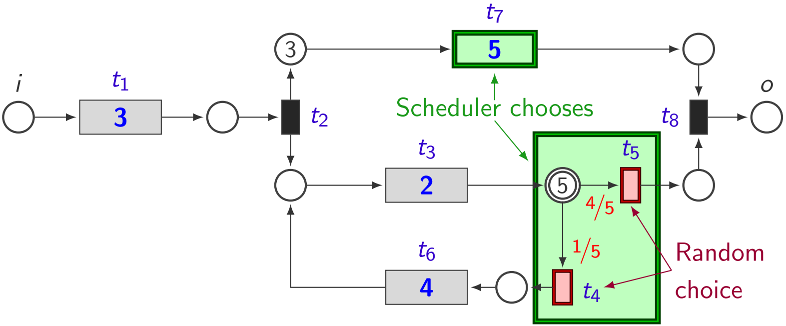
Run: t_1 t_2

Probability: 1

Time: 0 1 2 **3** 4 5 6 7 8 9 10 11



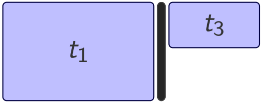
Example: Run of the workflow net with time



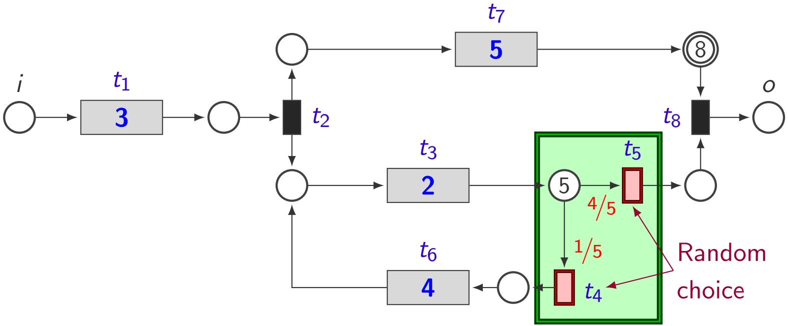
Run: t_1 t_2 t_3

Probability: 1

Time: 0 1 2 3 4 **5** 6 7 8 9 10 11



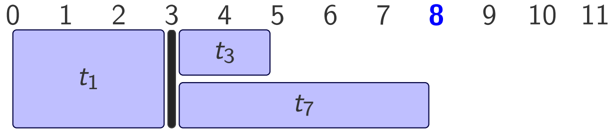
Example: Run of the workflow net with time



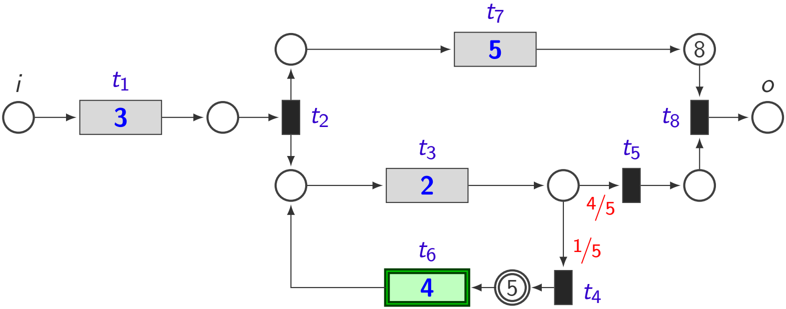
Run: t_1 t_2 t_3 t_7

Probability: 1

Time:

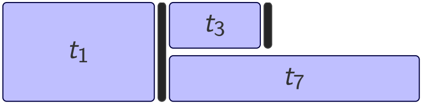


Example: Run of the workflow net with time

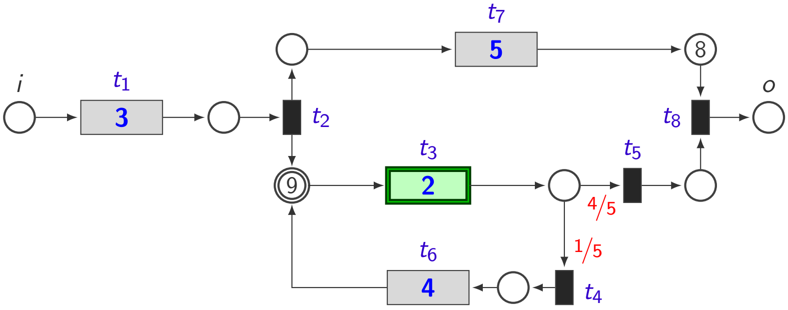


Run: t_1 t_2 t_3 t_7 t_4 Probability: $1/5$

Time: 0 1 2 3 4 5 6 7 **8** 9 10 11

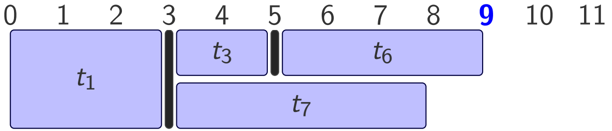


Example: Run of the workflow net with time

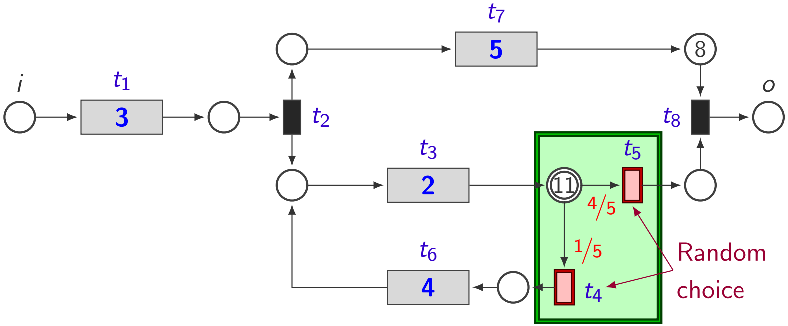


Run: t_1 t_2 t_3 t_7 t_4 t_6 Probability: $1/5$

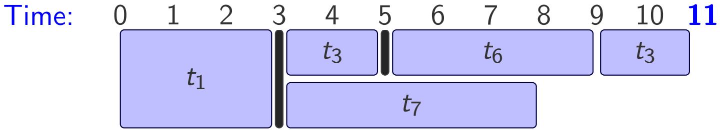
Time:



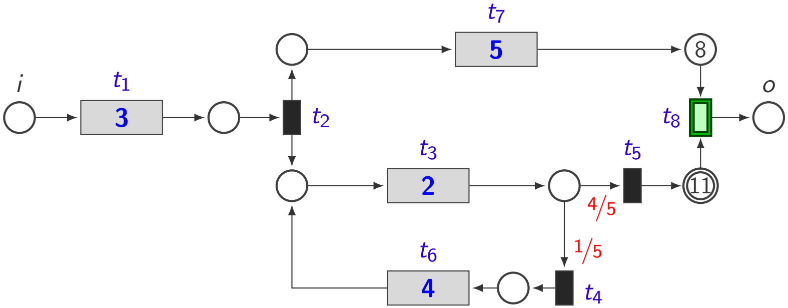
Example: Run of the workflow net with time



Run: t_1 t_2 t_3 t_7 t_4 t_6 t_3 Probability: $^1/_5$

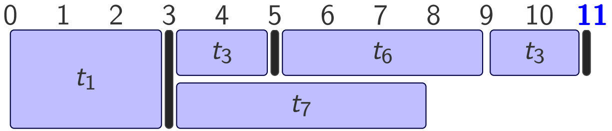


Example: Run of the workflow net with time

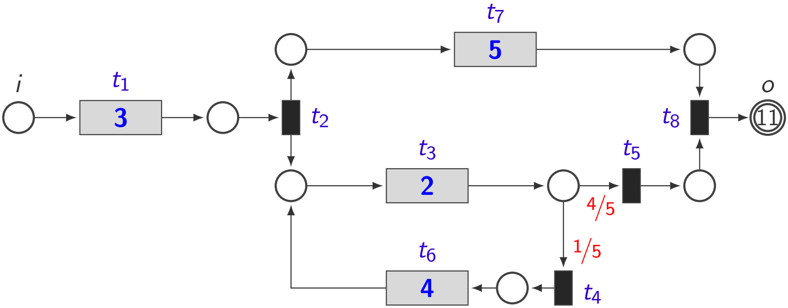


Run: t_1 t_2 t_3 t_7 t_4 t_6 t_3 t_5 Probability: $^1/_5 \cdot {}^4/_5$

Time:

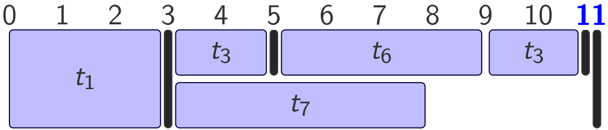


Example: Run of the workflow net with time



Run: t_1 t_2 t_3 t_7 t_4 t_6 t_3 t_5 t_8 Probability: $1/5 \cdot 4/5$

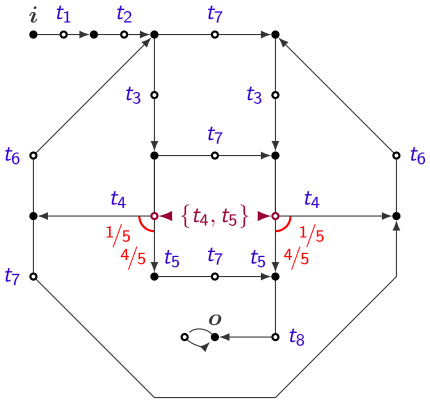
Time:



Semantics of timed probabilistic workflow nets

Semantics of TPWN defined by Markov decision process (MDP):

- Black nodes are markings, white nodes are conflict sets.
- Fixing a scheduler yields a Markov chain.
- *Expected time* then given by exp. time to reach $\boldsymbol{o}$ from $\boldsymbol{i}$.
- Time of executions given by *maximum* of concurrent and *sum* of sequential task times.



Computing the expected time: problems

Problem 1

Expected time may be *dependent* on the scheduler.

Problem 2

Unclear how to compute expected time, even for a fixed scheduler, as times are *not* purely *additive*.

This is in contrast to expected *cost* of a net.

Problem 3 [Botezatu, Völzer, Thiele, BPM'16]

Given a *free-choice* TPWN and a number k, deciding if the expected time exceeds k is *NP-hard* (requires times in *binary*).

Computing the expected time: contributions

Theorem

Given a *confusion-free* TPWN, the expected time is *independent* of the scheduler.

Theorem

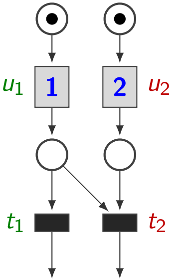
By fixing a certain "earliest-first" scheduler, the expected time can be computed from a finite exponentially-sized Markov chain with *additive times*.

Theorem

Given a *free-choice* TPWN where all *times* are 0 or 1 and all probabilities 1 or $1/2$, computing the expected time is *#P-hard*.

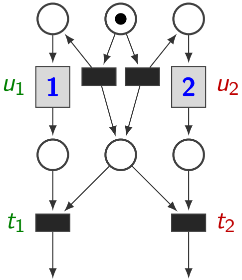
Confusion-free and free-choice nets

Confusion



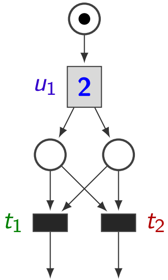
- Difficulty in resolving conflicts.
- Several semantics for time, unintuitive.

Confusion-free



- No interference of concurrency and conflicts.
- Semantic property, PSPACE-hard.

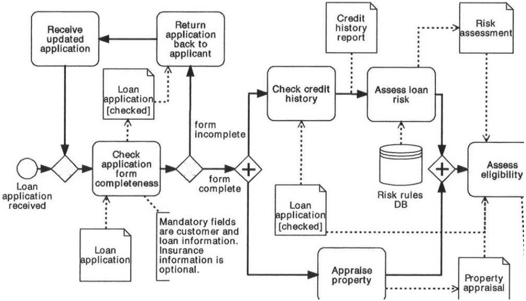
Free-choice

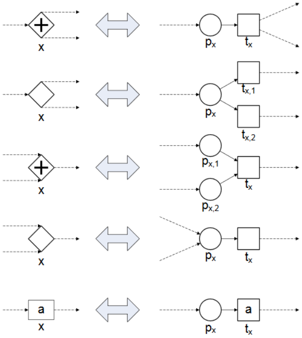


- Syntactic property.
- Implies confusion-freeness.

Free-choice workflow nets

- *Workflow graphs* are the core of BPNM 2.0 and translate into (and are essentially equivalent to) *free-choice workflow nets*.
- Of *2000* workflow nets (IBM, SAP): almost *1400* are free-choice.
- Many properties of free-choice workflow nets decidable in *polynomial time*: soundness, reachability, expected cost, . . .





Independence of the scheduler

Theorem

Given a confusion-free TPWN, the expected time is independent of the scheduler.

Further, the expected time is finite iff the net is sound.

Proof.

By adapting proof of independence of scheduler for expected cost [Esparza, Hoffmann, Saha, Perform. Eval. '17]. □

- We can fix a scheduler to obtain a Markov chain.
- Still unclear how to compute expected time from chain.

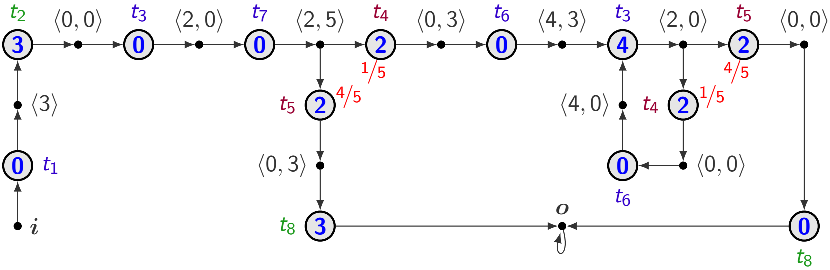
Computing the expected time

Theorem

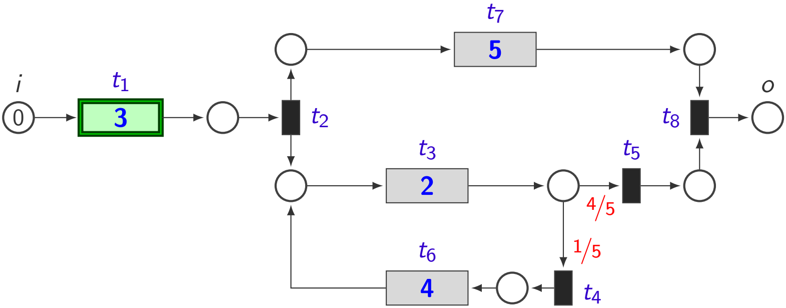
Given a confusion-free TPWN, the expected time can be computed in single exponential time.

Proof.

By "earliest-first" scheduler with finite memory yielding an exponentially-sized Markov chain with local additive times. □



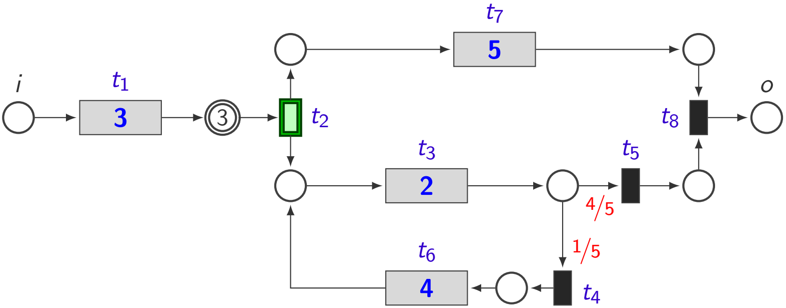
Example: Run of "earliest-first" scheduler



Run:

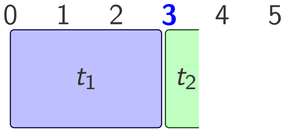
Time:

Example: Run of "earliest-first" scheduler

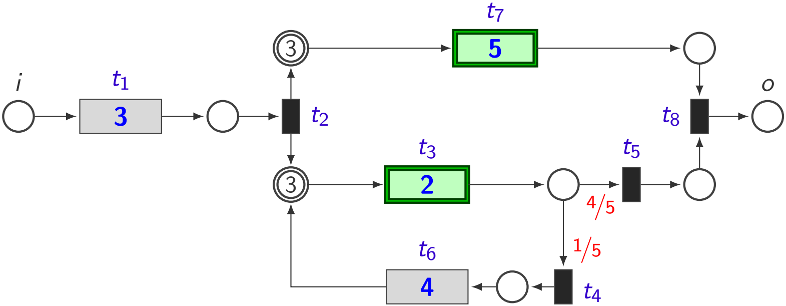


Run: t_1

Time: **0**

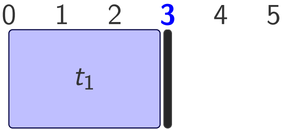


Example: Run of "earliest-first" scheduler

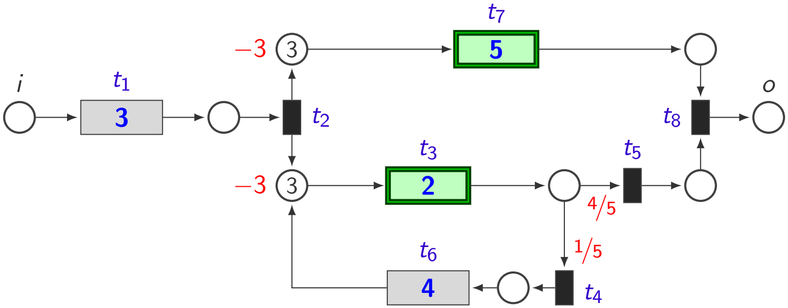


Run: t_1 t_2

Time: $\mathbf{0}+\mathbf{3}$

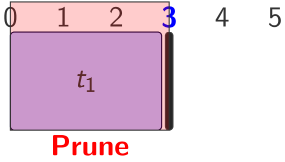


Example: Run of "earliest-first" scheduler

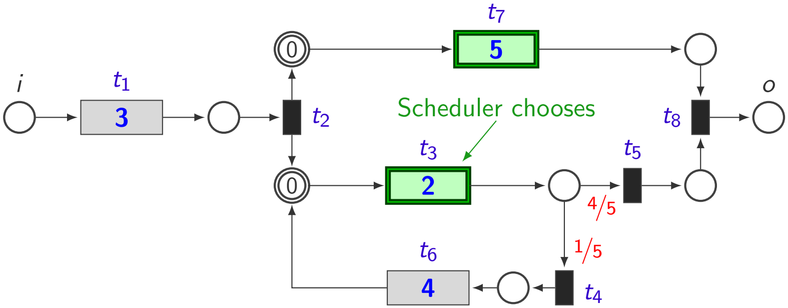


Run: t_1 t_2

Time: $\mathbf{0}+\mathbf{3}$

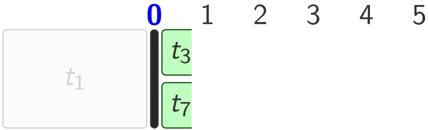


Example: Run of "earliest-first" scheduler

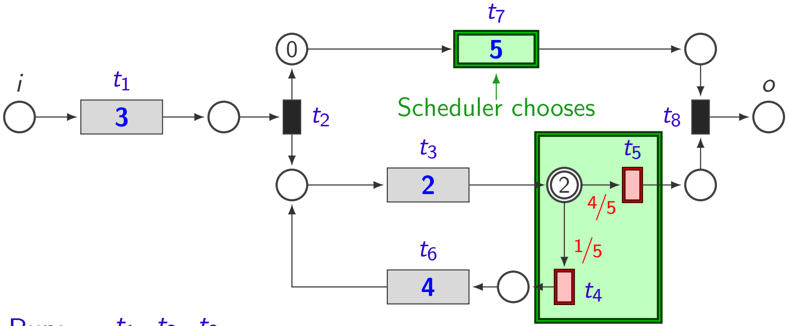


Run: t_1 t_2

Time: $\mathbf{0}+\mathbf{3}$

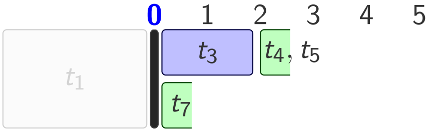


Example: Run of “earliest-first” scheduler

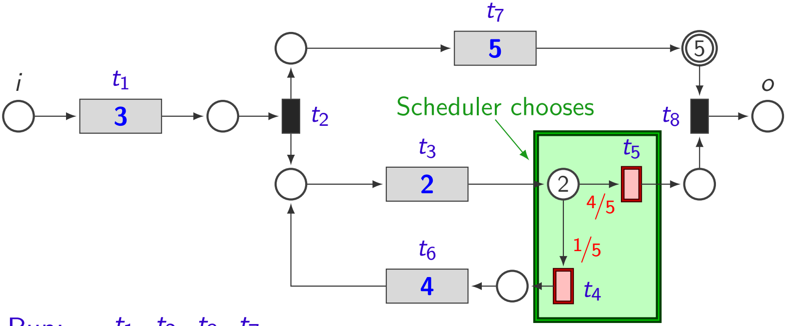


Run: t_1 t_2 t_3

Time: $\mathbf{0}+\mathbf{3}+\mathbf{0}$

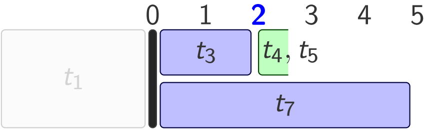


Example: Run of "earliest-first" scheduler

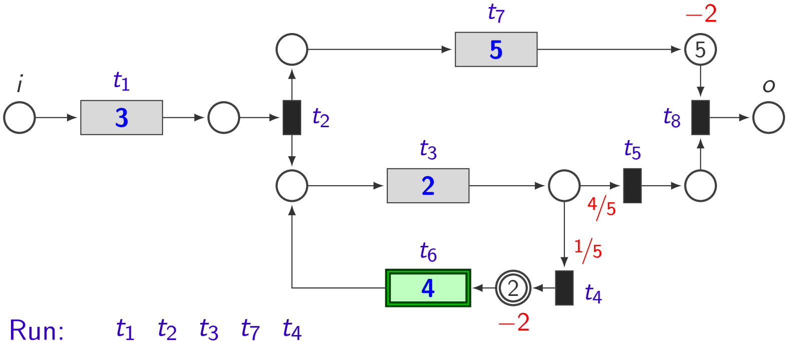


Run: t_1 t_2 t_3 t_7

Time: $\mathbf{0}+\mathbf{3}+\mathbf{0}+\mathbf{0}$

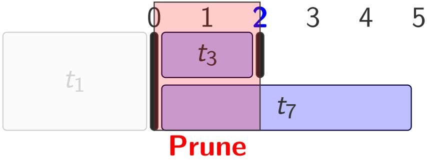


Example: Run of "earliest-first" scheduler

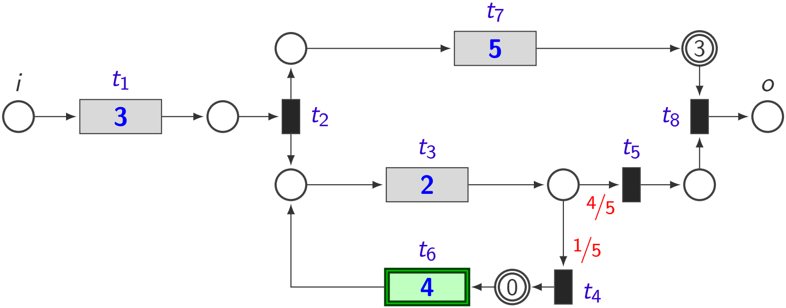


Run: t_1 t_2 t_3 t_7 t_4

Time: $\mathbf{0}+\mathbf{3}+\mathbf{0}+\mathbf{0}+\mathbf{2}$

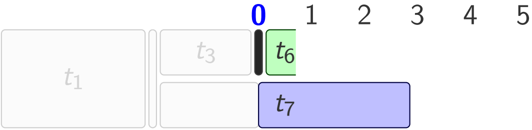


Example: Run of “earliest-first” scheduler

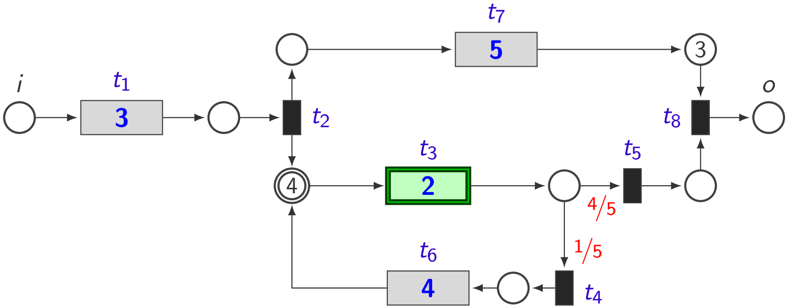


Run: t_1 t_2 t_3 t_7 t_4

Time: $\mathbf{0}+\mathbf{3}+\mathbf{0}+\mathbf{0}+\mathbf{2}$

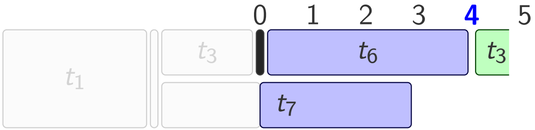


Example: Run of "earliest-first" scheduler

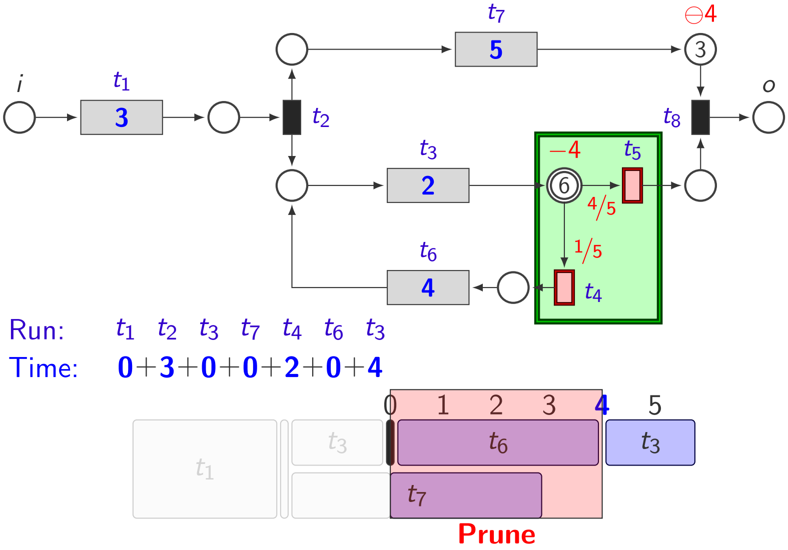


Run: t_1 t_2 t_3 t_7 t_4 t_6

Time: $\mathbf{0}+\mathbf{3}+\mathbf{0}+\mathbf{0}+\mathbf{2}+\mathbf{0}$



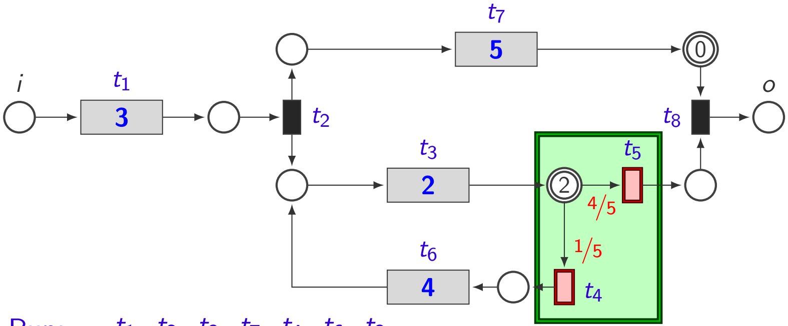
Example: Run of “earliest-first” scheduler



Run: t_1 t_2 t_3 t_7 t_4 t_6 t_3

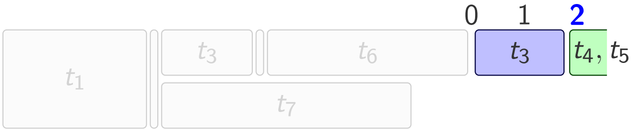
Time: **0**+**3**+**0**+**0**+**2**+**0**+**4**

Example: Run of “earliest-first” scheduler

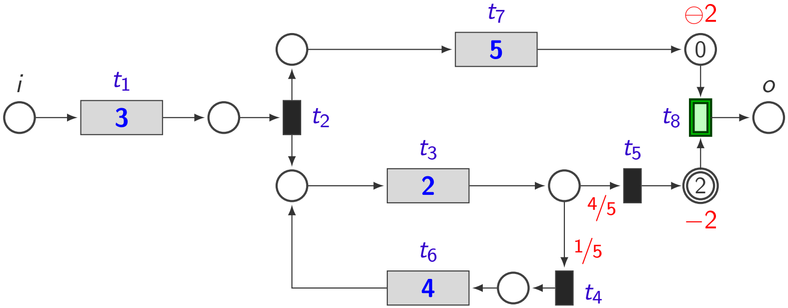


Run: t_1 t_2 t_3 t_7 t_4 t_6 t_3

Time: $\mathbf{0+3+0+0+2+0+4}$



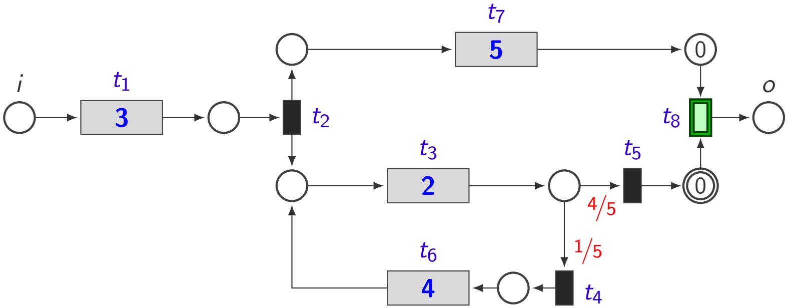
Example: Run of "earliest-first" scheduler



Run: t_1 t_2 t_3 t_7 t_4 t_6 t_3 t_5

Time: $\mathbf{0}+\mathbf{3}+\mathbf{0}+\mathbf{0}+\mathbf{2}+\mathbf{0}+\mathbf{4}+\mathbf{2}$

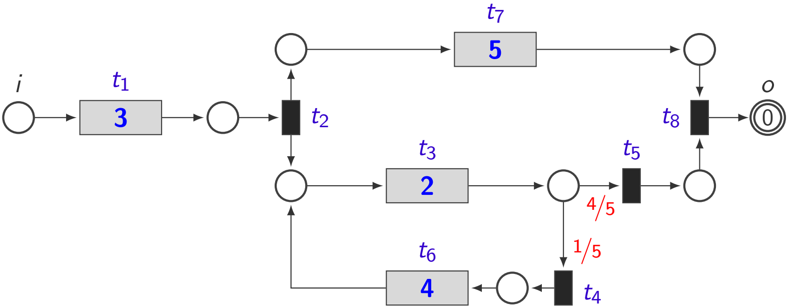
Example: Run of "earliest-first" scheduler



Run: t_1 t_2 t_3 t_7 t_4 t_6 t_3 t_5

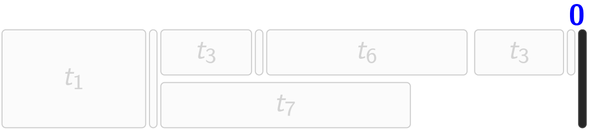
Time: $\mathbf{0+3+0+0+2+0+4+2}$

Example: Run of "earliest-first" scheduler

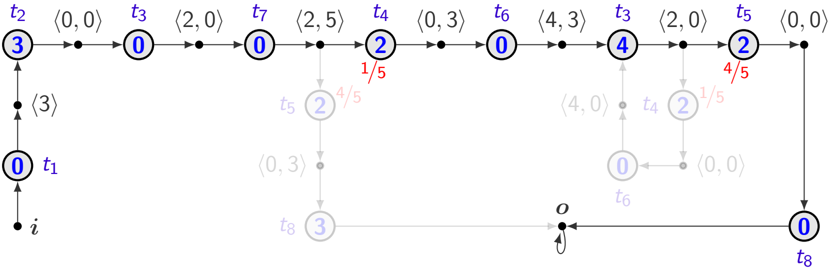


Run: t_1 t_2 t_3 t_7 t_4 t_6 t_3 t_5 t_8

Time: $\mathbf{0}+\mathbf{3}+\mathbf{0}+\mathbf{0}+\mathbf{2}+\mathbf{0}+\mathbf{4}+\mathbf{2}+\mathbf{0}=\mathbf{11}$

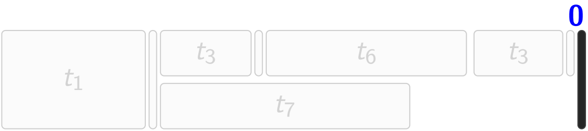


Example: Run of "earliest-first" scheduler

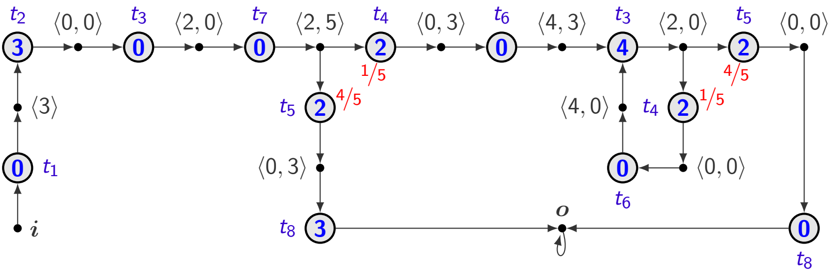


Run: t_1 t_2 t_3 t_7 t_4 t_6 t_3 t_5 t_8

Time: $\mathbf{0}+\mathbf{3}+\mathbf{0}+\mathbf{0}+\mathbf{2}+\mathbf{0}+\mathbf{4}+\mathbf{2}+\mathbf{0}=\mathbf{11}$



Example: Run of “earliest-first” scheduler



$$\text{ExpectedTime} = \text{ExpectedReward}(\boldsymbol{i} \to \boldsymbol{o}) = 8.9$$

Lower bound for complexity of computing the expected time

Theorem

Computing the expected time of a sound and acyclic free-choice TPWN where all times are 0 or 1 and all probabilities are 1 or $1/2$ is #P-hard.

Proof.

Reduction from expected duration of stochastic PERT network. $\square$

- #P-hard: allows reduction from #SAT, i.e. counting the number of satisfying assignments for a boolean formula.
- Computing an ϵ-approximation is also #P-hard.
- Computing the probability that that the expected time exceeds a given number is also #P-hard.

Comparison of complexities

Complexity of different problems for 1-safe workflow nets.

	Net type	
Problem	Arbitrary	Free-choice
Soundness	PSPACE-complete[2]	P[1]
Confusion-free if sound	PSPACE-complete[3]	$\mathcal{O}(1)$ (yes)

	Choice	
Problem	Confusion-free	Free-choice
Expected Cost	PSPACE-hard[3]	P[3]
Expected Time	**EXPTIME**	**#P-hard**

[1] van der Aalst '96 [2] Liu et al. '14 [3] Esparza et al. '17

Experimental evalation

- Implemented as package in **ProM** (Process Mining framework).
- Evaluated on 642 sound and free-choice workflow nets from IBM.

Net	Cyclic	Places	Transitions	Reach. Markings	Analysis time	Size of MC
m1.s30_s703	no	264	286	6117	43.8 ms	347
m1.s30_s596	yes	214	230	623	23.6 ms	234
b3.s371_s1986	no	235	101	$2 \cdot 10^{17}$	16.5 ms	102
b2.s275_s2417	no	103	68	237626	15.9 ms	431

- Evaluation on net from BPI Challenge 2017 for financial process.

Discretization of task times		Transitions	Exp. Time		Size of MC	Analysis Time
Individual deterministic mean		19	24 d	1 h	33	40 ms
	12 h	141	24 d	18 h	4054	244 ms
	6 h	261	24 d	21 h	15522	2.1 s
Histogram discretization	4 h	375	24 d	22 h	34063	10 s
	2 h	666	24 d	23 h	122785	346 s
	1 h	1117		—	422614	memout

Summary

- Semantics for expected time of confusion-free workflow nets.
- Algorithm to compute expected time of a workflow net.
- #P-hardness lower bound even for restricted net class.
- Efficient computation on large set of industrial examples.

Thank you!