Robust Execution of Probabilistic STNs

Luke Hunsberger[1] **Roberto Posenato**[2]

[1]Department of Computer Science, Vassar College, Poughkeepsie, NY USA

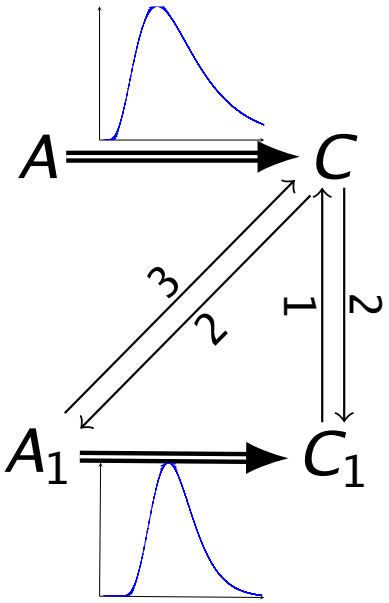
[2]Department of Computer Science, University of Verona, Verona, Italy

TIME 2024
October 28–30, 2024
Montpellier, France

Probabilistic Simple Temporal Network (PSTN)

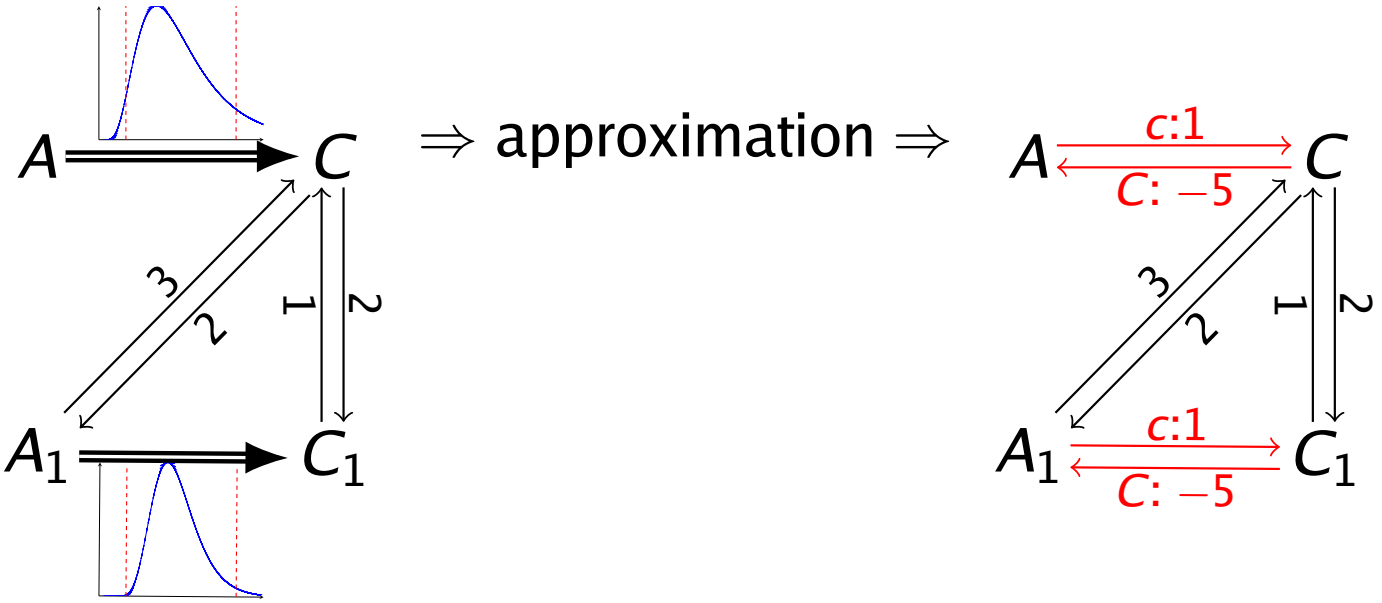
- Probabilistic Simple Temporal Network (PSTN) [8]:

  a data structure for representing and reasoning about events and actions with uncertain durations, modeled using continuous probability density functions.



Executing PSTN

- In the literature, there are approaches to manage the uncertainty during execution by approximating a PSTN by an STNU*, and then using an STNU execution strategy to execute the PSTN.



*Simple Temporal Network with Uncertainty [6]

Executing PSTN

Previous Approximation Approaches

- MIT researchers proposed to approximate PSTNs by STNUs "*choosing bounds for the approximating STNU's contingent links that capture as much probability mass of the probabilistic durations as possible while preserving the strong STNU's controllability (SC)* [1, 11, 9]
- In this way, it is possible to execute the PSTN with a SC STNU scheduler (that is an off–line scheduler!)

Two limitations:

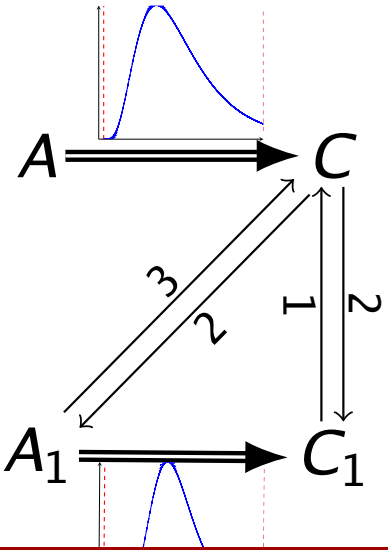
- *pdf*s can have infinite tails
  $\Rightarrow$ successfully executing a PSTN cannot be guaranteed.
- Strong Controllability is very restrictive!

*If (A, x, y, C) is a contingent link approximating a probabilistic duration (A, C, p), then the probability mass *captured* by the contingent link is $\int_x^y p(t)dt = F(y) - F(x)$, where F is the associated *cumulative distribution function (cdf)* for p.

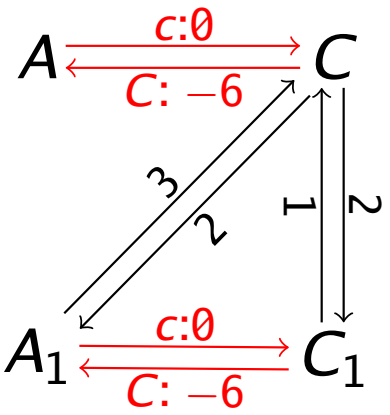
Approximating a PSTN by a DC STNU

More recent approach

- Wang [10] proposed an iterative method to approximate a PSTN by a dynamically controllable (DC) STNU:
  (1) Use a non–linear optimization (NLO) solver to generate fixed bounds for the STNU's contingent links that capture as much of the probability mass of the PSTN's durations as possible while also satisfying the ordinary constraints of the STNU.
  (2) Use Morris' $O(n^4)$–time DC–checking algorithm, modified to find a semi–reducible negative (SRN) cycle for non–DC networks.
  (3) If an SRN cycle found, resolve it by inserting additional constraints, then go back to Step 1.



⇒ approximation ⇒



NOT DC!

Approximating a PSTN by a DC STNU

More recent approach

- To solve an SRN cycle, Wang's approach requires inserting potentially very many disjunctive constraints;
- We proved that this can lead to sequences of nearly identical iterations of the optimize/DC–check/resolve process.
- Moreover, the methods does not consider the possibility of magic loops (i.e., cycles of exponential number of repeated contingent links constraints).
- Interesting approach, but the lack of details precludes a proper evaluation.

Approximating a PSTN by a DC STNU

Our Approach for generating the approximating STNU

Although we follow the same general approach of approximating a PSTN by a STNU, we propose the following significant improvements:

- use the FindSRNC algorithm [3] to detect when the approximating STNU is not DC (instead of Morris14!);
- solve negative cycles maximizing the combined probability mass of the probabilistic durations captured by the STNU's contingent links;
- regarding resolving negative cycles in non–DC STNUs, we formally proved that many of the constraints used by others when generating the STNU approximation are unnecessary;
- use the recent and fast $\text{minDisp}_{\text{ESTNU}}$ algorithm [4] to compute an equivalent dispatchable ESTNU[†] having a minimal number of edges;

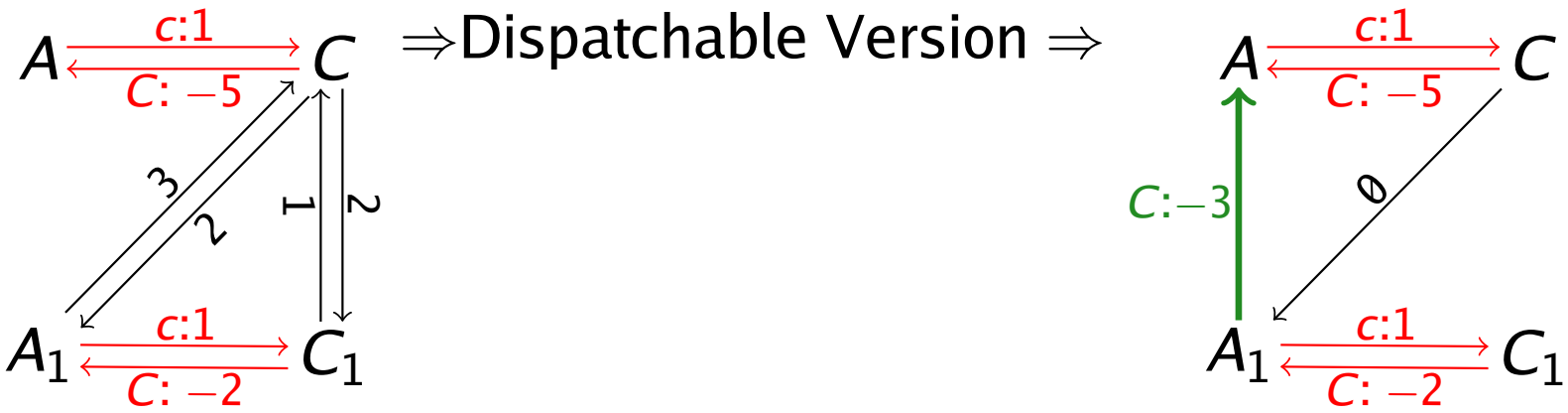
[†]ESTNU = Extended STNU (includes wait constraints)

Executing a PSTN by a DC STNU

- Use the flexible and efficient RTE* algorithm [5], proposed for STNUs, to execute the PSTN in real time (considering the dispatchable ESTNU).
  - RTE* allows the use of different execution strategies like midpoint strategy, earliest–first strategy, etc.
  - *The strategy's flexibility enables it to react to observations of probabilistic durations, even those that fall outside the fixed bounds of the corresponding contingent links.*
  - We think the midpoint strategy improves execution by adjusting to unexpected durations outside the STNU's fixed bounds.

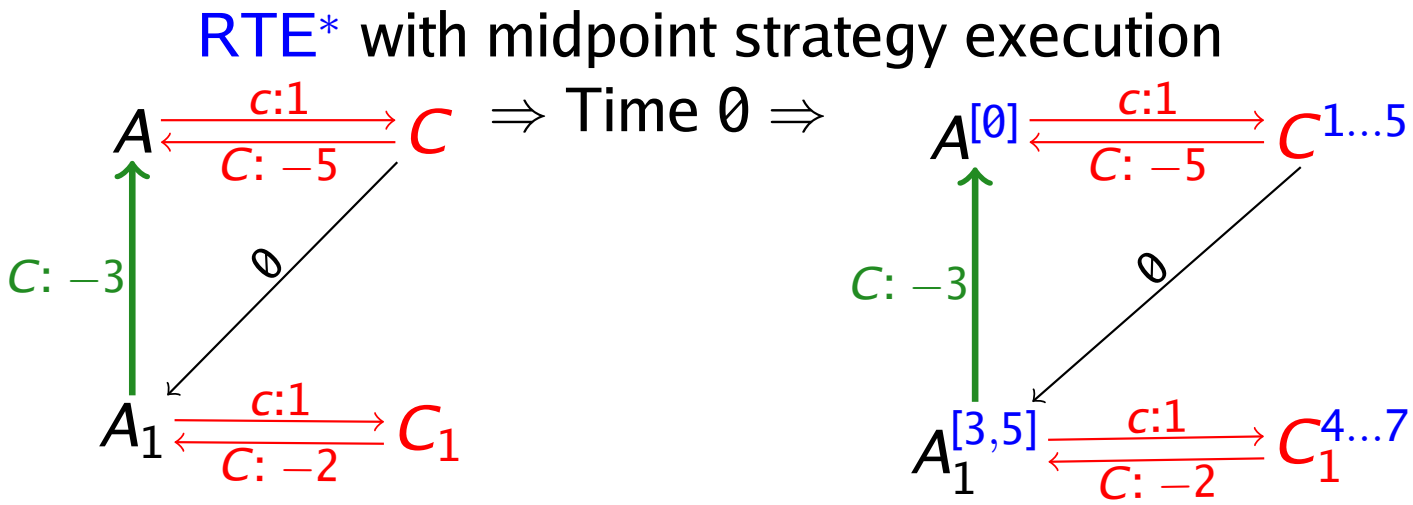
Executing a PSTN by a DC STNU

Example



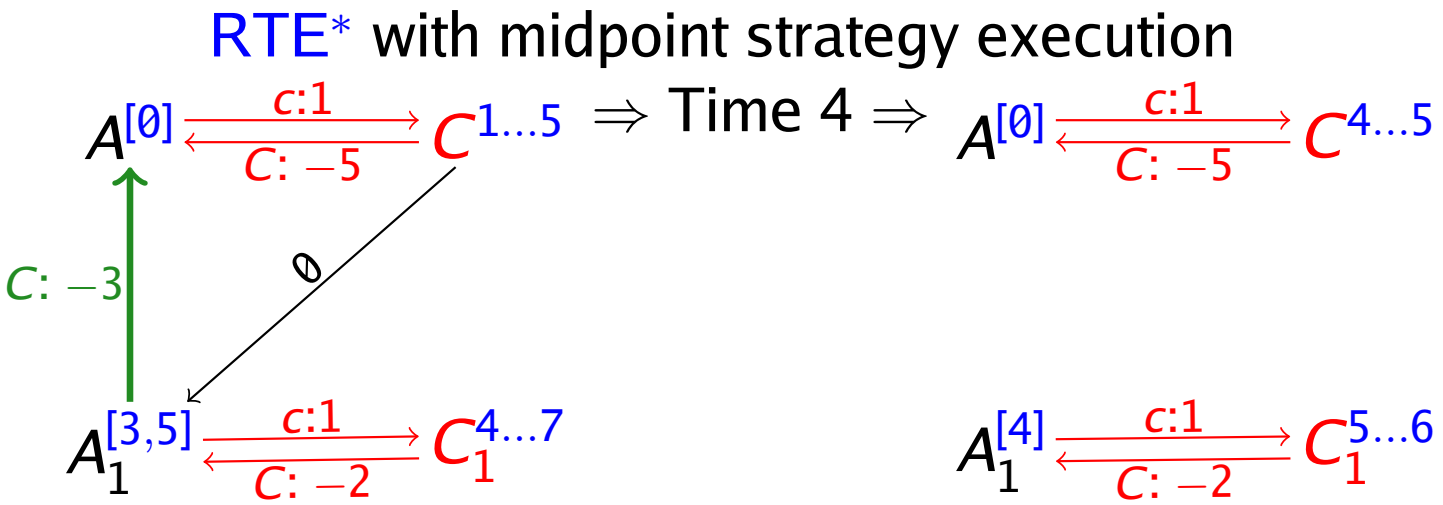
Executing a PSTN by a DC STNU

Example



Executing a PSTN by a DC STNU

Example



Executing a PSTN by a DC STNU

Example

RTE* with midpoint strategy execution

C is still not occurring,

while C_1 occurred correctly!

$$A^{[0]} \underset{C:-5}{\overset{c:1}{\rightleftarrows}} C^{4\ldots5} \Rightarrow \text{Time 6} \Rightarrow A^{[0]} \underset{C:-5}{\overset{c:1}{\rightleftarrows}} C^{\ldots}$$

$$A_1^{[4]} \underset{C:-2}{\overset{c:1}{\rightleftarrows}} C_1^{5\ldots6}$$

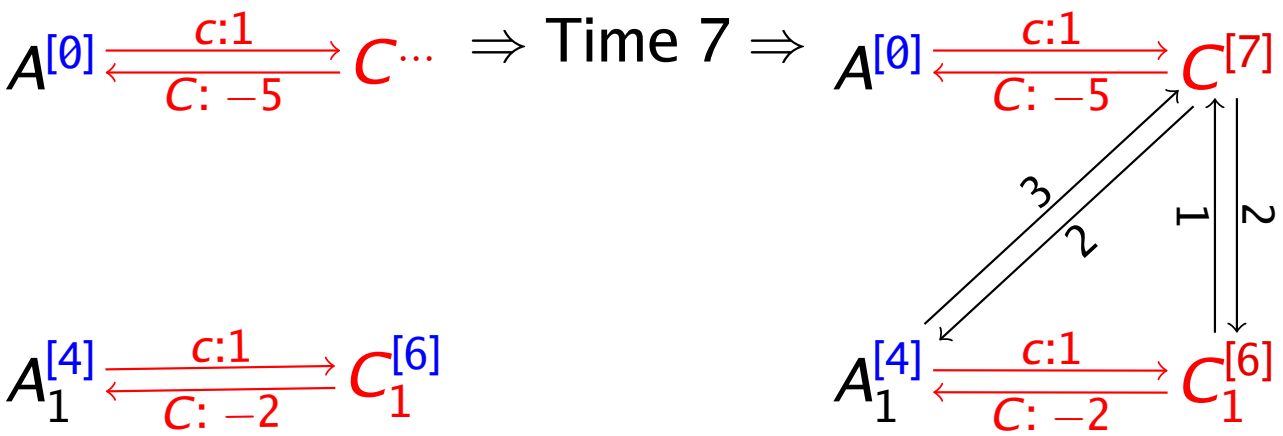
$$A_1^{[4]} \underset{C:-2}{\overset{c:1}{\rightleftarrows}} C_1^{[6]}$$

Executing a PSTN by a DC STNU

Example

RTE* with midpoint strategy execution

C is occurred outside its allowed range...,
but constraint are still satisfied!



Simple Temporal Networks with Uncertainty (STNUs)

Recent Relevant Algorithms for STNUs

1. FindSRNC solves the STNU DC–checking problem for STNUs, returning a *semi–reducible negative* (SRN) cycle when the input is not DC, in time $O(mn + k^2 n + kn \log n)$ [3].

2. FD_{STNU} solves the STNU DC–checking problem, returning an equivalent dispatchable ESTNU when the input is DC in time $O(mn + k^2 n + kn \log n)$. [2].

3. $\text{minDisp}_{\text{ESTNU}}$ returns a minimal dispatchable version of an ESTNU in time $O(n^3)$ [4]

4. RTE* executes an ESTNU in time $O(m + kn \log(kn))$ [5].

† n = #timepoints; m = #constraints; k = #contingent links.

Proof of Concept

Overview

- We implemented our algorithm inside the CSTNU Tool [7].
- Then, we experimentally evaluated the robust execution of PSTNs by:
  - generating random PSTN instances (we transformed random non–DC STNUs into PSTNs by assigning log–normal distributions to the contingent links);
  - determining the approximating STNU using `genApproxSTNU`;
  - executing PSTNs using the RTE* algorithm based on the approximating STNU, converted to a dispatchable STNU — durations for the probabilistic links were determined according to their distributions.

The percentage of successful executions across random trials provides a measure of the PSTN's robustness.

Proof of Concept

Results of generating DC approximating STNUs for PSTNs

Results using `genApproxSTNU` to generate DC approximating STNUs for PSTNs

#PSTNs	n	k	$\overline{m}$	exTime [s]	optTime [s]	#NLOprobs	%probMass
24	500	50	1558	0.191	0.141	0.96	77
24	1000	100	3136	0.223	0.042	1.00	67
14	1500	150	4713	0.573	0.100	1.21	43
17	2000	200	6289	0.914	0.046	1.11	53

- "exTime" is the average time to execute `genApproxSTNU`;
- "optTime" is the average time spent to execute `nlpOpt`;
- "#NLOprobs" is the average number of calls to `nlpOpt`;
- "%probMass" is the average probability mass of the probabilistic links captured by the approximating STNU;

As expected, the % of the probability mass captured by the approximating STNU fell as the number of contingent durations increased since the cumulative probability mass is a product of the probability masses.

Proof of Concept

Result of PSTNs execution

- We ran the RTE* algorithm 200 times on each dispatchable STNU, where the contingent durations were obtained by randomly sampling the associated log–normal distributions (15800 executions in total).
- To test the impact of the execution strategy on the rate of successful execution, we executed each network in the same situation (i.e., durations) twice: once with the earliest–first strategy and once with the midpoint strategy

Proof of Concept

Results RTE* execution on PSTNs

Earliest–First (EF) vs. *Midpoint (MP)*

#PSTNs	n	k	$\overline{m}$	execTP (μs)	%trials–in succ		%trials–out succ		%trials–out fail	
					EF	*MP*	EF	*MP*	EF	*MP*
24	500	50	2500	9.16	73	*73*	5	*5*	22	*22*
24	1000	100	5119	14.98	66	*66*	8	*6*	26	*28*
14	1500	150	7883	26.14	58	*58*	4	*7*	38	*35*
17	2000	200	106522	31.05	53	*53*	8	*8*	39	*39*

"%trials–in succ" = % trials all sampled durations are within the respective ctg. bounds; "%trials–out succ" = % of trials where ≥ 1 ctg. durations are outside the bounds, but the execution succeeded anyway!

- "%probMass" in Table 1 and "%trials–in succ" here confirm the prob. mass captured by STNU's ctg. links is = situations with successful exec.
- Not clear if EF or MP execution strategy can increase the rate of successful executions (the # different PSTNs is limited).
- The results provide evidence that the RTE* algorithm makes it possible to have successful executions even when one or more contingent durations are outside the STNU's bounds.

Conclusion

We propose:

- a new approach to the robust execution of PSTNs that takes advantage of several recent efficient algorithms;
- a new algorithm to generate an approximating STNU that aims to maximize the combined probability mass of the PSTN's probabilistic durations while maintaining the dynamic controllability of the STNU;
- a formal analysis of SRN cycles that provided new insights into how to efficiently resolve them while avoiding issues arising in past approaches;
- an empirical evaluation of our approach provides evidence of its effectiveness on robustly executing PSTNs derived from a publicly available benchmark.

References I

[1] Cheng Fang, Peng Yu, and Brian C. Williams.
Chance–constrained probabilistic simple temporal problems.
In *28th AAAI Conference on Artificial Intelligence (AAAI–2014)*, volume 3, pages 2264–2270, 2014.
doi:10.1609/aaai.v28i1.9048.

[2] Luke Hunsberger and Roberto Posenato.
A Faster Algorithm for Converting Simple Temporal Networks with Uncertainty into Dispatchable Form.
Information and Computation, 293(105063):1–21, 2023.
doi:10.1016/j.ic.2023.105063.

[3] Luke Hunsberger and Roberto Posenato.
A Faster Algorithm for Finding Negative Cycles in Simple Temporal Networks with Uncertainty.
In *31st International Symposium on Temporal Representation and Reasoning (TIME 2024)*, volume 318 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:15, 2024.

References II

doi:10.4230/LIPIcs.TIME.2024.9.

[4] Luke Hunsberger and Roberto Posenato.
Faster Algorithm for Converting an STNU into Minimal Dispatchable Form.
In *31st International Symposium on Temporal Representation and Reasoning (TIME 2024)*, volume 318 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:14, 2024.
doi:10.4230/LIPIcs.TIME.2024.11.

[5] Luke Hunsberger and Roberto Posenato.
Foundations of Dispatchability for Simple Temporal Networks with Uncertainty.
In *16th International Conference on Agents and Artificial Intelligence (ICAART 2024)*, volume 2, pages 253–263. SCITEPRESS, 2024.
doi:10.5220/0012360000003636.

References III

[6] Paul Morris, Nicola Muscettola, and Thierry Vidal.
Dynamic control of plans with temporal uncertainty.
In *17th Int. Joint Conf. on Artificial Intelligence (IJCAI–2001)*, volume 1, pages 494–499, 2001.
URL: https://www.ijcai.org/Proceedings/01/IJCAI-2001-e.pdf.

[7] Roberto Posenato.
CSTNU Tool: A Java library for checking temporal networks.
SoftwareX, 17:100905, 2022.
doi:10.1016/j.softx.2021.100905.

[8] Ioannis Tsamardinos.
A probabilistic approach to robust execution of temporal plans with uncertainty.
In *Methods and Applications of Artificial Intelligence (SETN 2002)*, volume 2308 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 97–108, 2002.
doi:10.1007/3-540-46014-4_10.

References IV

[9] Andrew Wang and Brian C. Williams.
Chance–Constrained Scheduling via Conflict–Directed Risk Allocation.
In *29th Conference on Artificial Intelligence (AAAI–2015)*, volume 29, 2015.
doi:10.1609/aaai.v29i1.9693.

[10] Andrew J. Wang.
Risk–bounded Dynamic Scheduling of Temporal Plans.
PhD thesis, Massachusetts Institute of Technology, 2022.
URL: https://hdl.handle.net/1721.1/147542.

[11] Peng Yu, Cheng Fang, and Brian Charles Williams.
Resolving uncontrollable conditional temporal problems using continuous relaxations.
In *24th International Conference on Automated Planning and Scheduling, ICAPS 2014*. AAAI, 2014.
doi:10.1609/icaps.v24i1.13623.

Probabilistic Simple Temporal Network (PSTN)

- Differently from many previous works, we assume that such *pdf*s are log–normal distributions because they better represent the concept of *uncertain action duration*.