Mining Temporal Networks: Results and Open Problems

Guido Sciavicco[1], Tiziano Villa[2] and Matteo Zavatteri[2]

[1]Department of Mathematics and Computer Science, University of Ferrara, Italy
[2]Department of Computer Science, University of Verona, Italy

Abstract
The design of temporal networks typically follows a top-down approach where a designer handcrafts a temporal network to model some concrete plan of interest. Instead, the bottom-up approach of *mining* is the process of building a temporal network from a set of execution traces of some (typically unknown) underlying process. Recent research showed that, due to the structural properties of temporal networks, such a task can be done in polynomial time. In this paper, we give an overview of the current status of our research and highlight open problems concerning Formal Methods and Artificial Intelligence.

Keywords
Mining temporal networks, cstnud, uncertainty, formal methods for AI

1. Introduction

Temporal networks are a possible framework to model temporal plans and check the consistency of their temporal constraints imposing delays and deadlines between the occurrences of pairs of events in the plan [1]. Over the years the core formalism of *Simple Temporal Networks* [1] has been extended in several ways to cope with uncontrollable durations [2], uncontrollable and controllable choices [3, 4] and, more recently, with combinations of them (see, e.g., [5, 6, 7, 8, 9, 10]). The most expressive formalisms of temporal networks are those that simultaneously handle all such features. *Conditional Simple Temporal Networks with Uncertainty and Decisions* (*CSTNUDs*, [9, 10]) is a recent formalism that is able to model controllable and uncontrollable durations as well as controllable and uncontrollable choices simultaneously.

Like any model-based engineering approach, creating a temporal network is a complex, time-consuming, and error-prone task, where discrepancies between the actual process and the obtained network might eventually emerge requiring that the designer refines or abstracts the model being created. This is a top-down, trial-and-error approach. Instead, the opposite, bottom-up approach is known in the literature under the name of *process mining* and it aims to *mine* (i.e., synthesize) process descriptions (or, more reasonably, model approximations) from execution traces (i.e., process logs). A *trace* describes a run of a process, whereas the set of all available traces can be thought of as a *log* of a process carried out many times. One of the first

OVERLAY 2021: 3rd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, September 22, 2021, Padova, Italy
guido.sciavicco@unife.it (G. Sciavicco); tiziano.villa@univr.it (T. Villa); matteo.zavatteri@univr.it (M. Zavatteri)

Table 1
Building blocks to design CSTNUDs.

Time points	Contingent links	Constraints
$A?$ $\quad B!$ $\quad C$	$A \xRightarrow{[2,8]} C$	$A \underset{\{\}:-1}{\overset{\{a,\neg b\}:10}{\rightleftarrows}} B$

contributions in process mining is that of Agrawal, Gunopulos, and Leymann [11], but, after this seminal work, many others followed by focusing on different process description languages [12, 13, 14, 15, 16, 17, 18]. The problem of mining temporal networks subject to uncontrollable parts received particular attention in [19] and [20]. In this paper, we briefly summarize our current results and discuss future directions.

2. Mining CSTNUDs

A CSTNUD involves a finite set of time-points $\mathcal{T}$, a finite set of booleans $\mathcal{B}$, a finite set of contingent links $\mathcal{L}$, and a finite set of constraints $\mathcal{C}$ [9, 10]. A classic convention identifies time points by upper-case letters ($A, B, \dots$) and booleans by lower-case ones ($a, b, \dots$). Both time points and booleans are partitioned in controllable and uncontrollable ones (i.e., $\mathcal{T} = \mathcal{T}_C \cup \mathcal{T}_U$ and $\mathcal{B} = \mathcal{B}_C \cup \mathcal{B}_U$), where controllable means that their value assignments can be *decided* by the executing agent, whereas uncontrollable means that their value assignments can only be *observed* once they take place. Time points are executed as soon as they are assigned real values. There exists a bijection $\beta\colon \mathcal{B} \rightleftharpoons \mathcal{T}_C$ between the set of (all) booleans and a subset of controllable time points to associate booleans to time points. As soon as $\beta(b)$ is executed, the truth value of b is assigned by the executing agent (if $b \in \mathcal{B}_C$), or observed (if $b \in \mathcal{B}_U$). Each contingent link models an uncontrollable duration and has the form (A, ℓ, u, C), where $A \in \mathcal{T}_C$, $C \in \mathcal{T}_U$, and $\ell, u \in \mathbb{R}$ with $0 < \ell \leq u$ [19, 20]. Once A is executed, then C will be observed to occur at a time such that $C - A \in [\ell, u]$. Finally, each constraint has the form $\mathcal{S}\colon\ A - B \leq k$ where $\mathcal{S}$ is a consistent set of literals over $\mathcal{B}$, $A, B \in \mathbb{R}$ and $k \in \mathbb{R}$. When all time points are executed and all booleans are assigned, if the truth assignment to the booleans satisfies $\mathcal{S}$, then $B - A \leq k$ holds. Whenever $\mathcal{S} = \emptyset$ the constraint is unconditional (i.e., it must always hold).

Every CSTNUD admits a graph-based representation in which nodes model time points, whereas directed labeled edges model contingent links (double) and constraints (single). Whenever a node is suffixed by ? or ! it means that its has an uncontrollable or controllable boolean associated respectively (for simplicity identified by the same letter in lower-case). Table 1 shows such core components. $A?$, $B!$, and C are time points, with A associated to the uncontrollable boolean a and B associated to the controllable boolean b. C is an uncontrollable time point. The contingent link $A \Rightarrow C$ models an uncontrollable duration between 2 and 8 time units. Finally, the arc $B \to A$ models the unconditional delay constraint $\{\}\colon\ A - B \leq -1$ meaning that B is always at least 1 after A, whereas $A \to B$ models the conditional deadline constraint $\{a, \neg b\}\colon\ B - A \leq 10$ meaning that if a is observed to be true and b is decided to be false, then B is within 10 after A.

Table 2
Meaning of the weakening rules: adding (first row), weakening (second and third row).

	Contingent Links			**Constraints**	
Current	**New**	**Merged**	**Current**	**New**	**Merged**
A	$A \xRightarrow{[1,1]} C$	$A \xRightarrow{[1,1]} C$	$A \xrightarrow{\{a,\neg b\}:1} B$	$A \xrightarrow{\{a,b\}:8} B$	$A \xrightarrow{\substack{\{a,b\}:8\\ \{a,\neg b\}:1}} B$
$A \xRightarrow{[2,8]} C$	$A \xRightarrow{[10,10]} C$	$A \xRightarrow{[2,10]} C$	$A \xrightarrow{\substack{\{a,\neg b\}:5\\ \{a,c\}:3}} B$	$A \xrightarrow{\{a\}:8} B$	$A \xrightarrow{\{a\}:8} B$
$A \xRightarrow{[3,9]} C$	$A \xRightarrow{[1,1]} C$	$A \xRightarrow{[1,9]} C$	$A \xrightarrow{\substack{\{\neg b\}:1\\ \{a\}:5}} B$	$A \xrightarrow{\{a,\neg b\}:8} B$	$A \xrightarrow{\substack{\{\neg b\}:8\\ \{a\}:8}} B$

We initially proposed an algorithm to mine CSTNUDs in [19] whose extended version appears in [20]. The algorithm processes a set of well-defined and coherent traces and generates a CSTNUD that contains all time points and booleans appearing in the traces and such that all temporal and truth value assignments of each trace satisfy in the CSTNUD all constraints restricted to those subsets of time points and booleans. Each trace is a set of events that happened at their corresponding absolute times plus the truth value assignment to the booleans (if any) (e.g., $Z = 0, A = 1, B = 2, \neg b, \dots$, see [19, 20]). Therefore, our mined networks have always a time point Z (zero-time point) from and to which all constraints are generated. The idea of the algorithm is fairly simple and can be summarized intuitively in these two main points:

1. Add time points, booleans, contingent links, and constraints whenever they are not in the CSTNUD being mined.
2. Weaken specific contingent links and constraints whenever they are already in the CSTNUD being mined.

Weakening means to *make contingent links and constraints more general* so that they can model executions coming from the processed traces. For example, if the current CSTNUD contains a contingent link (A, ℓ, u, C) and in the next trace the duration of such link is fixed to be k, then we modify the CSTNUD so that the "updated" contingent link is $(A, \min(\ell, k), \max(u, k), C)$. Likewise, consider a constraint $A \to B$ labeled by $\mathcal{S}\colon k$ in the CSTNUD, where either A or B is Z. Assume that the next trace entails a constraint $A \to B$ labeled by $\mathcal{S}'\colon k'$. If $\mathcal{S} \subseteq \mathcal{S}'$, then we replace the label $\mathcal{S}\colon k$ of the original constraint with $\mathcal{S}\colon \max(k, k')$ (i.e., we operate on the numeric value only). Instead, if $\mathcal{S}' \subset \mathcal{S}$, we replace the label $\mathcal{S}\colon k$ of the original constraint with $\mathcal{S}'\colon \max(k, k')$ (i.e., we operate on both the set of literals and numeric value). Of course, if more constraints satisfy the applicability condition of the rule they will be modified all together. Table 2 provides some intuitive examples.

The last step of our algorithm applies a final condition that further relaxes numeric values on constraints to impose a coherence on those constraints whose union of preconditions is a consistent set of literals. We need this last step because we process traces that might be partial. We do not discuss it here for simplicity. Overall, the take-home message is simple:

Always keep the most general constraint.

Since we deal with temporal constraints that are interval bounds, we can work with min/max numeric values. Likewise, since the preconditions on the applications of constraints are sets, we can work with sub/supersets. The asymptotic complexity of our algorithm is upper bounded by $\mathcal{O}(|\mathcal{I}|^2 \times (K + |\mathcal{T}|))$ where $\mathcal{I}$ is the set of input traces and K is the length of the longest trace in it. We also developed `cstnud-miner` an open-source software prototype that is available at https://github.com/matteozavatteri/cstnud-miner.

3. Mining of Temporal Networks: Open Problems

We exploited the well-defined monotone mathematical structure of temporal networks, and we succeeded in the design of a correct algorithm to mine CSTNUDs with a property of interest that we called *significance*. Roughly, a CSTNUD is significant whenever it contains all time points and booleans appearing in the traces and the temporal and truth value assignments arising from each processed trace satisfy the constraints restricted to these variables in the mined CSTNUD. This is what our theory achieved so far. We outline here some future directions.

1. There exist three main kinds of controllability of temporal networks subject to uncontrollable parts: weak, strong, and dynamic. Weak controllability is when all uncontrollable temporal and truth value assignments are known before starting the execution. Strong controllability is the dual case in which no uncontrollable part will be known during execution. Dynamic controllability is when we make our execution decisions in real-time depending on which uncontrollable temporal and truth value assignments we are observing. All three kinds of controllability boil down to consistency whenever there is no uncontrollable part. In [20] we carried out an experimental evaluation by generating sets of traces belonging to all subformalisms of CSTNUDs. Afterwards, we tested for consistency (if no uncontrollable parts were present), weak, strong and dynamic controllability otherwise. We found that our algorithm mines CSTNUDs which are not strongly controllable (we provided a counter example). Regarding consistency, weak, and dynamic controllability (depending on the type of network) we could not find a counter example. This makes us believe that our algorithm mines CSTNUDs which are weakly and also dynamically controllable (and thus mines consistent networks when all time points and booleans are controllable). However, a formal proof of this statement is still missing and requires to prove the existence of an execution strategy for the mined network.
2. If the aim of (1) would lead to a negative result, and worse still, to a proof that mining controllable CSTNUDs (whose set of constraints is not, of course, empty) is a hard problem, then an attempt toward Machine Learning techniques would be justified to obtain an algorithm to mine (hopefully) controllable temporal networks. Of course, since such AI approaches may fail, particular attention should be devoted to its “success rate”.
3. Assuming to have (2), it remains to be understood if it might (in practice) be more competitive than our current algorithm on huge sets of input traces.
4. Last but not least, several extensions of the algorithm are needed to mine networks such as, for example, those in [21, 22, 23] to handle both time and resource constraints together.

In conclusion these open problems call for a fruitful synergy between Formal Methods and Machine Learning techniques from Artificial Intelligence.

Acknowledgments

This work was partially supported by MIUR, Project *Italian Outstanding Departments, 2018-2022* and by INdAM, GNCS 2020, Project *Strategic Reasoning and Automated Synthesis of Multi-Agent Systems.*

References

[1] R. Dechter, I. Meiri, J. Pearl, Temporal constraint networks, Artif. Intell. 49 (1991) 61–95.

[2] T. Vidal, H. Fargier, Handling contingency in temporal constraint networks: from consistency to controllabilities, Jour. of Exp. & Theor. Artif. Intell. 11 (1999) 23–45.

[3] P. R. Conrad, B. C. Williams, Drake: An efficient executive for temporal plans with choice, JAIR 42 (2011) 607–659.

[4] I. Tsamardinos, T. Vidal, M. E. Pollack, CTP: A new constraint-based formalism for conditional, temporal planning, Constraints 8 (2003) 365–388.

[5] A. Cimatti, L. Hunsberger, A. Micheli, R. Posenato, M. Roveri, Dynamic controllability via timed game automata, Acta Informatica 53 (2016) 681–722.

[6] E. Karpas, S. J. Levine, P. Yu, B. C. Williams, Robust execution of plans for human-robot teams, in: ICAPS '15, AAAI Press, 2015, pp. 342–346.

[7] S. J. Levine, B. C. Williams, Concurrent plan recognition and execution for human-robot teams, in: ICAPS '14, AAAI, 2014, pp. 490–498.

[8] P. Yu, C. Fang, B. C. Williams, Resolving uncontrollable conditional temporal problems using continuous relaxations, in: ICAPS '14, AAAI, 2014, pp. 341–349.

[9] M. Zavatteri, Conditional simple temporal networks with uncertainty and decisions, in: TIME 2017, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017, pp. 23:1–23:17.

[10] M. Zavatteri, L. Viganò, Conditional simple temporal networks with uncertainty and decisions, Theoretical Computer Science 797 (2019) 77–101.

[11] R. Agrawal, D. Gunopulos, F. Leymann, Mining process models from workflow logs, in: EDBT '98, Springer, 1998, pp. 469–483.

[12] J. Carmona, J. Cortadella, M. Kishinevsky, A region-based algorithm for discovering Petri nets from event logs, in: BPM 2008, volume 5240 of *LNCS*, Springer, 2008, pp. 358–373.

[13] W. M. P. van der Aalst, Process Mining - Data Science in Action, Springer, 2016.

[14] J. E. Cook, A. L. Wolf, Discovering models of software processes from event-based data, TOSEM 7 (1998) 215–249.

[15] J. Herbst, A machine learning approach to workflow management, in: ECML '00, Springer, 2000, pp. 183–194.

[16] L. Maruster, W. M. P. van der Aalst, A. J. M. M. Weijters, A. van den Bosch, W. Daelemans, Automated discovery of workflow models from hospital data, in: BNAIC '01, 2001, pp. 25–26.

[17] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, A. J. M. M. Weijters, Workflow mining: A survey of issues and approaches, Data Knowl. Eng. 47 (2003) 237–267.

[18] W. M. P. van der Aalst, A. Weijters, L. Maruster, Workflow mining: Which processes can be rediscovered?, in: Eindhoven University of Technology, 2002, pp. 1–25.
[19] G. Sciavicco, M. Zavatteri, T. Villa, Mining Significant Temporal Networks Is Polynomial, in: TIME 2020, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 11:1–11:12.
[20] G. Sciavicco, M. Zavatteri, T. Villa, Mining cstnuds significant for a set of traces is polynomial, Information and Computation (in press) (2021).
[21] M. Zavatteri, R. Rizzi, T. Villa, Dynamic Controllability and (J, K)-Resiliency in Generalized Constraint Networks with Uncertainty, in: ICAPS 2020, AAAI Press, 2020, pp. 314–322.
[22] M. Zavatteri, C. Combi, L. Vigano, Resource controllability of business processes under conditional uncertainty, Journal on Data Semantics (2021) 1–21.
[23] C. Combi, R. Posenato, L. Viganò, M. Zavatteri, Conditional simple temporal networks with uncertainty and resources, Journal of Artificial Intelligence Research 64 (2019) 931–985.