Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

Dynamic Controllability and (J,K)-Resiliency in Generalized Constraint Networks with Uncertainty

Matteo Zavatteri, Romeo Rizzi, Tiziano Villa
Department of Computer Science, University of Verona, Italy
{matteo.zavatteri, romeo.rizzi, tiziano.villa}@univr.it

Abstract

A number of formal models have been proposed to address temporal and resource allocation problems under uncertainty. Such models are typically shipped with an embedded notion of *dynamic controllability*, enclosing the ability to always make the right decisions, during execution, according to the observed uncontrollable events that *always happen*. In the business process management community, resource allocation was recently studied to deal with uncontrollable choices, whereas in the security community it was studied to face the uncontrollable availability of resources. The latter is a kind of dynamic controllability known as *resiliency* where uncontrollable events *might also not happen*. To the best of our knowledge, approaches handling resiliency on top of dynamic controllability still remain unexplored. To bridge this gap, we propose *Generalized Constraint Networks with Uncertainty (GCNUs)*, a model that we devised to address resource controllability more widely, boosting expressiveness while considering several sources of uncertainty simultaneously. We define dynamic controllability and (J, K)-resiliency of GCNUs. We reason on the structure of these problems, carry out a complexity analysis and provide algorithms to solve them.

Introduction and Application Domain

In the business process management community (BPM) a workflow is a mathematical abstraction for the modeling, validation and execution of a business process. A workflow consists of a collection of tasks to be executed with respect to some partial order to achieve one or more business goals. Workflows typically deal with temporal and resource constraints, sometimes in isolation, sometimes simultaneously. Moreover, workflows might also deal with uncontrollable parts related to temporal durations, resource commitments, task executions, availability of resources, notifications of conditions, choices of the workflow path to take or, even worse, any combination of them.

When all components in a workflow are under control we deal with a *satisfiability problem* calling for a fixed solution satisfying all constraints. Instead, when some component is out of control we deal with a *dynamic controllability problem*, where the synthesis of a fixed solution is not enough. Indeed, dynamic controllability implies the existence of a *strategy* to operate (possibly differently) on the part under our control depending on the behavior of some uncontrollable events that we can only observe while executing. This means that, depending on the observed uncontrollable events, we might decide to schedule the same tasks at different times or commit different resources to them.

For example, when a patient enters the emergency room, the severity of his condition is not known a priori but it is established by a physician *while* the workflow is being executed. Since the result of this condition discriminates what medical procedures have (not) to be executed, and which resources need to be committed, our strategy must guarantee to complete the workflow by executing all relevant tasks and satisfying all relevant constraints regardless of the result of (any combination) of uncontrollable events. During execution, we can never backtrack while committing resources. This means that we must avoid situations in which, if a patient is urgent, no physician is available because we chose to assign the "wrong" physician to some previous task.

A possible way to check dynamic controllability of a workflow is to reduce it to a corresponding (temporal) constraint network for which controllability checking algorithms already exist (Combi and Posenato 2009; Combi et al. 2014; Zavatteri et al. 2017; Eder, Franceschetti, and Köpke 2018; Posenato, Zerbato, and Combi 2018).

In (Zavatteri et al. 2017), an approach to address controllability of workflows with resources and uncontrollable choices is provided. The approach maps workflow paths to *constraint networks* (Dechter 2003), relies on directional consistency (Dechter and Pearl 1987) to guarantee no backtracking when assigning users to tasks, and reasons on the intersection of common workflow paths to achieve a dynamic user assignment. After that, *constraint networks under conditional uncertainty* (Zavatteri and Viganò 2018) were provided to handle uncontrollable choices natively.

Dynamic controllability has also appeared in the security community under the name of *resiliency*. Roughly, resiliency abstracts dynamic controllability allowing that uncontrollable events might also not happen. This is different from the dynamic controllability we discussed so far, in which uncontrollable events are always going to happen.

Historically, the *workflow resiliency problem* is the problem of finding an assignment of users to tasks satisfying all constraints while coping with the absence of users during execution (Wang and Li 2010). Resiliency divides in *static*, *decremental* and *dynamic*. In *static* resiliency, up to K users might be absent before executing the workflow and never become available for that execution. In *decremental* resiliency, up to K users might be absent before or during execution and, again, they never become available for that execution (i.e., once they are gone, they are gone forever). In *dynamic* resiliency, up to K (possibly different) users might be absent *before executing any task* and they may in general turn absent and available continuously, before or during execution.

Several approaches consider static resiliency only (e.g., (dos Santos et al. 2017; Lowalekar, Tiwari, and Karlapalem 2009; Paci et al. 2008)), one of them also considers decremental resiliency (Paci et al. 2008), whereas, to the best of our knowledge, the only exact approach to all kinds of resiliency is provided in (Zavatteri and Viganò 2019b) that reduces the problem to reachability of game automata. See also (dos Santos and Ranise 2017) for a recent survey.

Recently, (Fong 2019) introduced one-shot resiliency as the first attempt to impose an upper bound on the number of times that users might turn absent (specifically one). However, one-shot resiliency is actually (one-shot) decremental resiliency as once users turn absent, they remain so.

To the best of our knowledge, further investigations on applying and bounding resiliency on top of dynamic controllability have not been extensively carried out yet.

Contributions and Organization

We first provide background on constraint networks. Then,

1. We define *Generalized Constraint Networks with Uncertainty (GCNUs)* as a core model. GCNUs extend classic constraint networks by adding variables with uncontrollable assignments, variables with uncontrollable picking, precedence constraints and abstracting the set of constraints as a boolean formula in which classic relational constraints and precedence constraints become atoms.
2. We define dynamic controllability, (J, K)-decremental and (J, K)-dynamic resiliency of GCNUs (the latter two on top of the former). We formalize these problems as two-player games between Controller and Nature and we discuss their structure. Roughly, J is the maximum number of times that values in the domains of variables might be absent, whereas K is the maximum number of values that can be absent simultaneously.
3. We prove that dynamic controllability, (J, K)-decremental and (J, K)-dynamic resiliency of GCNUs are PSPACE-complete. We give algorithms to solve them.

Finally, we compare with related work, draw conclusions and discuss future work.

Note that, for lack of space and because of its triviality, we excluded static resiliency. Indeed, in static resiliency resources turn absent maximum once, before the execution starts. As a result, we just need to check if there exists a subset of resources that once removed makes the remaining network uncontrollable.

Background

Definition 1. *A* Constraint Network (CN) *is a tuple $\langle X, V, D, C\rangle$, where $X = \{x_1, \dots, x_n\}$ is a finite set of variables, $V = \{v_1, \dots, v_m\}$ is a finite set of discrete values, $D \subseteq X \times V$ is the* domain relation *(we write $D(x) = \{v \mid (x, v) \in D\}$ to shorten the domain of $x \in X$), whereas C is a finite set of relational constraints R_{S_i} each defined over a* scope *of variables $S_i \subseteq X$ such that if $X = \{x_{i_1}, \dots, x_{i_j}\}$, then $R_{S_i} \subseteq D(x_{i_1}) \times \dots \times D(x_{i_j})$. A CN is consistent if every variable $x \in X$ can be assigned a value $v \in D(x)$ such that all constraints in C are satisfied. Deciding consistency of CNs is NP-complete.*

Generalized CNs with Uncertainty

We provide a new formalism to deal with a few sources of uncertainty simultaneously and we model the arising decision problems as two-player games between Controller and Nature. We evolve the concept of variable assignment into that of variable *execution*. Executing a variable $x \in X$ means to (i) *pick* it (i.e., choosing it from X) and (ii) *assign* it a value in $D(x)$ (in this order).

Definition 2. *A* Generalized Constraint Network with Uncertainty (GCNU) *is a tuple $\langle X, V, D, P, F\rangle$, where:*

- *$X = X_C \cup X_N$ is a finite set of variables partitioned in variables with controllable assignment (i.e., those assigned by Controller) and variables with uncontrollable assignment (i.e., those assigned by Nature), respectively.*
- *V and D are the same of those given in Definition 1.*
- *$P \subseteq X \times X$ defines the uncontrollable picking relation. We write $Y = \{y \mid (x, y) \in P\}$ to represent the set of variables with uncontrollable picking. For each $y \in Y$, $A(y) = \{x \mid (x, y) \in P\}$ represents the set of variables activating y. We say that y is* active *if all variables in $A(y)$ have been executed and y has not been executed yet. If y is active, Nature will sooner or later pick it. The graph having set of nodes X and set of edges P must be a DAG.*
- *F is a boolean formula over relational and precedence constraints that play the roles of atomic components along with $\top$ and $\bot$ representing true and false as usual. Relational constraints are the same of those discussed in Definition 1. Precedence constraints have the form $x < y$ imposing that x is executed before y.*

When $X_N = \emptyset$ and $P = \emptyset$, the GCNU is actually a generalized constraint network (GCN) *specifying no uncertainty (in that case we drop X_N and P from the specification).*

We write $x = v$ and $x \neq v$ as shorts for $R_{\{x\}} = \{(v)\}$ and $R'_{\{x\}} = D(x) \setminus R_{\{x\}}$. Despite in this paper we only use relational and precedence constraints, there is no prohibition on the usage of global constraints as further atoms to represent compactly relational ones (e.g., $\texttt{all\_diff}(x_{i_1}, \dots, x_{i_n})$ filers $D(x_{i_1}) \times \dots \times D(x_{i_n})$ by keeping all tuples not specifying the same value more than once).

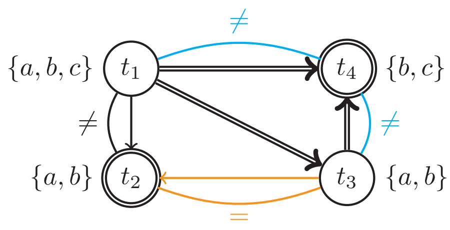


Figure 1: An example of GCNU encoding an access controlled workflow under uncertainty.

Let $N = \langle X, V, D, P, F\rangle$ be a GCNU. If F only contains precedence constraints and binary relational constraints, then we can easily represent N as a graph as follows. The set of nodes coincides with X. Domains are shown as labels near nodes. Nodes appearing as heads of double edges represent variables with uncontrollable picking, whereas double circled nodes represent variables with uncontrollable assignment (nodes having both characteristics model variables with both uncontrollable picking and assignment). The set of edges is partitioned in directed double edges, directed single edges and undirected single edges. A double edge $x \Rightarrow y$ models $(x, y) \in P$, whereas single edges model F as follows. Let $F' \equiv \mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_m$ be the m clauses obtained by computing the CNF of F. We assign a color to each clause (with the exception that each unit clause is assigned black). For each $\mathcal{C}_i$, we represent all literals in $\mathcal{C}_i$ as colored edges. If a literal is negative, we first turn it positive as follows: $\neg(x < y)$ becomes $y < x$, whereas $\neg R_{\{x,y\}}$ becomes $R'_{\{x,y\}} = (D(x) \times D(y)) \setminus R_{\{x,y\}}$. A directed edge $x \rightarrow y$ in the graph models a precedence constraint $x < y$. An undirected edge between x and y models a relational constraint $R_{\{x,y\}}$. We must always satisfy all black constraints (i.e., all unit clauses) and at least one constraint for each other color (i.e, one literal per clause).

Figure 1 depicts a GCNU encoding a workflow with four tasks t_1, t_2, t_3, t_4, three users a, b, c, two precedence constraints, three constraints imposing that the users executing t_1 and t_2, t_3 and t_4, t_1 and t_4 must be different (separation of duties), whereas those executing t_2 and t_3 must be the same (binding of duties). Moreover, t_2 and t_4 are subject to uncontrollable user assignments, whereas t_3 and t_4 are subject to uncontrollable pickings with t_3 active as soon as t_1 is executed and t_4 active as soon as both t_1 and t_3 have been executed. The precedence constraint and relational constraint in orange as well as those in blue are such that we just need to satisfy only one constraint per color. The formal specification of Figure 1 is $N = \langle X, V, D, P, F\rangle$, where $X = \{t_1, t_2, t_3, t_4\}$, $V = \{a, b, c\}$, $D(t_1) = \{a, b, c\}$, $D(t_2) = D(t_3) = \{a, b\}$, $D(t_4) = \{b, c\}$, $P = \{(t_1, t_3), (t_1, t_4), (t_3, t_4)\}$ and $F \equiv (t_1 < t_2) \wedge (t_1 \neq t_2) \wedge (t_1 \neq t_4 \vee t_3 \neq t_4) \wedge (t_3 < t_2 \vee t_2 = t_3)$.

Controllability and (J,K)-Resiliency

We define dynamic controllability and (J, K)-resiliency of GCNUs as two-player games. Let $m\colon A \rightarrow B$ be a mapping. We write $D(m) \subseteq A$ to refer to the domain on which m is actually specified. If $D(m) = A$, then m is total. Partial otherwise. We write $|D(m)|$ to refer to the cardinality of m.

Let $N = \langle X, V, D, P, F\rangle$ be a GCNU. An *assignment* is a mapping $\alpha\colon X \rightarrow V$ enforcing that for each $x \in D(\alpha)$, $\alpha(x) \in D(x)$. An assignment satisfies an $R_{\{x_{i_1},\ldots,x_{i_n}\}}$ in F iff $\{x_{i_1}, \ldots, x_{i_n}\} \subseteq D(\alpha)$ and $(\alpha(x_{i_1}), \ldots, \alpha(x_{i_n})) \in R_{\{x_{i_1},\ldots,x_{i_n}\}}$ (in symbols, $\alpha \models R_{\{x_{i_1},\ldots,x_{i_n}\}}$). An *ordering* ω is a bijection between X and the set $\{n \mid n \in \mathbb{N}, n < |X|\}$, where for each $x \in X$, $\omega(x)$ represents the index i of x in the ordering. Given a pair (α, ω), the satisfaction of F (in symbols, $(\alpha, \omega) \models F$) is defined as follows.

- $(\alpha, \omega) \models \top$ (always) and $(\alpha, \omega) \not\models \bot$ (never)
- $(\alpha, \omega) \models R_{\{x_{i_1},\ldots,x_{i_n}\}}$ iff $\alpha \models R_{\{x_{i_1},\ldots,x_{i_n}\}}$
- $(\alpha, \omega) \models x < y$ iff $\omega(x) < \omega(y)$
- $(\alpha, \omega) \models (F)$ iff $((\alpha, \omega) \models F)$
- $(\alpha, \omega) \models \neg F$ iff $(\alpha, \omega) \not\models F$
- $(\alpha, \omega) \models F_1 \Box F_2$ iff $(\alpha, \omega) \models F_1 \Box (\alpha, \omega) \models F_2$, where $\Box \in \{\wedge, \vee, \Rightarrow, \ldots\}$ is any binary boolean connective.

Dynamic Controllability

We model dynamic controllability of GCNUs as a two-player game between Controller and Nature. The game proceeds in rounds until all variables have been executed. Each round consists of three sequential phases handling overall the *picking* (in the first two) and the *assignment* (in the last one) of a single still unexecuted variable $x \in X$. Both Controller and Nature can pick and assign variables according to which sets these variables belong to resulting in 4 cases:

1. Controller picks, Controller assigns ($x \notin Y$ and $x \in X_C$)
2. Controller picks, Nature assigns ($x \notin Y$ and $x \in X_N$)
3. Nature picks, Controller assigns ($x \in Y$ and $x \in X_C$)
4. Nature picks, Nature assigns ($x \in Y$ and $x \in X_N$)

Nature has priority over Controller when both players want to pick a variable.

Game 1 (dynamic controllability)**.** *Let $N = \langle X, V, D, P, F\rangle$ be a GCNU. Let α be a partial mapping from X to V and ω be a partial mapping from X to $\{n \mid n \in \mathbb{N}, n < |X|\}$. At the beginning, $D(\alpha) = D(\omega) = \emptyset$, whereas at the end of the game α is an assignment (not necessarily total) and ω an ordering. Let $U = X \setminus D(\omega)$ be an alias for the set of unexecuted variables. Each round is as follows.*

Phase 1 *always happens. Nature makes exactly one of these two moves, choosing which one when both are available.*

npick *is available iff there exists $x \in Y \cap U$ with $A(x) \cap U = \emptyset$. Nature picks x and sets $D(\omega) = D(\omega) \cup \{x\}$ and $\omega(x) = |D(\omega)| - 1$. With this move Nature disables Phase 2 preventing Controller from picking a variable.*

pass *is available iff $U \setminus Y \neq \emptyset$. With this move Nature enables Phase 2 allowing Controller to pick a variable.*

Phase 2 *happens iff Nature passed in Phase 1.*

cpick *is always available. Controller picks $x \in U \setminus Y$ and sets $D(\omega) = D(\omega) \cup \{x\}$ and $\omega(x) = |D(\omega)| - 1$.*

Phase 3 *happens iff $D(x) \neq \emptyset$, where x is the picked variable (no matter by which player). The following two moves are mutually exclusive.*

cassign *is available iff $x \in X_C$. Controller chooses $v \in D(x)$ and sets $D(\alpha) = D(\alpha) \cup \{x\}$ and $\alpha(x) = v$.*

nassign *is available iff $x \in X_N$. Nature chooses $v \in D(x)$ and sets $D(\alpha) = D(\alpha) \cup \{x\}$ and $\alpha(x) = v$.*

When the game is over, Controller wins if α is total and $(\alpha, \omega) \models F$ (note that α is an assignment and ω an ordering). Nature wins otherwise. Thus, the game is determined.

Definition 3. *A GCNU is* dynamically controllable *if Controller has a winning strategy for Game 1.* Uncontrollable *if it is Nature that has a winning strategy for that game.*

In other words, at any round of the Game, either Nature picks a variable with uncontrollable picking or Controller picks a variable with controllable picking. The former is mandatory when Controller cannot pick any variable, whereas the latter only happens when Nature passed. Note that since P is acyclic, when all variables with controllable picking have been executed, at least a variable in Y exists and is active (otherwise, the game would already be over). After a variable has been picked (no matter by which player) Controller assigns it a value if the variable has controllable assignment, Nature otherwise. Regardless of the case, if the domain of the variable is empty, α is not extended and will not be total at the end of the game implying $(\alpha, \omega) \not\models F$.

Figure 1 is dynamically controllable. A possible winning strategy for N_2 is the following.

Controller picks t_1 and assigns c to it. Then,

1. If Nature passes, Controller picks t_2 and Nature assigns it a value in its domain. After that, Nature picks t_3 and Controller assigns to it the same value assigned to t_2. Finally, Nature picks t_4 and assigns to it a value in its domain.
2. If Nature picks t_3, then Controller assigns b to t_3. Then,
   (a) If Nature passes, Controller picks t_2 and Nature assigns it a value in its domain. After that, Controller waits for the game to finish with Nature picking t_4 and assigning it a value in its domain.
   (b) If Nature picks t_4, then she also assigns it a value in its domain. After that, Controller picks t_2 and Nature assigns it a value in its domain.

In all cases, it holds that α is total and $(\alpha, \omega) \models F$.

(J,K)-Resiliency

Resiliency adds another layer of uncertainty on top of GCNUs: *Nature might strike by removing values from V.* In what follows we formally define two kinds of (J, K)-resiliency on top of Game 1, where J and K represent the bounds of these problems (i.e., Nature's restrictions). Specifically, J is the maximum number of times that Nature can strike, whereas K is the maximum number of values that Nature can remove. Before proceeding, let us get back to the application domain discussed in the introduction to give some context. Most of times, business processes might require to be resilient only with respect to a specific subset of resources and not all of them. For example, when the employed resources are both users and machineries, we might need resiliency of personnel only, or machineries only or both. In what follows, we allow for any combination of these scenarios by partitioning V in two disjoint subset V_I and V_E modeling respectively the set of values we include and exclude from the analysis, respectively. Of course, one, none or both of these subsets may be empty.

Game 2 ((J, K)-decremental resiliency)**.** *Let N, α, ω as in Game 1 (this time $V = V_I \cup V_E$). Let $J, K \in \mathbb{N}$ such that $J \leq \min\{K, |X|\}$ and $K \leq |V_I|$. Each round is as follows.*

Phase 0 *always happens. Nature makes exactly one of these two moves, choosing which one when both are available.*

strike *is available iff $J > 0$. Nature chooses $R \subseteq V_I$ such that $0 < |R| \leq K$. Nature computes $V_I' = V_I \setminus R$, $V = V_I' \cup V_E$, $J = \min\{J-1, K-|R|\}$, $K = K-|R|$ and for each $x \in X$, $D(x) = D(x) \cap V$.*

pass *is always available. Nature does nothing.*

Phase 1, 2, 3 and winning conditions *are the same of Game 1.*

Definition 4. *A GCNU is (J, K)*-decrementally resilient *if Controller has a winning strategy for Game 2.* Breakable *if it is Nature that has a winning strategy for that game.*

In other words, Nature might strike at the beginning of maximum J different rounds by removing from V_I up to K values overall. To understand the (implicit) upper bound of J consider these two cases:

1. $\min\{K, |X|\} = K$. The slowest way in which Nature can remove all K values is to remove at most one value per round. Nature can do that since $K \leq |X|$. As a result, at each round where she strikes, J and K should be decremented by 1 (one strike has been used, one value has been removed). But this implicitly bounds Nature to strike maximum K times. Thus, J is upper bounded by K.
2. $\min\{K, |X|\} = |X|$. This case is similar but with respect to the number of rounds (that equals the number of variables). The slowest way in which Nature can remove all K values is to remove 1 value in each of the first $|X| - 1$ rounds and $K - |X| - 1$ values in the last round. Once again Nature can do so because $|X| \leq K$. As a result, J is decremented by 1 at any round, whereas K is decremented by 1 in the first $|X|-1$ rounds and by $K-|X|-1$ in the last (with respect to the original K). But this implicitly bounds Nature to strike exactly $|X|$ times. Thus, J is upper bounded by $|X|$.

Therefore, J is overall upper bounded by $\min\{K, |X|\}$.

Instead, when Nature strikes by removing more than 1 value per round, we might need to decrement J by more than 1 in order to restore its upper bound condition. Suppose that Nature removes $K' \leq K$ values from V_I. The round can end up in two possible situations: either $J-1 \leq K-K'$ (and this is still fine), or $J - 1 > K - K'$ (and this needs restoring). The first case happens when $K' \leq K - J + 1$, whereas the second one when $K' > K-J+1$. We provide the following unconditional update to handle both cases: first, we compute

$J = \min\{J-1, K-K'\}$ and then $K = K - K'$. That is, the new J and K from that point of the game on. This way, if $K' \leq K - J + 1$, then $J = \min\{J-1, K-K'\} = J - 1 \leq K - K'$, whereas if $K' > K - J + 1$, then $J = \min\{J-1, K-K'\} = K - K' \leq K - K'$. In both cases, the new J is upper bounded by the new K (which is always the previous K minus K'). Note that, at the beginning of the game, $J \leq \min\{K, |X|\}$. As a result, after the first strike, $J - 1 < |X|$ holds. That's why, it is safe for the unconditional update to focus on J and K only.

Game 3 ((J, K)-dynamic resiliency)**.** *Let N, α, ω and K as those given in Game 2. Let $J \in \mathbb{N}$ such that $J \leq |X|$ and let R be a temporary set variable. Each round is as follows.*

Phase 0 *always happens. Nature sets $R = \emptyset$ and then makes exactly one of these two moves, choosing which one when both are available.*

strike *is available iff $J > 0$ and $K > 0$. Nature chooses $K' \in \{1, K\}$, sets R to a possible K'-subset of V_I and computes $J = J - 1$.*

pass *is always available. Nature does nothing.*

Phase 1, 2, 3 and winning conditions *are the same of Game 1 with the difference that in Phase 3 Controller and Nature choose values from $D(x) \setminus R$ instead of $D(x)$.*

Definition 5. *A GCNU is* (J, K)-dynamically resilient *if Controller has a winning strategy for Game 3.* Breakable *if it is Nature that has a winning strategy for that game.*

In other words, Nature might strike at the beginning of maximum J different rounds, each time by removing from V_I up to K (possibly different) values. If Nature strikes at some round, such values are "fictitiously" removed for *that round only* and will be available again in the next. Therefore, J is upper bounded by the number of rounds only.

Decremental resiliency makes sense for $J > 0$ (note that if $K = 0$, then $J \leq \min\{K, |X|\} = 0$), whereas dynamic resiliency for both $J > 0$ and $K > 0$ (as J is not related to K), otherwise Nature can never strike and therefore Game 2 and 3 boil down to Game 1 (dynamic controllability). When $J = \min\{K, |X|\}$ (decremental) and $J = |X|$ (dynamic) no strategy of Nature is excluded. These are the worst cases in which Nature has "full power". In such cases, we just talk about *K-decremental* and *K-dynamic* resiliency.

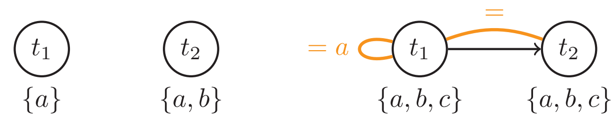
Regardless of the type of resiliency, a reader might fairly wonder what happens if in a round Nature first strikes by removing a subset of values and then in Phase 3 she also has to assign the picked variable. In that case, Nature assigns to that variable a value among those remained (if any).

Further Results on Resiliency

We conclude this section by discussing implication results among these problems and showing a few characteristics of interest of Nature's strategies.

Lemma 1. *Let $N = \langle X, V, D, P, F \rangle$ be a GCNU. Then,*

1. *K-dynamic $\Rightarrow$ K-decremental*
2. *(J, K)-dynamic $\Rightarrow$ (J', K')-dynamic, for each $J' \leq J$ and each $K' \leq K$.*



(a) $(1,1)$-dynamic $\not\Rightarrow$ $(1,1)$-decremental.

(b) No 2-winning strategy to break 2-decremental resiliency

Figure 2: Counterexamples to some implications. A self loop on x labeled by "$= v$" models the constraint $x = v$.

3. *(J, K)-decremental $\Rightarrow$ (J', K')-decremental, for each $J' \leq J$ and each $K' \leq K$ implying $J' \leq \min\{K', |X|\}$.*

Proof. A formal proof is omitted for lack of space. However, the reason is that implicated games are particular instances of implicant ones. In Case 1, Nature has full power in both games. Thus, if Nature loses the game for K-dynamic, then she also loses that for K-decremental (which is a weaker one). In Cases 2 and 3 the reason is because when Nature loses the implicant games with full power, then she will also lose the *same games* with less power (i.e., when she is restricted to strike less or remove fewer values or both). □

Note that, in general, when $1 \leq J < |X|$, then (J, K)-dynamic $\not\Rightarrow$ (J, K)-decremental. The reason is that to simulate (J, K)-decremental with (J, K)-dynamic, Nature should be able to strike in all rounds to guarantee that the once some values are gone, they are gone forever. But this is not possible since there exist $|X| - J \geq 1$ rounds in which she cannot strike. Figure 2a provides a counterexample in which we assume $V_I = V$ and $J = K = 1$.

The GCN is $(1,1)$-dynamically resilient but not $(1,1)$-decrementally resilient. Note that for decremental resiliency we have that $J = \min\{K, |X|\} = 1$. Indeed, Controller's strategy for Game 3 is the following (note that since this network is not a GCNU, Nature moves only in Phase 0, if she decides so):

1. If Nature strikes in Phase 0 of Round 1 we have two cases:
   (a) If Nature removes a, then Controller picks t_2 and assigns b to it. Then, in Round 2 Controller picks t_1 and assigns a to it.
   (b) If Nature removes b, then Controller picks t_1 and assigns a to it. Then, Controller picks t_2 and assigns either a or b to is (it does not matter).
2. If Nature passes in Phase 0 of Round 1, Controller picks t_1 and assigns a to it. Then,
   (a) If Nature strikes in Round 2 by removing either a or b, then Controller picks t_2 and assigns to it the only remained value.
   (b) If Nature passes in Round 2, then Controller picks t_2 and assigns to it any value he likes.

Instead, Nature's strategy for Game 2 is: At the beginning of Round 1, strike by removing a. Note that Controller loses because α is not total at the end.

Figure 1 is not resilient for any possible kind of resiliency. Nature strategy for all kinds of resiliency is: Strike in Round 1 by removing c. This strategy forces Controller to assign either a or b to t_1 in Round 1 (as $D(t_1) \setminus \{c\} = \{a, b\}$). After that, regardless of the Round, when Controller picks t_2, Nature can always assign to t_2 the same value assigned to t_1 violating the relational constraint between them.

Definition 6. *Nature plays according to a K-strategy if she removes exactly K values whenever she strikes.*

Lemma 2. *Let N be a GCN and let J and K as in Game 3. If Nature has a winning strategy for (J, K)-dynamic resiliency played on N, then Nature has a K-winning strategy.*

Proof. A detailed formal proof would involve a Prover, a Verifier, two parallel runs of the same game and a strategy stealing technique. But in a few sentences, the intuition is the following. Every time Nature's winning strategy suggests to strike at some round by removing a subset of values R, then Nature strikes at that very same round by removing instead $R \cup R'$, where $|R \cup R'| = K$ and $R' \subseteq V_I \setminus R$. That is, Nature possibly adds extra values to R (preserving the original ones) in order to remove a K-subset of V_I. Note that if R is already a K-subset, Nature just replicates the move (as $R' = \emptyset$). Note also that Nature can always extend R since K is never decremented in Game 3 and all removed values in a round will be available again in the next. Therefore, extending R to a K-subset R' cannot do any harm. □

Unfortunately, K-winning strategies do not exist for K-decremental resiliency played on GCNs. Figure 2b provides a counterexample for $K = 2$ (recall that $J = \min\{K, |X|\} = 2$). If Nature plays according to a 2-strategy, then she loses (note that Nature strikes maximum once as the update on J after the first strike is $J = \min\{J - 1, K - 2\} = 0$). Indeed, If Nature strikes in Round 1 by removing either $\{a, b\}$ or $\{a, c\}$ or $\{b, c\}$, then Controller picks t_1 and assigns to it either c or b or a, respectively (the remained value). After that, Controller picks t_2 and assigns to it the same value of t_1. Instead, if Nature passes in Round 1, then Controller picks t_1 and assigns a to it. After that, in Round 2, regardless of what Nature does, Controller picks t_2 and assigns to it any remained value.

Yet, the GCN is not 2-decrementally resilient at all. Nature's winning strategy is: Strike in Round 1 by removing a. Strike in Round 2 by removing the value that Controller assigned to t_1 in Round 1.

K-winning strategies do not exist for K-decremental resiliency played on GCNUs either as GCNUs contain GCNs.

Complexity of Controllability and Resiliency

We now investigate the complexity of dynamic controllability and resiliency of GNCUs.

Theorem 1. *Deciding dynamic controllability of GCNUs is PSPACE-complete.*

Proof. **Hardness:** Let $\Phi \equiv \mathcal{Q}_1 x_1, \ldots, \mathcal{Q}_n x_n \varphi$ be a QBF where $\mathcal{Q}_i \in \{\exists, \forall\}$, $1 \leq i \leq n$ and $\varphi \equiv \mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_m$ is a 3-CNF specifying m clauses over $x_1, \ldots, x_n$. We provide two polynomial time reductions to show that any source of

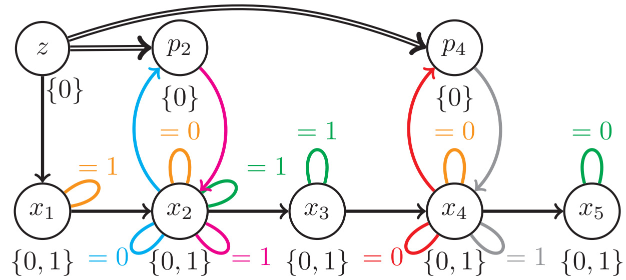


Figure 3: Reducing $\Phi \equiv \exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_2 \vee x_3 \vee \neg x_5)$ to a GCNU with uncontrollable variable pickings only.

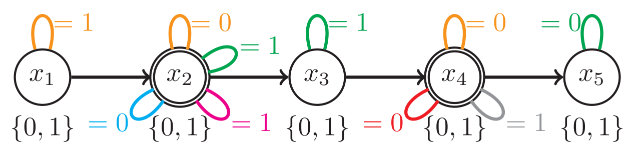


Figure 4: Reducing the same Φ of Figure 3 to a GCNU with uncontrollable variable assignments only.

uncertainty of GCNUs, even considered in isolation, makes the problem PSPACE-hard.

Reduction 1. We reduce Φ to a GCNU $N = \langle X, V, D, P, F \rangle$ with variables with uncontrollable picking as the only source of uncertainty. $V = \{0, 1\}$. $X_N = \emptyset$. X_C contains a variable z such that $D(z) = \{0\}$. For each "$\exists x_i$" in Φ, we add a variable x_i to X_C such that $D(x_i) = \{0, 1\}$. For each "$\forall x_j$" in Φ, we add a pair of variables x_j and p_j to X_C such that $D(x_j) = \{0, 1\}$, $D(p_j) = \{0\}$, $(z, p_j) \in P$ and we add to F two conjuncts $(p_j < x_j \Rightarrow x_j = 0)$ and $(x_j < p_j \Rightarrow x_j = 1)$ in order to simulate an uncontrollable value assignment to x_j depending on if Nature picks p_j before or after x_j is executed: in the first case, $x_j = 0$, whereas in the second case $x_j = 1$. Let x_1 be the first variable appearing in the quantified part of Φ. We add to F a conjunct $(z < x_1)$ as well as $n - 1$ conjuncts of the form $(x_i < x_{i+1})$ in order to mirror the order in which variables appear in the quantified part of Φ. Finally, we add m conjuncts $(x_1 = b_1 \vee x_2 = b_2 \vee x_3 = b_3)$ to encode each clause $(\lambda_1 \vee \lambda_2 \vee \lambda_3)$ in φ, where $x_i = 0$ if $\lambda_i = \neg x_i$ and $x_i = 1$ if $\lambda_i = x_i$ $(i = 1, 2, 3)$. Now, Φ is satisfiable iff N is dynamically controllable: the assignment of truth values to the quantified variables in Φ coincides with that of "$0, 1$" to the corresponding variables in N (the assignments to z and all p_j do not matter).

Reduction 2. We reduce Φ to a GCNU $N = \langle X, V, D, P, F \rangle$ with variables with uncontrollable assignment as the only source of uncertainty. We simplify the previous reduction. $V = \{0, 1\}$. For each "$\exists x_i$" in Φ, $x_i \in X_C$. For each "$\forall x_i$" in Φ, $x_i \in X_N$. For each $x \in X$, $D(x_i) = \{0, 1\}$. $P = \emptyset$. F is the same given in Reduction 1 deprived of all conjuncts involving z and p_j variables. Once again Φ is satisfiable iff N is dynamically controllable: the assignment of truth values to the quantified variables in Φ coincides with that of "$0, 1$" to the corresponding variables in N.

Membership: Algorithm 1 is a PSPACE algorithm to decide dynamic controllability of any GCNU. □

Figure 3,4 show examples of Reduction 1,2, respectively. We omit their formal specifications for lack of space.

Theorem 2. *Deciding* (J, K)-decremental *and* (J, K)-dynamic *resiliency of GCNUs are PSPACE-complete.*

Proof. **Hardness:** Inherited from Theorem 1. **Membership:** Algorithms 2 and 3 are PSPACE algorithms. □

Related Work

Dynamic user assignment was investigated for access-controlled workflows in (Zavatteri et al. 2017). After that, *constraint networks under conditional uncertainty (CNCUs)* were provided as a more formal model handling uncontrollable choices and precedence constraints natively (Zavatteri and Viganò 2018). Deciding dynamic controllability of CNCUs is PSPACE-complete (Zavatteri, Rizzi, and Villa 2019). Those works do not investigate resiliency and the only source of uncertainty is represented as uncontrollable boolean propositions labeling variables and constraints. The class of supported constraints in those works is less expressive than the one in ours. There, we have labeled relational constraints and precedence constraints, whereas here relational and precedence constraints become atoms of a boolean formula. Our framework can transparently be extended to handle global constraints as further atoms.

The workflow satisfiability problem (WSP) is the problem of finding an assignment of users to tasks satisfying all authorization constraints. The workflow resiliency problem is a dynamic WSP coping with the absence of users as the *unique source of uncertainty* (Wang and Li 2010). In static resiliency, users might be absent before starting. In decremental resiliency, users might also turn absent during execution. In dynamic resiliency, users might come and go. One-shot resiliency is the first attempt to impose restrictions on Nature by bounding the number of strikes to at most one (Fong 2019). Despite several works investigated workflow satisfiability and resiliency (dos Santos et al. 2017; Lowalekar, Tiwari, and Karlapalem 2009; Paci et al. 2008; Crampton et al. 2017; Mace, Morisset, and van Moorsel 2014; 2016) to the best of our knowledge only one faced the strategy synthesis problem in an exact way for all kinds of resiliency (Zavatteri and Viganò 2019b). A recent survey on workflow satisfiability and resiliency is given in (dos Santos and Ranise 2017). None of these works studied resiliency on top of other sources of uncertainty. Also, beside one-shot resiliency, to the best of our knowledge, no other work faced the problem of limiting Nature's power in the sense of number of strikes nor explored the arising structure. We defined (J, K)-resiliency to meet this end and we also showed non-trivial results regarding the upper bound of J and its relation with K in decremental resiliency. One-shot resiliency is equivalent to $(1, K)$-decremental resiliency.

GCNUs are a discrete event system model. The exact timing at which variables are executed is not important (we just

Algorithm 1: `GcnuDC`(N) is an AND/OR search tree whose depth size is bounded by $\mathcal{O}(|X|)$. Note that efficient algorithms to generate all K-subsets of a given set in PSPACE are discussed in (Ruskey and Williams 2009) for (s, t)-combinations.

Input: A GCNU $N = \langle X, V, D, P, F \rangle$
Output: Yes, if N is dynamically controllable. No otherwise.

`GcnuDC` (N)
  Let α, ω be an empty assignment and ordering.
  return `Phase1_2`$(X, V, D, P, F, \alpha, \omega)$

`Phase1_2` $(U, V, D, P, F, \alpha, \omega)$
  if $U = \emptyset$ **then**
    return $(\alpha, \omega) \models F$
  return `npick`$(U, V, D, P, F, \alpha, \omega) \wedge$ `cpick`$(U, V, D, P, F, \alpha, \omega)$

`npick` $(U, V, D, P, F, \alpha, \omega)$
  for $y \in Y \cap U$ **do**
    if $A(y) \cap U = \emptyset$ **then**
      $\omega' \leftarrow \omega \cup \{\omega'(y) \leftarrow |D(\omega)|\}$
      $U' \leftarrow U \setminus \{y\}$
      if $\neg$`Phase3`$(U', V, D, P, F, \alpha, \omega', y)$ **then**
        return No
  return Yes

`cpick` $(U, V, D, P, F, \alpha, \omega)$
  for $x \in U \setminus Y$ **do**
    $\omega' \leftarrow \omega \cup \{\omega'(x) \leftarrow |D(\omega)|\}$
    $U' \leftarrow U \setminus \{x\}$
    if `Phase3`$(U', V, D, P, F, \alpha, \omega', x)$ **then**
      return Yes
  return No

`Phase3` $(U, V, D, P, F, \alpha, \omega, x)$
  if $x \in X_C$ **then**
    return `cassign`$(U, V, D, P, F, \alpha, \omega, x)$
  return `nassign`$(U, V, D, P, F, \alpha, \omega, x)$

`cassign` $(U, V, D, P, F, \alpha, \omega, x)$
  for $v \in D(x)$ **do**
    $\alpha' \leftarrow \alpha \cup \{\alpha'(x) \leftarrow v\}$
    if `Phase1_2`$(U, V, D, P, F, \alpha', \omega)$ **then**
      return Yes
  return No

`nassign` $(U, V, D, P, F, \alpha, \omega, x)$
  if $D(x) = \emptyset$ **then**
    return No
  for $v \in D(x)$ **do**
    $\alpha' \leftarrow \alpha \cup \{\alpha'(x) \leftarrow v\}$
    if $\neg$`Phase1_2`$(U, V, D, P, F, \alpha', \omega)$ **then**
      return No
  return Yes

focus on the ordering). The number of resources is fixed. If a GCNU is controllable/resilient then there exists a strategy to execute it guaranteeing that when all variables have been executed all constraints are satisfied. Thus, resources never

Algorithm 2: `GcnuDecR`(N). `Phase1_2`, `upick`, `cpick`, `Phase3`, `uassign` and `cassign` are extended to carry J, K as extra parameters with `uassign` and `cassign` internally calling `Phase0` instead of `Phase1_2`. Only `strike` modifies J and K.

Input: A GCNU $N = \langle X, V = V_I \cup V_E, D, P, F\rangle$, two positive integers J, K s.t. $0 \leq J \leq \min\{K, |X|\}$ and $0 \leq K \leq |V_I|$.
Output: Yes, if N is (J, K)-decrementally resilient. No otherwise.

```
GcnuDecR (N)
    Let α, ω be an empty assignment and ordering.
    return Phase0(U, V, D, P, F, α, ω, J, K)
Phase0 (U, V, D, P, F, α, ω, J, K)
    return Phase1_2(U, V, D, P, F, α, ω, J, K) ∧
     strike(U, V, D, P, F, α, ω, J, K)
     ▷Phase1_2 = pass
strike (U, V, D, P, F, α, ω, J, K)
    if J > 0 then
        for K′ ∈ {1, K} do
            foreach K′-subset R of V_I do
                V′_I ← V_I \ R
                D′ ← {(x, v) | (x, v) ∈ D, v ∉ R}
                if ¬Phase1_2(U, V′_I ∪ V_E, D′, P, F,
                  α, ω, min(J − 1, K − K′), K − K′)
                  then return No;
    return Yes
...
```

Algorithm 3: `GcnuDynR`(N). `Phase1_2`, `upick`, `cpick`, `Phase3`, `uassign` and `cassign` are extended to carry J, K and R as extra parameters. `uassign` and `cassign` internally call `Phase0` instead of `Phase1_2`, this time looking for values in $D(x) \setminus R$. Only `strike` modifies J.

Input: A GCNU $N = \langle X, V = V_I \cup V_E, D, P, F\rangle$, two positive integers J, K such that $0 \leq J \leq |X|$ and $0 \leq K \leq |V_I|$.
Output: Yes, if N is (J, K)-dynamically resilient. No otherwise.

```
GcnuDynR (N)
    Let α, ω be an empty assignment and ordering.
    return Phase0(U, V, D, P, F, α, ω, J, K)
Phase0 (U, V, D, P, F, α, ω, J, K)
    return Phase1_2(U, V, D, P, F, α, ω, J, K, ∅) ∧
     strike(U, V, D, P, F, α, ω, J, K)
     ▷Phase1_2 = pass
strike (U, V, D, P, F, α, ω, J, K)
    if J > 0 ∧ K > 0 then
        for K′ ∈ {1, K} do
            foreach K′-subset R of V_I do
                if ¬Phase1_2(U, V, D, P, F, α, ω,
                  J − 1, K, R) then return No;
    return Yes
...
```

run out completely. As a result, there are no discrepancies between the model and the execution environment like in approaches considering robustness envelopes (Cashmore et al. 2019). GCNUs are not a stochastic framework and in contrast to POMDP (Kaelbling, Littman, and Cassandra 1998) we do not have beliefs/rewards here. So it never happens that we need to replan due to mismatching beliefs.

Several extensions of simple temporal networks (STNs) (Dechter, Meiri, and Pearl 1991) have been put forth to handle uncontrollable durations and uncontrollable choices, either in isolation (Morris, Muscettola, and Vidal 2001; Tsamardinos, Vidal, and Pollack 2003; Levine and Williams 2014; Yu, Fang, and Williams 2014; Hunsberger, Posenato, and Combi 2015) or simultaneously (Hunsberger, Posenato, and Combi 2012; Cimatti et al. 2016; Zavatteri 2017; Zavatteri and Viganò 2019a). Attempts to handle resources on top of temporal networks are given in (Combi et al. 2017; 2019). Variables with uncontrollable picking might resemble contingent time points in STNUs (Morris, Muscettola, and Vidal 2001). However, they strongly differ from them as variables with uncontrollable picking may have many activation variables whereas contingent time points have exactly one. In (Cairo et al. 2017) it was proved that temporal networks with optional variable executions can be turned into a temporal networks with no optional variable execution. GCNUs follow that philosophy guaranteeing to work with a lighter formalism. GCNUs might model optional variable executions by means of an horizon variable h such that all variables executed before h are considered really executed, whereas those executed after h are not. Note that we might impose conditional constraints in F of the form $(x < h \Rightarrow \ldots)$. In this paper we focus on a qualitative time model in which we only have a concept of before and after and for several reasons discussed above, GCNUs differ from approaches like (Laborie and Rogerie 2008). And again, no temporal network model addresses resiliency.

Conclusions and Future Work

We introduced *generalized constraint networks with uncertainty (GCNUs)* and defined dynamic controllability, (J, K)-decremental and (J, K)-dynamic resiliency. The latter two on top of the former. In dynamic controllability uncontrollable events always happen, whereas in (J, K)-resiliency they might not. We formalized the semantics for dynamic controllability and (J, K)-resiliency as two-player games. We analyzed the structure of these games. We proved that the decision versions of these problems are all PSPACE-complete. We provided algorithms to answer them.

As future work we plan to investigate strategy synthesis algorithms for these problems by relying on results such as that in Lemma 6 to achieve state space reduction techniques. We also plan to deepen complexity analysis of GCNUs for other combinations of controllability and resiliency.

References

Cairo, M.; Hunsberger, L.; Posenato, R.; and Rizzi, R. 2017. A Streamlined Model of Conditional Simple Temporal Networks - Semantics and Equivalence Results. In *TIME 2017*, 10:1–10:19. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Cashmore, M.; Cimatti, A.; Magazzeni, D.; Micheli, A.; and Zehtabi, P. 2019. Robustness envelopes for temporal plans. In *AAAI 2019*, 7538–7545. AAAI Press.

Cimatti, A.; Hunsberger, L.; Micheli, A.; Posenato, R.; and Roveri, M. 2016. Dynamic controllability via timed game automata. *Acta Informatica* 53(6-8):681–722.

Combi, C., and Posenato, R. 2009. Controllability in Temporal Conceptual Workflow Schemata. In *BPM 2009*, 64–79. Springer.

Combi, C.; Gambini, M.; Migliorini, S.; and Posenato, R. 2014. Representing business processes through a temporal data-centric workflow modeling language: An application to the management of clinical pathways. *IEEE T. Systems, Man, and Cybernetics: Systems* 44(9):1182–1203.

Combi, C.; Posenato, R.; Viganò, L.; and Zavatteri, M. 2017. Access controlled temporal networks. In *ICAART 2017*, 118–131. ScitePress.

Combi, C.; Posenato, R.; Viganò, L.; and Zavatteri, M. 2019. Conditional simple temporal networks with uncertainty and resources. *JAIR* 64:931–985.

Crampton, J.; Gutin, G.; Karapetyan, D.; and Watrigant, R. 2017. The bi-objective workflow satisfiability problem and workflow resiliency. *JCS* 25(1):83–115.

Dechter, R., and Pearl, J. 1987. Network-based heuristics for constraint-satisfaction problems. *Artif. Intell.* 34(1).

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artif. Intell.* 49(1-3):61–95.

Dechter, R. 2003. *Constraint processing*. Elsevier.

dos Santos, D. R., and Ranise, S. 2017. A survey on workflow satisfiability, resiliency, and related problems. http://arxiv.org/abs/1706.07205.

dos Santos, D. R.; Ranise, S.; Compagna, L.; and Ponta, S. E. 2017. Automatically finding execution scenarios to deploy security-sensitive workflows. *JCS* 25(3):255–282.

Eder, J.; Franceschetti, M.; and Köpke, J. 2018. Controllability of orchestrations with temporal sla: Encoding temporal xor in cstnud. In *iiWAS 2018*, 234–242. ACM.

Fong, P. W. L. 2019. Results in workflow resiliency: Complexity, new formulation, and asp encoding. In *CODASPY'19*, 185–196. ACM.

Hunsberger, L.; Posenato, R.; and Combi, C. 2012. The Dynamic Controllability of Conditional STNs with Uncertainty. In *PlanEx at ICAPS 2012*, 1–8.

Hunsberger, L.; Posenato, R.; and Combi, C. 2015. A Sound-and-Complete Propagation-based Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal Networks. In *TIME 2015*, 4–18. IEEE.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artif. Intell.* 101(1-2):99–134.

Laborie, P., and Rogerie, J. 2008. Reasoning with conditional time-intervals. In *FLAIRS 2008*, 555–560. AAAI.

Levine, S. J., and Williams, B. C. 2014. Concurrent Plan Recognition and Execution for Human-Robot Teams. In *ICAPS 2014*. AAAI Press.

Lowalekar, M.; Tiwari, R. K.; and Karlapalem, K. 2009. Security policy satisfiability and failure resilience in workflows. In *FIDIS Summer School*, 197–210. Springer.

Mace, J. C.; Morisset, C.; and van Moorsel, A. 2014. Quantitative Workflow Resiliency. In *ESORICS 2014*, 344–361. Springer.

Mace, J. C.; Morisset, C.; and van Moorsel, A. 2016. WRAD: Tool Support for Workflow Resiliency Analysis and Design. In *SERENE 2016*, 79–87. Springer.

Morris, P. H.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *IJCAI 2001*, 494–502. Morgan Kaufmann Publishers Inc.

Paci, F.; Ferrini, R.; Sun, Y.; and Bertino, E. 2008. Authorization and User Failure Resiliency for WS-BPEL Business Processes. In *ICSOC '08*, 116–131. Springer.

Posenato, R.; Zerbato, F.; and Combi, C. 2018. Managing Decision Tasks and Events in Time-Aware Business Process Models. In *BPM 2018*, 102–118. Springer.

Ruskey, F., and Williams, A. 2009. The coolest way to generate combinations. *Discrete Math.* 309(17):5305–5320.

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

Wang, Q., and Li, N. 2010. Satisfiability and Resiliency in Workflow Authorization Systems. *TISSEC* 13(4).

Yu, P.; Fang, C.; and Williams, B. C. 2014. Resolving uncontrollable conditional temporal problems using continuous relaxations. In *ICAPS 2014*. AAAI Press.

Zavatteri, M., and Viganò, L. 2018. Constraint networks under conditional uncertainty. In *ICAART 2018*, 41–52. SciTePress.

Zavatteri, M., and Viganò, L. 2019a. Conditional simple temporal networks with uncertainty and decisions. *TCS* 797:77–101.

Zavatteri, M., and Viganò, L. 2019b. Last man standing: Static, decremental and dynamic resiliency via controller synthesis. *JCS* 27(3):343–373.

Zavatteri, M.; Combi, C.; Posenato, R.; and Viganò, L. 2017. Weak, strong and dynamic controllability of access-controlled workflows under conditional uncertainty. In *BPM 2017*, 235–251. Springer.

Zavatteri, M.; Rizzi, R.; and Villa, T. 2019. Complexity of Weak, Strong and Dynamic Controllability of CNCUs. In *OVERLAY 2019*, volume 2509, 83–88. CEUR-WS.org.

Zavatteri, M. 2017. Conditional simple temporal networks with uncertainty and decisions. In *TIME 2017*, 23:1–23:17. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.