Reactive Synthesis for Expected Impacts

Emanuele Chini
Department of Computer, Control and Management Engineering, University "La Sapienza", Rome, Italy.
emanuele.chini@uniroma1.it
Department of Computer Science, University of Verona, Verona (Italy)
emanuele.chini@univr.it

Pietro Sala
Department of Computer Science, University of Verona, Verona (Italy)
pietro.sala@univr.it

Andrea Simonetti
Department of Computer Science, University of Verona, Verona (Italy)
andrea.simonetti@studenti.univr.it

Omid Zare
Department of Computer Science, University of Verona, Verona (Italy)
omid.zare@univr.it

As business processes become increasingly complex, effectively modeling decision points, their likelihood, and resource consumption is crucial for optimizing operations. To address this challenge, this paper introduces a formal extension of the Business Process Model and Notation (BPMN) that incorporates choices, probabilities, and impacts, referred to as BPMN+CPI. This extension is motivated by the growing emphasis on precise control within business process management, where carefully selecting decision pathways in repeated instances is crucial for conforming to certain standards of multiple resource consumption and environmental impacts. In this context we deal with the problem of synthesizing a strategy (if any) that guarantees that the expected impacts on repeated execution of the input process are below a given threshold. We show that this problem belongs to PSPACE complexity class; moreover we provide an effective procedure for computing a strategy (if present).

1 Introduction

BPMN (Business Process Model and Notation) has emerged as a pivotal formalism in the realm of process management, offering a standardized method for detailing business processes in various sectors, including healthcare and industry. Its graphical notation facilitates the clear and precise representation of process flows, enabling stakeholders to comprehend, analyze, and improve business operations. In the healthcare sector, BPMN plays a critical role in implementing patient care guidelines [26]. Similarly, in the industrial domain, it aids in the efficient management of manufacturing and supply chain processes, ensuring timely delivery of products and services [15]. In these domains, increasing attention has arisen in the past decade on the topic of Business Processes Management, where the choice of traces on the control side is paramount. These applications demand measurement and employ, as a means for selection, notions such as *cost-awareness*[23], *energy-awareness*[5], and *resource-awareness* [11], which naturally induce scenarios where multiple measurements must be controlled.

In this paper, we proceed under the implicit assumption that all costs, energies, and resources utilized are positive and exhibit additive characteristics. This implies that our process instances solely deplete resources to fulfill their objectives without the capability to generate resources. As we will demonstrate, this restriction contributes to favorable computational properties.

Moreover, we use the probabilistic split, referred to as *nature*, which signifies a decision based on a probability distribution beyond the worker's control. For instance, in healthcare, a *nature* is the chance

A. Achilleos and A. Francalanza (Eds.): Fifteenth International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 2024).
EPTCS ??, 2024, pp. 1–18, doi:10.4204/EPTCS.??.??

of developing gastritis when taking Brufen 600 with a probability of 1%. Similarly, in industrial applications, machinery wear and tear may influence the production process, requiring maintenance stops during production.

Finally, time consumption for tasks is considered, as they will be equipped with specific durations.

Our approach here is twofold. First, we aim to introduce a formal BPMN extension that addresses execution in the presence of all the previously mentioned components, namely, BPMN plus Choices/ Probability/Impacts (BPMN+CPI). Next, we seek to provide a dynamic control mechanism, i.e., a strategy, for BPMN execution. This is to ensure, where possible, that the expected impacts remain below a set of user-defined thresholds. To elegantly juggle all these concepts within a single framework, we enrich the standard Petri Net semantics for BPMN to capture impacts, durations, and probabilities. We call this model of computation the Simultaneous Probabilistic Impactful Network (SPIN). We then define a graph representing all possible executions of SPIN. This graph is combined with a natural modification of classical reachability games to derive the desired strategy, if any. The primary aim of this study is to determine, for a process formalized in BPMN+CPI, whether a controller exists that can accurately execute each step of the process while ensuring that the expected value of each resource, across repeated process instances, remains within predefined thresholds.

Upon establishing the computational model for BPMN+CPI, we tackle the challenge of synthesizing a strategy for a specified process in BPMN+CPI, given a set of expected value thresholds. This is achieved through the following steps:

1. *Semantics by Petri Nets*. After defining how to translate a BPMN+CPI into a SPIN, we define the semantics of both of them by giving the semantics of SPIN alone as an extension of classical Petri net semantics. This includes introducing time durations for places, probabilistic transitions, and the possibility (under certain conditions) of executing a set of enabled transitions simultaneously instead of one at a time;
2. *Computation Graph*. All possible computations for the given SPIN are represented as a graph. In this graph, each node represents a path of executions, and any edge between two computations indicates that the source computation can be extended to the target computation by firing one or more enabled transitions in the source computation;
3. *Classical Reachability Game Graph Transformation* [27]. By transforming the computation graph into a classical reachability game graph, where spoiler nodes (typically denoted by $\Box$) represent choices made by nature, we assess the existence of a "good" set of final states that can "attract" the initial state. If such a set exists, we can infer the existence of our strategy.

The paper is organized as follows. In Section 2, we present and describe related work and the state-of-the-art algorithms for finding strategies in computational models that can encode BPMN+CPI through suitable translation. In Section 3, we illustrate a practical example of a BPMN process in an industrial setting, followed by a formal definition of the BPMN+CPI model, detailing the components of choices, probabilities, and impacts. Since we restrict ourselves to acyclic graphs, at the end of this section, we briefly discuss a simple way of dealing with loops within the proposed framework. In Section 4, we provide the complexity bounds for the strategy synthesis problem for BPMN+CPI. While Section 4 deals with the decision problem of establishing whether a strategy exists or not, Section 5 focuses on effectively synthesizing a strategy given a BPMN+CPI process and a bound for expected impacts. Finally, Section 6 highlights our main findings, their theoretical and practical impacts, and future research avenues.

Methods	Costs	Durations	Strategy
UPPAAL-Stratego	multiple, not considered for strategy	explicitly defined, time is continuous	•non-deterministic • state explosion due to subset construction
PRISM	multiple negative allowed	implicit via multiple states	• ϵ-approximated strategy • increases exponentially w.r.t $1/\epsilon$
MPG-MDP	multiple negative allowed	implicit via multiple states	• infinite plays • BPMN+CPI would need difficult encoding • game averages values on a per-step basis
Our method	multiple, only positive	explicitly defined	• deterministic • exact strategy by integrating rewards and probabilities • game averages values on a per-instance basis

Table 1: A summary of the features of the tool introduced in this study and the problems addressed by UPPAAL-Stratego, PRISM, and MPG-MDP, respectively.

2 Related work

The most commonly accepted semantics for BPMN processes, used for both formal tasks like monitoring, verification, and querying, and application-driven tasks like process discovery and execution forecasting, is the Petri Net semantics. In this approach, a BPMN process is mapped into a Petri Net [12]. This mapping retains several beneficial properties, including the crucial feature that the resulting net is 1-bounded [6], meaning that from an initial state with one token, all configurations will have at most one token per place. Under this 1-boundedness assumption, the Petri Net reduces to an exponentially succinct representation of a finite automaton (FA) [18], where all labelings can be represented as sets of places holding one token, making the number of states finite. If the language of this automaton is defined by its transitions, the resulting FA is deterministic (DFA). Thus, many formal problems, such as querying, emptiness checking, strategy synthesis (reachability games), and Linear Temporal Logic (LTL) model checking, can be equivalently viewed in BPMN, 1-bounded Petri Nets, or succinct DFAs, as transformations between these representations can be performed in LOGSPACE.

Incorporating resources into BPMN processes is well-explored in process optimization literature. In [23], an extension of the classical BPMN notation is proposed to evaluate the overall cost of process diagrams, comparing costs associated with tasks as single values or intervals to find the most cost-effective way to perform the intended job. Our contribution specifically focuses on the positive impacts of such integration, further allowing the specification of impacts as arrays of cost values to express monetary costs and other resources or requirements. In [9], Combi et al. outlined a method for enforcing distinctive temporal behaviors by introducing temporal patterns (e.g., minimum and/or maximum durations) linked to tasks. They proposed creating reusable, duration-aware process models using existing BPMN elements, capturing duration constraints at various abstraction levels, and checking for duration constraint violations at runtime. Duran et al. [14] introduced a rewriting logic executable specification of BPMN extended with time and probabilities, allowing stochastic expressions to specify task durations and flow delays. Herbert et al. [19] formalize an extension of the BPMN language incorporating probabilistic non-deterministic branching. Additionally, they present an algorithm for translating such models into MDPs expressed in the syntax of the PRISM model checker [22]. This facilitates precise quantitative analysis of business processes. We have adopted a similar extension of BPMN to introduce non-deterministic behaviour (for nature nodes), which is frequently observed in real-world application scenarios. Probabilities are linked to gateway branching behaviors, enabling discrete-event simulation and automatic stochastic

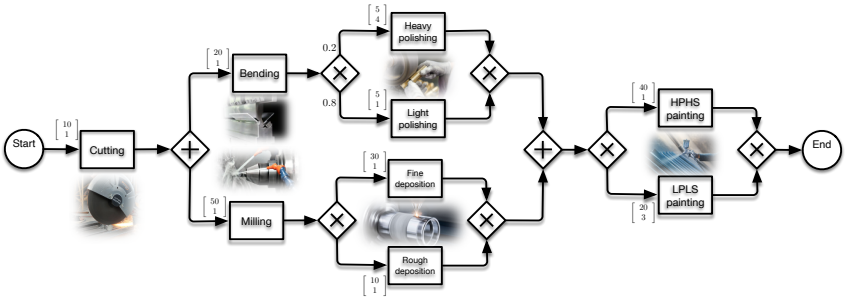


Figure 1: An example of BPMN+CPI diagram for an industrial process.

verification of various properties. Our work will consider task durations by imposing stringent time constraints, ensuring that each task extends over a time interval precisely equal to its duration, which possibly affects which choice is enabled first in a given execution. Additionally, incorporating probabilities into BPMN situates our research within the specialized domain of Markov Decision Processes (MDPs) [17], significantly enhancing the applicability of BPMN in decision-making under uncertainty.

Moving beyond BPMN, our approach primarily involves devising a strategy within an MDP enhanced with vectors of positive impacts. The objective is to ensure that the strategy's expected value does not exceed a specific threshold. The realm of strategy synthesis for MDPs has been extensively explored, leading to notable breakthroughs like the PRISM model checker [22]. PRISM has emerged as a key instrument, evolving over time to incorporate sophisticated features for strategizing within MDP contexts. Another notable development is UPPAAL-Stratego [10], an extension of the well-regarded UPPAAL-TIGA [4], which solves the strategy synthesis problem for games played on timed automata incorporating both costs and probabilities. From a theoretical perspective, albeit less focused on specific tools, our issue shares similarities with Mean Payoff Games (MPG)[28] as applied to MDP (MPG-MDP)[8]. The differences and similarities between our proposed method and the current state of the art are concisely summarized in Table 1. While we focus on a system with probabilities, we are aware of other formalisms that allow impact vectors with negative contributions, such as infinite energy games [2].

3 BPMN+CPI: Processes with Choices, Probabilities, and Impacts

In this section, we begin by informally illustrating the concept of BPMN+CPI through an intuitive example of a metal manufacturing process together with an initial, intuitive understanding of the expected impacts induced by a strategy in Section 3.1. These concepts are then formalized in Section 3.2, where we also state the core problem of this work: finding an optimal strategy that minimizes the overall impact. Finally, in Section 3.3, we discuss the advantages and drawbacks of reducing diagrams with loops to acyclic ones from the perspective of strategy synthesis.

3.1 Motivating Example

The BPMN+CPI diagram of Figure 1 depicts a metal manufacturing process that involves cutting, milling, bending, polishing, depositioning, and painting a metal piece. It consists of a single-entry-single-exit (SESE) diagram, with a choice, a nature, and an impact for each task, which is defined as a numbers vector. The bracketed numbers next to each activity represent impact vectors $\begin{bmatrix} a \\ b \end{bmatrix}$ where a = cost of the task and b = hours/men required to complete the task. For instance, cutting the metal piece costs 10 units (e.g.,

currency, resource, etc.), and requires 1 unit of time or manpower (e.g., 1 hour or 1 worker). In Figure 1, the nature's probability of each chosen path is indicated with the numbers next to decision points. For example, there's a high probability (0.8) of the process moving from bending to light polishing and a low probability (0.2) of it moving to fine heavy polishing.

Whenever the process is executed, the worker and nature make a series of choices, which result in a path executed on the BPMN with a total impact vector for that specific instance. Let's now assume that, for economic reasons, the process must stay within a certain bound. Therefore, our interest is always to stay below that bound. However, we have to consider that the path also depends on the natures within the process, of which we do not know the choice a priori, but we only have the probability of going one way or the other. Consequently, we can formulate a strategy, defined as a series of choices taken while considering the nature and a maximum expected impact, to manage to reach the end of the process with a certain impact vector.

Strategy example: after cutting the metal piece, we have two tasks after the parallel split node, so we do the bending and milling in parallel. Then, after milling we have two options to choose from, here we choose fine deposition. After bending, we have two options to choose from: we choose light polishing with the probability of 0.8. Then, we have two final tasks to choose from: we select LPLS painting. Finally, we have the maximum expected impact of $\begin{bmatrix}115\\11\end{bmatrix}\times 0.2+\begin{bmatrix}135\\8\end{bmatrix}\times 0.8=\begin{bmatrix}131\\8.6\end{bmatrix}$.

A strategy is defined as winning only if the expected impact vector is below the bound. Therefore, the goal is to find a winning strategy. Consider, for example, that you want to keep the BPMN+CPI visible in Figure 1 under the limit of $ei=\begin{bmatrix}155\\7.5\end{bmatrix}$. In this case, the strategy shown is not a winning strategy. In fact, it presents a maximum expected impact greater than the bound ei. Below we propose an example of a winning strategy.

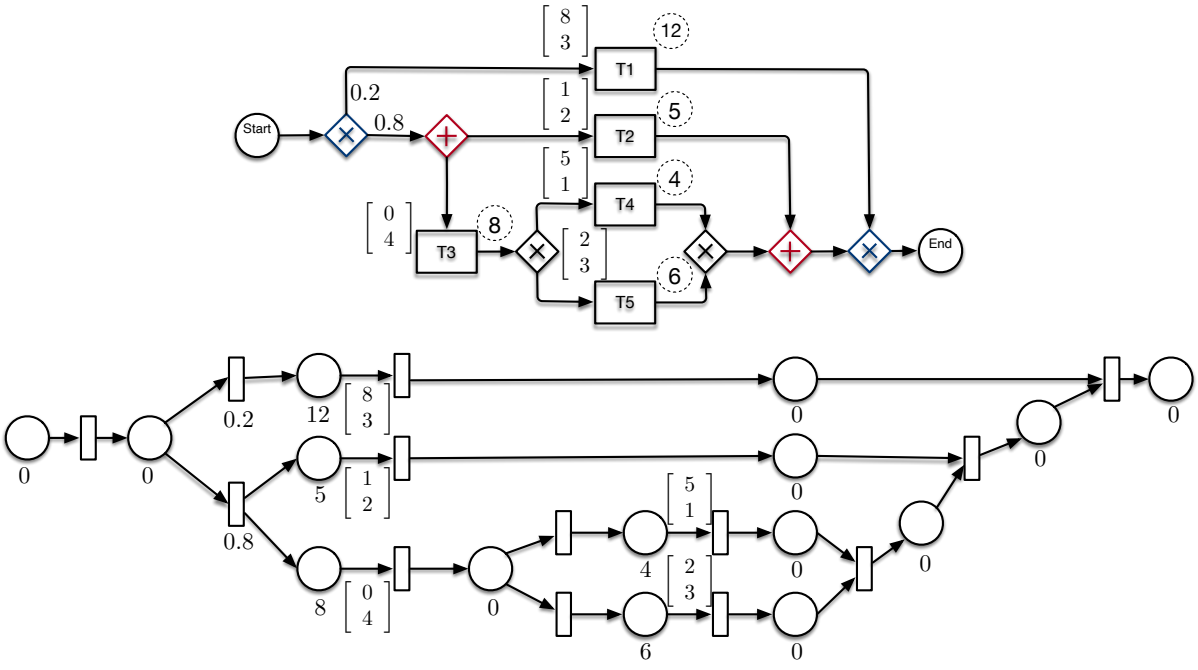
Wining strategy example: after cutting we perform milling in parallel with bending. we have two options that come after milling; we choose fine deposition. We have two options to choose from after bending; we choose light polishing with a probability of 0.8. Then, we have two final tasks to choose from and select HPHS painting this time. Finally, we have $\begin{bmatrix}135\\9\end{bmatrix}\times 0.2+\begin{bmatrix}155\\6\end{bmatrix}\times 0.8=\begin{bmatrix}151\\6.6\end{bmatrix}\leq ei$, so this strategy successfully keeps the overall impact below the expected impact.

3.2 Problem Formulation

In this section, we formally state the BPMN+CPI semantics. First, we define the concept of Structured Single-Entry Single-Exit (SESE) BPMN, Figure 2, as follows.

Definition 1. *A* structured single-entry-single-exit diagram*, from now on simply a SESE diagram, is a directed graph* $\mathcal{D}=(V,E,E_{\top},\mathcal{T})$ *where* (V,E) *is a directed graph,* $E_{\top}\subseteq E$, $\mathcal{T}:V\to\{event,task,join,split\}$ *such that:*

1. *for each* $v\in V$ *if* $\mathcal{T}(v)=event$ *then there exists at most one edge departing from* v*, there exists at most one edge entering* v*, and at least one edge departing from* v *or entering* v*, i.e.,* $|\{(v,v')\in E\}|\leq 1$, $|\{(v',v)\in E\}|\leq 1$, *and* $|\{(v',v)\in E\}\cup\{(v,v')\in E\}|>0$*;*
2. *there exists exactly two distinct nodes* $\hat{v},\check{v}$ *in* V *such that* $\hat{v}$ *has not incoming edges and* $\check{v}$ *has not outgoing edges, i.e.,* $\{(v,\hat{v})\in E\}=\{(\check{v},v)\in E\}=\emptyset$*;*
3. *for each* $v\in V$ *if* $\mathcal{T}(v)=task$ *there exists exactly one edge departing from* v *and one edge entering* v*, i.e.,* $|\{(v,v')\in E\}|=|\{(v',v)\in E\}|=1$*;*
4. *for each* $v\in V$ *if* $\mathcal{T}(v)=split$ *there exists exactly two edges departing from* v *and one edge entering* v*, i.e.,* $|\{(v,v')\in E\}|=2$ *and* $|\{(v',v)\in E\}|=1$*;*
5. $E_{\top}\subseteq\{(v,v'):\mathcal{T}(v)=split\}$ *and for each* $v\in V$ *if* $\mathcal{T}(v)=split$ *we have* $|\{v':(v,v')\in E_{\top}\}|=1$*;*



(b)

Figure 2: A BPMN+CPI utilizing all the components considered in this work (a) and its SPIN translation (b).

6. *for each* $v \in V$ *if* $\mathcal{T}(v) = join$ *there exists exactly one edge departing from* v *and two edges entering* v*, i.e.,* $|\{(v,v') \in E\}| = 1$ *and* $|\{(v',v) \in E\}| = 2$*;*

Every non-SESE BPMN diagram can be translated into a SESE diagram as demonstrated in [13].

In particular, in the rest of this work, we will restrict ourselves to acyclic SESE diagrams. We will discuss this limitation and how it can be overcome in Section 3.3.

We define BPMN+CPI processes as follows.

Definition 2. *A BPMN+CPI is a tuple* $Pcpi = (\mathcal{D}, \mathcal{P}, \mathcal{I}, \delta)$ *where* $\mathcal{D} = (V, E, \mathcal{T})$ *is a SESE diagram, and* $\mathcal{P} : split(V) \to \mathbb{R}_{[0,1]}$ *is a* partial function*,* $\mathcal{I} : task(V) \to (\mathbb{R}_{\geq_0})^k$ *with* $k \in \mathbb{N}$*, and* $\delta : task(V) \to \mathbb{N}^+$*.*

Let us notice that since $\mathcal{P}$ is a partial function, it suffices to encode the natural split gateways in the diagrams, i.e., the one with associated probabilities. Then we may define V_{nature} as the set $V_{nature} = Dom(\mathcal{P})$ and, on the other hand, for the choice of the system V_{choice} as $V_{choice} = split(V) \setminus V_{nature}$.

Let us now extend the semantics of classical Petri nets [25] in order to capture the semantics of BPMN+CPI process.

Definition 3. *A Simultaneous Probabilistic Impactful Network (*SPIN*) is a tuple* $N = (PT = P \cup T, T_p, \Delta, I, Pr, D)$ *where* P *and* T *are finite disjoint set of places and transition, respectively,* $T_p \subseteq T$*,* $\Delta \subseteq (P \times T) \cup (T \times P)$*,* $I : T \to \mathbb{N}^k$*,* $D : P \to \mathbb{N}$*, and* $Pr : T_p \to [0,1]$*.*

Given a SPIN $N = (PT = P \cup T, T_p, \Delta, I, Pr, D)$ for each $pt \in PT$ let $incoming(pt) = \{pt' \in PT : (pt', pt) \in \Delta\}$ and let $outgoing(pt) = \{pt' \in PT : (pt, pt') \in \Delta\}$. Here we focus on a specific restriction of SPIN called *structured acyclic* SPIN.

Definition 4. *We say that a* SPIN $N = (PT = P \cup T, T_p, \Delta, I, Pr, D)$ *is* structured and acyclic *if and only if the directed graph* (PT, Δ) *is acyclic, and the following conditions hold:*

1. *for each* $pt \in PT$ *we have,* $|incoming(pt)| \leq 2$ $|outgoing(pt)| \leq 2$ *and* $|incoming(pt)| + |outgoing(pt)| \leq 3$*;*
2. *there exists a unique partition* $\mathcal{T}_p = \{t_1, \bar{t}_1\}, \ldots, \{t_m, \bar{t}_m\}$ *of* T_p *such that* $Pr(t_i) = 1 - Pr(\bar{t}_i)$*,* $|outgoing(t_i)| = |outgoing(\bar{t}_i)| = 1$*, and* $incoming(t_i) = incoming(\bar{t}_i)$*;*
3. *there exists a unique set cover* $PT_1, \ldots PT_m$ *of* PT *such that for each pair* PT_i, PT_j *of the cover* $PT_i \cup PT_j$ *also belongs to the cover and the following conditions hold:*
   - $pt \in PT$ *we have that there exists at most two incoming and two outgoing edges, and the cardinality of the incoming and outgoing edges is at most* 3*, i.e.,* $\{(pt', pt)\}$*.*
   - *for each pair* PT_i, PT_j $PT_i \cap PT_j = \emptyset$*, or* $PT_i \subseteq PT_j$*, or* $PT_j \subseteq PT_i$*;*
   - *for each* $PT_i \neq PT$ *there exists a unique element* $pt_{in(i)} \in PT_i$ *(resp.,* $pt_{out(i)} \in PT_i$*) such that* $\{pt_{in(i)}\} = \{pt : (pt', pt) \in \Delta, pt' \notin PT_i, pt \in PT_i\}$ *(resp.,* $\{pt_{out(i)}\} = \{pt : (pt, pt') \in \Delta, pt' \notin PT_i, pt \in PT_i\}$*);*
   - *for each* $PT_i \neq PT$ *all the elements of* PT_i *are reachable from* $pt_{in(i)}$ *via* Δ *and all the elements of* PT_i *can reach* $pt_{out(i)}$ *via* Δ*.*

The class of SPIN, as captured by Definition 4, is the counterpart of acyclic BPMN+CPI. The formal translation from BPMN+CPI to SPIN provided which enriches the work [12], is not here shown for the sake of brevity. However, an example that includes the main BMPN elements is shown in Figure 2.

Let us notice that by the above definition a structured acyclic SPIN, a SPIN from now on, features exactly one place p_0 with $incoming(p_0) = \emptyset$ and a unique place p_f with $outgoing(p_f) = \emptyset$. Let us define a switch function $sw : T_p \to T_p$ such that for every $t \in T_p$ $\{sw(t), t'\} \in \mathcal{T}_p$. Basically sw act as a tool that allow us, for every probabilistic transition t, to access the unique other probabilistic $\bar{t}$ transition which shares the same incoming place of t.

Let us now formally define how computations work for SPINs. Given a SPIN $N = (PT = P \cup T, T_p, \Delta, I, Pr, D)$, a state $q : P \to \mathbb{N} \cup \{\epsilon\}$ is a function that maps places in temporal units, where ϵ states that the specific place has not been visited yet, or that it has already been visited.

Initial state q_0 and final state q_f for a SPIN are defined as follows:

$$q_0(p) = \begin{cases} 0 & \text{if } p = p_0 \\ \epsilon & \text{otherwise} \end{cases} \quad ; \quad q_f(p) = \begin{cases} 0 & \text{if } p = p_f \\ \epsilon & \text{otherwise} \end{cases} .$$

We will say that a transition $t \in T$ is enabled in a state q if and only if, for all $p \in incoming(t)$, $q(p) \geq D(p)$. Let us introduce now the concept of saturated state.

Definition 5. *Given a state q for a spin* $N = (PT = P \cup T, T_p, \Delta, I, Pr, D)$ *we say that q is saturated if and only if there exists at least one transition* $t \in T$ *which is enabled in q*

Since in a not saturated state q no transition $t \in T$ is enabled the net will be stuck in q. Then the intuition behind not saturated states is that the corresponding BPMN+CPI process is waiting for one or more tasks to terminate before going further. For getting out of such not saturated states we introduce a special transition t_w, the so called *wait transition* which encode the passing of one time units and it is enabled only in not saturated states.

Unlike classical Petri Nets, where each transition is fired one at the time here may fire either t_w or a subset of T called maximal non-conflicting enabled transition set.

Definition 6. *Given a state q for a spin* $N = (PT = P \cup T, T_p, \Delta, I, Pr, D)$ *and a subset* $\overline{T} \subseteq T$ *we say that* $\overline{T}$ *is a* maximal non-conflicting enabled transition set*, MNCE for short, in q if and only if the following conditions hold:*

1. *for each* $t \in \overline{T}$ *we have that t is enabled in q (*enabled*);*
2. *for each* $t, t' \in \overline{T}$ *with* $t \neq t'$ *we have* $(incoming(t) \cup outgoing(t)) \cap (incoming(t') \cup outgoing(t')) = \emptyset$ *(*non-conflicting*);*
3. *for any* $t \in T \setminus \overline{T}$ *we have that* $\overline{T} \cup \{t\}$ *violates the above two conditions (*maximal*);*

Given a set of transitions $\overline{T} \subseteq T$, let

$$OutPlaces(\overline{T}) = \bigcup_{t \in \overline{T}} outgoing(t)$$

and let

$$Places(\overline{T}) = \bigcup_{t \in \overline{T}} incoming(t) \cup OutPlaces(\overline{T}).$$

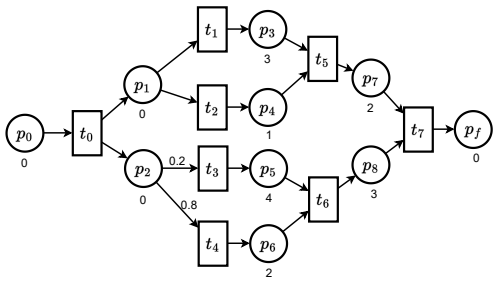


Figure 3: A SPIN for illustrating *MNCE* and probabilistic variants.

Now we are ready to define the transition relation between states in a SPIN. Let $N = (PT = P \cup T, T_p, \Delta, I, Pr, D)$ a spin for any pair of states q,q' for it we have:

$$q \xrightarrow{t_w} q' \textbf{ iff } \begin{array}{c} q \text{ is not saturated} \\ \text{and} \\ q'(p) = \begin{cases} q(p)+1 & \text{if } q(p) \in \mathbb{N} \\ \epsilon & \text{otherwise} \end{cases} \end{array} ; \quad q \xrightarrow{\overline{T}} q' \textbf{ iff } \begin{array}{c} q \text{ is saturated, } \overline{T} \text{ is an } \textit{MNCE} \text{ in } q, \\ \text{and} \\ q'(p) = \begin{cases} q(p)+1 & \text{if } q(p) \in \mathbb{N} \text{ and} \\ & p \notin Places(\overline{T}) \\ 0 & \text{if } p \in OutPlaces(\overline{T}) \\ \epsilon & \text{otherwise} \end{cases} \end{array}.$$

Definition 7. *A computation $c = q_0 \xrightarrow{\overline{T_1}} \dots \xrightarrow{\overline{T_n}} q$ in a* SPIN *is a sequence of sets of transitions $\overline{T_i}$ where for each $1 \leq i \leq n$ we have that $\overline{T_i}$ is either t_w or an MNCE for q_{i-1}.*

A computation $c = q_0 \xrightarrow{\overline{T_1}} \dots \xrightarrow{\overline{T_n}} q$ in a SPIN is called a final computation if $q = q_f$. Stated that $I(t_w) = 0^k$ we can compute $I(c) = \sum_{t \in \bigcup_{i=1}^{n} \overline{T_i}} I(t)$ the impact associated with the computation c and $p(c) = \prod_{t \in \bigcup_{i=1}^{n} \overline{T_i} \cap T_p} Pr(t)$, the probability associated with the computation c. Let $T_{\not p} = T \setminus T_p$, that is, the set of transitions devoid of probabilistic transition, a strategy is defined as follows.

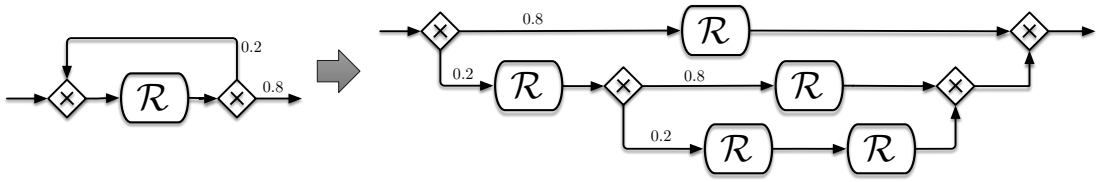
Definition 8. *Let $\mathbb{C}$ be the set of all the computations for a* SPIN*, we can define a strategy $S : \mathbb{C} \to 2^{T_{\not p}} \cup \{t_w\}$, a function that maps computations either into subsets of $T_{\not p}$ or into t_w.*

So, starting from a computation c in which we have reached the last state of the sequence, a strategy $S(c)$ tells us which are the next non-probabilistic transitions that are going to be *fired*. For all computations $c = q_0 \xrightarrow{\overline{T_1}} \dots \xrightarrow{\overline{T_n}} q$ we implicitly assume that $S(c)$ is t_w if q is not saturated and for and does not exists an enabled transition $t \in T_{\not p} \setminus S(c)$ such that $t \cup S(c)$ is non-conflicting, i.e., $S(c)$ may always be completed into an *MNCE* for q. Given a computation $c = q_0 \xrightarrow{\overline{T_1}} \dots \xrightarrow{\overline{T_i}} q_i \xrightarrow{\overline{T}_{i+1}} \dots \xrightarrow{\overline{T_n}} q$, we refer to the first i transitions sets of the sequence with the term sub-computation, written $c_{[0\dots i]}$.

Definition 9. *Given a strategy S, a play of S is a computation $c = q_0 \xrightarrow{\overline{T_1}} \dots \xrightarrow{\overline{T_n}} q$, such that for all sub-computations $c_{[0\dots i]}$, $S(c_{[0\dots i]}) \in \overline{T}_{i+1}$.*

Let $Games(S)$ be the set of all the *final* computations in $\mathbb{C}$ which are also plays of S.

Definition 10. *Given a vector bound $\mathbb{EI} \in \mathbb{N}^k$, a strategy S is said to be winning for $\mathbb{EI}$ if and only if* $\sum_{c \in Games(S)} p(c) I(c) \leq \mathbb{EI}$.



(a) (b)

Figure 4: An example of a 2-unraveling of a loop region.

Finally, we highlight the problem we aim to resolve throughout this work.

Problem 1. *Given a structured acyclic* SPIN *and an expected vector bound* $\mathbb{EI}$ *decide whether or not there exists a winning strategy* S *for* $\mathbb{EI}$ *in* SPIN.

Given a generic state p, e.g., $q = \{p_1 \mapsto 0, p_2 \mapsto 0\}$ (for the sake of brevity, because the other positions are equal to ϵ are not inserted in q) from the diagram in Figure 3. Then, we are interested in the *MNCE* set of transitions and suppose we are in state q. In this case, all the *MNCE* are $\{t_1,t_3\},\{t_1,t_4\},\{t_2,t_3\},\{t_2,t_4\}$. Now, consider a transition set $\overline{T}$ where $\overline{T} = \{t_1\}$; it is clear that $\overline{T}$ is not *MNCE* because it is not maximal as it does not consider a transition that originates from p_2, e.g. can be extended to $\{t_1,t_2\}$. Let's suppose now that we have $\overline{T} = \{t_1,t_4,t_5\}$. In this case, it is not *MNCE* because it contains t_5 that is not enabled. Finally, let's suppose that we have $\overline{T} = \{t_1,t_2,t_4\}$; it is not *MNCE* because it contains t_1,t_2 that have the same origin in p_1. In fact, $incoming(t_1) = incoming(t_2) = p_1$. We now propose an example to clarify what we mean by $Pvariant(\overline{T})$ 11. Consider the *MNCE* $\{t_1,t_3\}$, the $Pvariant$ is $Pvariant(\{t_1,t_3\}) = Pvariant(\{t_1,t_4\}) = \{\{t_1,t_3\},\{t_1,t_4\}\}$. Notice that one variant of $\{t_1,t_3\}$ is $\{t_1,t_3\}$, because one variant of $\overline{T}$ is always $\overline{T}$.

3.3 Dealing with Loops

Despite the whole work being based on acyclic SESE diagrams, we are aware that an important component of such diagrams is missing, i.e., loops. We briefly discuss a simple method for handling loops in our framework, and we are interested in exploring more elegant and theoretical options in future developments. In our framework, the split node v that induces a loop must be a nature one, i.e., $v \notin V_{nature}$, to represent a more general problem. If $v \in V_{choice}$, we restrict our search to strategies that avoid further loop iterations due to the non-negative nature of impacts. We highlight a set $V_{loop} \subseteq V_{nature}$, identifying split nodes encapsulating a loop region. We introduce a function $maxloop : V_{loop} \to \mathbb{N}$ to encode the maximum loop iterations, allowing us to unravel the cyclic structure into an acyclic one. An example of this unraveling is provided in Figure 4, where $maxloop(v) = 2$ results in a chain of 2 copies of v nested into each other. Each additional iteration reduces the contribution to the expected impact by an order of magnitude. This approach is simple to understand and implement and can be parametrized by the user. However, it may result in an exponential increase in size for multiple nested loops, even if $max(Img(maxloop))$ is small. This could affect the feasibility of finding a winning strategy.

We would like to point out that the finite user-parametrized loop unraveling is one of the simplest and most common approaches adopted in the BPMN field [12] in order to deal with loops. For the time being, our tool (see the end of Section 5.2 for further) deals with loops by the method described above, which is still good for contexts that do not put to much emphasis on high numbers of iterations of the loops, for

quick experiments, or for comparison with more sophisticated methods to come.

4 Computational Complexity

In this section, we provide a complexity upper bound for Problem 1, that is PSPACE, by means of the Algorithm 2. The lower bound for the complexity, which is within NP-HARD and PSPACE (NP-HARD lower bound may be provided by a reduction similar to the one presented in Section 5 for k cost game) is still an open problem. First, we have to observe that due to the duration constraints, we may have an exponential number of wait steps if we express such durations in binary. However, this may be easily dealt with if we consider the fact that chains of wait transition by their very definition do not generate possible branching in the computation. Let Q the set of all possible states, we define a function $sat : Q \to Q$ as follows: $sat(q) = \begin{cases} q & \text{if q is saturated} \\ sat(q') & \text{with } q \xrightarrow{t_w} q' \text{otherwise} \end{cases}$.

Basically, the *sat* function take a state q and returns the next saturated state that can be obtained by q. Now we can provide the definition of saturating transition between two saturated states q, q':

$$q \overset{\overline{T}}{\Rightarrow} q' \quad \textbf{iff} \quad \begin{array}{l} q \text{ is saturated, } \overline{T} \text{ is an } \textit{MNCE} \text{ for } q \text{ and either } q \xrightarrow{\overline{T}} q' \text{ with } q' \text{ saturated} \\ \text{or there exists } q'' \text{ such that } q \xrightarrow{\overline{T}} q' \text{ and } sat(q'') = q' \end{array}$$

It is easy to see that a partial strategy S that is defined only on the computations c which end in a saturated state q is as good as a complete strategy since there is only one "move" allowed in a not-saturated state. The decision algorithm for Problem 1 makes use of Algorithm 1, that given a state q computes $sat(q)$ in logarithmic space by means of binary arithmetic.

Algorithm 1: Saturate(q, N)

Input: a state q of a SPIN $N = (PT = P \cup T, T_p, \Delta, I, Pr, D)$
Output: $sat(q)$

1 **if** *there exists* $t \in T$ *s.t.* t *is enabled in* q **then**
2 **return** q
3 **let** $\overline{T} \subseteq T$ s.t. for each $t \in \overline{T}$ and for each $p \in \textit{incoming}(t)$ we have $q(p) \neq \epsilon$
4 **foreach** $t \in \overline{T}$ **do**
5 $k_t \leftarrow \max\{D(p) - q(p) : p \in \textit{incoming}(t)\}$
6 $k \leftarrow n\min\{k_t : t \in \overline{T}\}$
7 **let** q' s.t. $\forall p \in P$

$$q'(p) = \begin{cases} \epsilon & \text{if } q(p) = \epsilon \\ q(p) + k & \text{otherwise} \end{cases}$$

8 **return** q'

Our decision procedure relies on the following notion of variant for and *MNCE* .

Definition 11. *Given an MNCE* $\overline{T}$ *in* q *an MNCE* $\hat{T}$ *in* q *is a* probabilistic variant *of* $\overline{T}$ *if the following conditions hold: 1.* $\overline{T} \cap (T \setminus T_p) = \hat{T} \cap (T \setminus T_p)$*; 2.* $\forall t \in T_p$ *s.t.* $t, sw(t) \notin \overline{T}$ *we have* $t, sw(t) \notin \hat{T}$*; 3.* $\forall t \in (\hat{T} \cap T_p)$ *either* $t \in \hat{T}$ *or* $sw(t) \in \hat{T}$*.*

Informally speaking, a probabilistic variant for an *MNCE* $\overline{T}$ in q is still an *MNCE* $\hat{T}$ in q which shares with $\overline{T}$ all the non-probabilistic transitions. Given an *MNCE* $\overline{T}$ in q, we denote with $Pvariant(\overline{T}, q)$ the set of all and only the probabilistic variants of $\overline{T}$ in q. Clearly, we have $\overline{T} \in Pvariant(\overline{T}, q)$.

Algorithm 2 employs a non-deterministic approach to ascertain the existence of a viable strategy for a given instance of Problem 1. This is achieved by dynamically enumerating all possible plays, thereby maintaining only a single play in memory at any given moment. This method ensures polynomial memory utilization while providing a comprehensive evaluation of potential strategies.

For the sake of brevity, we do not provide the full proof that Algorithm 2 works in polynomial space. However, we informally provide the key arguments of the proof:

- Algorithm 2 is non-deterministic because it guesses the correct move (if any) at line 9, where $\overline{T} \cap (T \setminus T_p)$ represents the output of the current strategy;
- $Saturate$ operates in LOGSPACE and deals with the binary representation of durations for places;
- Given that N is acyclic, we have that any transition is considered at most for one recursive call to $StrategyExists$. Therefore, the number of nested procedure calls is bounded by $|T|$ since t_w transitions are collapsed via the function $Saturate$;
- In principle, $|Pvariant(\overline{T}, q)|$ (line 10 of Algorithm 2) may be of the order of $2^{|T|}$. However, since only one element $\hat{T} \in Pvariant(\overline{T}, q)$ is needed at a time for updating $\overline{rei}$ via the recursive call in the body of the for loop (line 12 of Algorithm 2), it is possible to set up an enumeration to keep the space polynomial at each step.

Since each play may be represented in polynomial space, we have the following result.

Theorem 1. *Problem 1 is NP-HARD and belongs to the complexity class PSPACE.*

However, our primary objective is to formulate a strategy rather than merely verifying its existence. Consequently, Section 5 is dedicated to addressing the strategy synthesis problem for BPMN+CPI. This section elaborates on the proposed solution, central to the functionality of the effective prototype that we have developed and implemented.
The exact complexity of Problem 1 is still open, we know that it can be proved to be NP-HARD by means of a reduction from the Partition problem introduced in Section 5 for k-cost reachability games.

The NP-HARD lower bound may be achieved by building a game devoid of nature nodes in a way that resembles the one-player restriction of the generalized game proposed in [16], but here Partition is used instead of SAT as the NP-HARD problem we reduce from. In [16], the authors provide a QSAT reduction for the unrestricted case, thus obtaining a PSPACE-HARD lower bound. Such a reduction is not directly applicable in our setting since our winning conditions embrace all possible plays, not a single one. In other words, in [16], a faulty strategy may be detected by witnessing a faulty single play it generates, while in our setting, a faulty strategy may be detected only by considering a subset (possibly all) the plays it generates. For this reason, at this point, we cannot conjecture the exact lower bound for the complexity of Problem 1 without further analysis.

5 Synthesizing Strategies

In this section, we will take advantage of the SPIN translation which has been fully described in Section 3.2. This tree has the foundational semantics of classical Petri Nets [25] for BPMN process. These concepts serve as the mathematical and logical basis for describing a graph-game representation and how the strategy is discovered presented below.

5.1 A k-cost Reachability Game

In this section, we will introduce a graph-game representation for dealing with the synthesis of strategies given a BPMN+CPI diagram $\mathcal{D} = (V, E, E_\top, \mathcal{T})$ which decides whether there exists a strategy that guarantees that the expected impact of a diagram is dominated by a given impact vector bound $\mathbb{I}$.

Definition 12. *A k-cost game board is a tuple $\mathcal{B} = (P = P_\circ \cup P_\square, p_0, F, C, M)$ such that $p_0 \in P$, $M \subseteq P \times P$, $F \subseteq P$ with $\{(m, m') : m \in F\} = \emptyset$ (i.e, there aren't outgoing edges from F), $C : P \to \mathbb{R}^k$, (P, M) is a directed acyclic graph.*

Definition 13. *Given k-cost game board $\mathcal{B} = (P = P_\circ \cup P_\square, p_0, F, C, M)$ a strategy is a function $s : P^* \to P$ such that: for every $\rho \in P^*$ we have $(\rho[-1], s(\rho)) \in M$.*

Algorithm 2: Recursive Procedure for Solving Problem 1

Input: a SPIN $N = (P, PT = T \cup T_p, \Delta, I, Pr, D)$ and $\mathbb{EI} \in \mathbb{N}^k$
Output: $ei \in \mathbb{R}^k$ with $ei \leq \mathbb{EI}$ if there exists a strategy with residual expected impact ei, and FAIL otherwise

let q_0 be the initial state of N;
return `StrategyExists(Saturate` $(q_0, N), 0^k, 1, \mathbb{EI})$

Procedure `StrategyExists(`*q, im, cp, rei*`)`**:**
  Data: A saturated state q of N, the value cp of the cumulative probability of the current play, $im \in \mathbb{R}^k$ the current impact for the play, $rei \in \mathbb{R}^k$ the residual expected impact currently available for consumption.
  Result: $rei \in (\mathbb{R}^+)^k$ if there exists a strategy from the current state q that that has rei residual w.r.t. ei, and FAIL otherwise
  if *q is final* **then**
    if $rei \not\leq 0^k$ **then**
      FAIL
    return $rei - (cp \cdot im)$
  let $\overline{T}$ an *MNCE* for q'
  $\overline{rei} \leftarrow rei$
  foreach $\hat{T} \in Pvariant(\overline{T}, q)$ **do**
    let q' s.t. $q \xrightarrow{\hat{T}} \hat{q}'$
    $\overline{rei} \leftarrow StrategyExist($`Saturate(q', N)`$, im + \sum_{t \in \hat{T}} I(t), cp \cdot \prod_{t \in \hat{T} \cap T_p} Pr(t), \overline{rei})$
    if $rei \not\geq 0^k$ **then**
      FAIL
  return $\overline{rei}$

Definition 14. *Given a k-cost game board $\mathcal{B} = (P = P_\circ \cup P_\square, p_0, F, C, M)$ and a strategy s, a successful play $\rho \in P^*$ is generated by s in $\mathcal{B}$ if and only if: (i) $\rho[0] = p_0$; (ii) $\rho[-1] \in F$; (iii) for every $0 < i < |\rho|$ if $\rho[i-1] \in P_\circ$ then $\rho[i] = s(\rho[0 : i])$.*

Let P_s^* be the set of all the possible plays generated by s.

Definition 15. *Given s we say that P_s^* is closed if for each $\rho \in P_s^*$ and for each $0 \leq i < |\rho| - 1$ such that $\rho[i] \in P_\square$ then for each $(\rho[i], p) \in M$ we have that there exists $\rho' \in P_s^*$ with $\rho'[0 : i] = \rho[0 : i]$ and $\rho'[i+1] = p$.*

Given a P_s^* we let $final(P_s^*)$ the set $final(P_s^*) = \{\rho[-1] : \rho \in P_s^*\}$.

Problem 2. *Given a k-cost game board $\mathcal{B} = (P = P_\circ \cup P_\square, p_0, F, C, M)$ and a cost $c \in \mathbb{R}^k$ determine whether or not there exists a strategy s for which P_s^* is closed and $\sum_{p \in final(P_s^*)} C(p) \leq c$.*

A strategy s is positional if and only if for every $\rho, \rho' \in P^*$ we have that $\rho[-1] = \rho'[-1]$ implies $s(\rho) = s(\rho')$. For the purpose of our game, w.l.o.g. a positional strategy may be redefined as $s : P_\circ \to P$.

Problem 3. *Given a k-cost game board $\mathcal{B} = (P = P_\circ \cup P_\square, p_0, F, C, M)$ and a cost $c \in \mathbb{R}^k$ determine whether or not there exists a **positional** strategy s for which P_s^* is closed and $\sum_{p \in final(P_s^*)} C(p) \leq c$.*

Theorem 2. *For every k-cost game board and each cost vector $c \in \mathbb{R}^k$ we have that $(\mathcal{B}, c)$ is a positive instance of Problem 2 if and only if $(\mathcal{B}, c)$ is a positive instance of Problem 3*

It is easy to prove that Problem 3 belongs to the complexity class NP, by simply provide a succinct certificate, that is, given an instance $(\mathcal{B} = (P = P_\circ \cup P_\square, M, p_0, F, C), c)$ of Problem 3 guess a subset $M' \subseteq M$ such that $\{(p, p') \in M : p \in P_\square\} \subseteq M'$ and for each $p \in P_\circ$ either $\{(p, p') \in M\} = \emptyset$ or there exists a unique edge $(p, p') \in M'$. Then, let F' be the subset of F reachable from p_0 in the M'-induced sub-graph $(P_\circ \cup P_\square, M')$ we have that M' is a solution if and only if $\sum_{p \in F'} C(p) \leq c$. The NP-HARD lower bound for Problem 3, and thus for Problem 2, is proved by a reduction from the following NP-HARD problem.

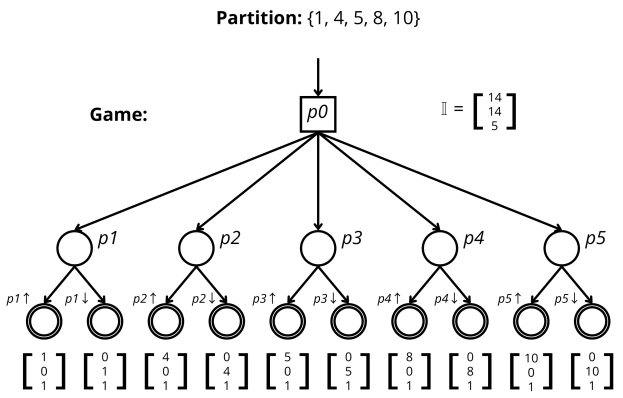


Figure 5: Reduction from Partition to k-cost game Problem.

Problem 4. (Distinct Partition) *Given a set of natural numbers $S = \{n_1, \ldots, n_m\}$ decide whether or not there exists a partition (S_1, S_2) of S such that $\sum_{n \in S_1} n = \sum_{n \in S_2} n$.*

As formulated by Korf in [21], Problem 4 is actually NP-complete. We recall this in Theorem 3.

Theorem 3. *Distinct Partition (Problem 4) is NP-Complete [21].*

There exists a simple LOG-SPACE reduction from Distinct Partition to Problem 3, and thus to Problem 2, for $k \geq 3$. The reduction is very simple, it suffices to transform the distinct partition problem $S = \{n_1, \ldots, n_m\}$ into an instance of Problem 2 $(\mathcal{B}_S = (P = P_\circ \cup P_\square, M, p_0, F, C), c_S)$ as follows:

1. $P_\circ = \{p^i, p^i_\uparrow, p^i_\downarrow : 1 \leq i \leq m\}$,
2. $P_\square = \{p_0\}$,
3. $M = \{(p_0, p^i) : 1 \leq i \leq m\} \cup \{(p^i, p^i_\uparrow), (p^i, p^i_\downarrow) : 1 \leq i \leq m\}$,
4. $F = \{p^i_\downarrow, p^i_\uparrow : 1 \leq i \leq m\}$,
5. $C(p^i_\uparrow) = [n_i, 0, 1]$ and $C(p^i_\downarrow) = [0, n_i, 1]$ for each $1 \leq i \leq m$,
6. $c_S = \left[\frac{\sum_1^m n_i}{2}, \frac{\sum_1^m n_i}{2}, m\right]$.

An example of the proposed reduction is given in Figure 5. It is easy to prove that $(\mathcal{B}_S, c_S)$ is a positive instance of Problem 2 if and only if S is a positive instance of the distinct partition problem.

Theorem 4. *Problem 3 and Problem 2 for $k \geq 3$ are NP-Complete problems.*

5.2 From BPMN+CPI to k-cost Reachability Game

We conclude this section by providing the direct translation from an instance $(N, \mathbb{I})$ of Problem 1 into a k-cost game $(\mathcal{B}, \mathbb{I})$, which admits a solution if and only if the problem $(N, \mathbb{I})$ admits a solution.

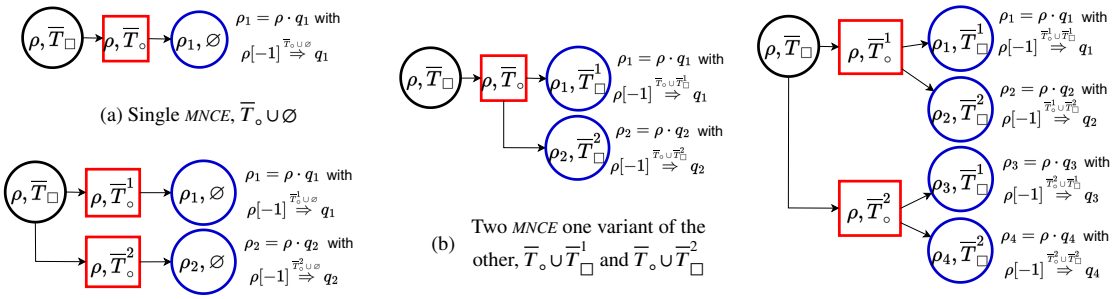


Figure 6: Different scenarios involving at most one choice and at least one probabilistic split.

Moreover, if $(\mathcal{B}, \mathbb{E}\mathbb{I})$ admits a solution, i.e., it is a positive instance of Problem 2, such a solution will effectively represent a strategy for the original problem.

Before providing this translation, we introduce a couple of useful definitions. Given an *MNCE* $\overline{T}$ for a state q, we define two sets: $\overline{T}_\square = \overline{T} \cap T_p$ and $\overline{T}_\circ = \overline{T} \setminus T_\square$. Additionally, for any $\overline{T}_\square \subseteq T_p$, let $Pr(\overline{T}_\square) = \prod_{t \in \overline{T}_\square} Pr(t)$; clearly, $Pr(\emptyset) = 1$. For any $\overline{T}_* \subseteq T$, let $\mathcal{I}(\overline{T}_*) = \sum_{t \in \overline{T}_*} \mathcal{I}(t)$; clearly, $\mathcal{I}(\emptyset) = 0$. Finally, let Q be the set of all possible saturated states on N and C^+_Q be the set of all possible non-empty combinations of elements in Q. Given a combination $\rho \in C^+_Q$, we denote its last element as $\rho[-1]$.
Given an instance $(N, \mathbb{E}\mathbb{I})$ of we define a k-cost game board $\mathcal{B}_N = (S = S_\circ \cup S_\square, s_0, F, C, M)$ as follows:

$$S_\circ = \{(\rho, \overline{T}_\square) \in C^+_Q \times 2^{T_p} : \rho[-1] \textit{ is saturated}\}, \quad S_\square = \left\{(\rho, \overline{T}_\circ) : \begin{array}{l} \rho \in C^+_Q, \text{ there exists } \overline{T}_\square \subseteq T_p s.t. \\ \overline{T}_\circ \cup \overline{T}_\square \textit{ is an MNCE for } \rho[-1] \end{array}\right\},$$

$$s_0 = (sat(q_0), \emptyset), \qquad F = \{(\rho, \overline{T}_\square) \in C^+_Q \times 2^{T_p} : \rho[-1] = q_f\},$$

$$\text{and } M = \{((\rho, \overline{T}_\square), (\rho, \overline{T}_\circ)) : (\rho, \overline{T}_\square) \in P_\circ, (\rho, P_\square) \in P_\square\} \cup \{((\rho, \overline{T}_\circ), (\rho q, \overline{T}_\square)) : \rho[-1] \overset{\overline{T}_\circ \cup \overline{T}_\square}{\Rightarrow} q\}.$$

Graphical examples of how the relation M is build in the case when the *MNCE* $\overline{T} = \overline{T}_\circ \cup \overline{T}_\square$ satisfies $|\overline{T}_\circ| \leq 1$ and $|\overline{T}_\square| \leq 1$ are provided in Figure 6.
Lastly, for the cost function, let M^* denote the reflexive and transitive closure of M. For any $s \in F$, the cost function $C(s)$ is defined as:

$$C(s) = \left(\prod_{(\rho, \overline{T}_\square) \in S_\circ : (s_0, (\rho, \overline{T}_\square)), ((\rho, \overline{T}_\square), s) \in M^*} Pr\left(\overline{T}_\square\right) \right) \cdot \left(\sum_{(\rho, \overline{T}_*) \in S_\circ \cup S_\square : (s_0, (\rho, \overline{T}_*)), ((\rho, \overline{T}_*), s) \in M^*} \mathcal{I}\left(\overline{T}_*\right) \right)$$

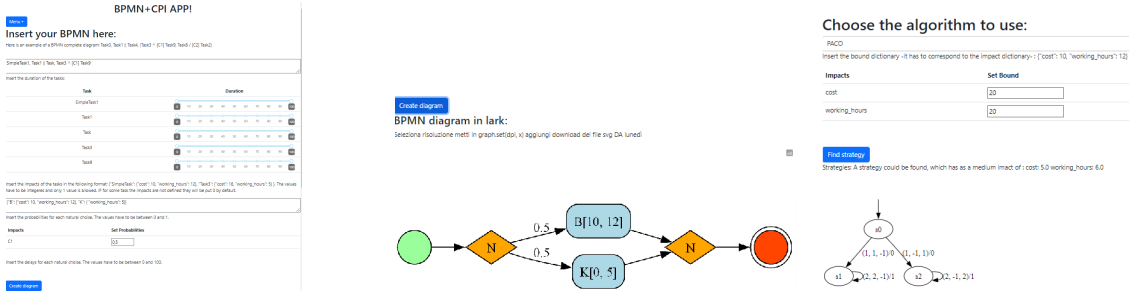
The formula described assigns to each final state $s \in F$ the contribution to the expected impact generated by paths terminating at s. Now, as a final measure, we resolve the k-cost game by selecting [1] a

[1] This is implemented by evaluating all possible subsets $F' \subseteq F$ such that $\sum_{s \in F'} C(p) \leq \mathbb{E}\mathbb{I}$ and for each $s' \in F \setminus F'$, $\sum_{s \in F' \cup \{s'\}} C(s) > \mathbb{E}\mathbb{I}$. We consider only the maximal admissible subsets of F, as they can "attract" the initial state if and only if at least one of their subsets does.

subset $F' \subseteq F$ such that the total expected impact satisfies: $\sum_{s \in F'} C(s) \leq \mathbb{EI}$

We employ the standard attractor procedure as described in [27], initiating with $Attr^0 = F'$ in $\mathcal{B}$. A positive outcome, along with the strategy formulated by the attractor procedure, is confirmed if there exists $k \in \mathbb{N}$ such that $s_0 \in Attr^k$. While the attractor procedure itself runs in polynomial time, approximately $\mathcal{O}(nm)$ for a graph with n nodes and m edges, the non-deterministic selection of a candidate $Attr^0$ from the set of final states remains computationally intensive, since the number of final states may be exponential in the size of SPIN thus the above procedure for synthetizing a strategy operates in NEXPTIME.

Implementation The algorithm described in this section, known as *PACO*, has been developed and is accessible at `https://github.com/ansimonetti/PACO`. *PACO* is designed as a Dash App [20]. The process is written in Lark syntax [1], with all choices, probabilities, and impacts clearly defined, as visible in Figure 7a and printed using Graphviz [3] and PyDot [7], as shown in Figure 7b. A specific section is dedicated to defining the expected impacts vector. Subsequently, the AALpy automata [24] is employed to provide a strategy, as previously described, if one exists. If one is found, the algorithm returns it together with the associated impact factors. Moreover, it prints the tree associated with the strategy, indicating which tasks have to be done to complete the process within the bound vector as shown in Figure 7c.



(a) (b) (c)

Figure 7: Example of using our Dash App: defining the BPMN in our Dah App 7a, print the BPMN using Lark 7b and example of founded strategy using *PACO* 7c

6 Conclusion

In this study, we developed a BPMN extension, denoted as BPMN+CPI, designed to handle execution in the presence of impacts, probabilistic splits, and choices. The semantics for this extension were formulated using an enriched version of Petri Nets, namely, SPIN. The primary objective of this work was to create a system capable of informing users about the existence of a strategy for a given process and user-defined thresholds. This involves determining whether there is a controller capable of executing each step of the process while ensuring that the expected value of each resource across repeated process instances remains within the predefined thresholds.

First, we proved that the associated decision problem, i.e., determining if such a controller exists, belongs to the complexity class PSPACE. Then, we provided an effective method for building the controller by modifying classical reachability games over graphs. Based on these theoretical results, we implemented a tool capable of determining the existence of a strategy given a BPMN+CPI process and a given

threshold $\mathbb{E}\mathbb{I}$. This tool is currently under development, but a working prototype is available online for the benefit of the community.

For future work, we envision two promising extensions. The first, theoretical, aims to deal with loops in the workflow in a non-approximated fashion and to propose alternative algorithms for solving the problem, potentially closing the complexity gap, which currently stands between PSPACE and NP. The second, more practical extension, focuses on better representing the obtained strategy by integrating it into the choice gateway of the BPMN+CPI, for instance, representing decisions with a set of inequalities involving intervals of values for the impact components observed in specific choice nodes.

Acknowledgments This work has been carried out while Emanuele Chini was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome in collaboration with the University of Verona.

References

[1] (2024): *Lark - Parsing Library & Toolkit*. Available at `https://github.com/lark-parser/lark`. Accessed: 2024-04-20.

[2] Parosh Aziz Abdulla, Mohamed Faouzi Atig, Piotr Hofman, Richard Mayr, K. Narayan Kumar & Patrick Totzke (2014): *Infinite-state energy games*. In Thomas A. Henzinger & Dale Miller, editors: *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, ACM, pp. 7:1–7:10, doi:10.1145/2603088.2603100.

[3] Sebastian Bank (2024): *Graphviz*. Available at `https://github.com/xflr6/graphviz`. Accessed: 2024-04-20.

[4] Gerd Behrmann, Agnes Cougnard, Alexandre David, Emmanuel Fleury, Kim G Larsen & Didier Lime (2007): *UPPAAL-Tiga: Time for Playing Games! (Tool Paper)*. In: *Computer Aided Verification: 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007. Proceedings 19*, Springer, pp. 121–125, doi:10.1007/978-3-540-73368-3_14.

[5] Cinzia Cappiello, Maria Grazia Fugini, GR Gangadharan, Alexandre Mello Ferreira, Barbara Pernici & Pierluigi Plebani (2010): *First-step toward energy-aware adaptive business processes*. In: *On the Move to Meaningful Internet Systems: OTM 2010 Workshops: Confederated International Workshops and Posters: International Workshops: AVYTAT, ADI, DATAVIEW, EI2N, ISDE, MONET, OnToContent, ORM, P2P-CDVE, SeDeS, SWWS and OTMA. Hersonissos, Crete, Greece, October 25-29, 2010. Proceedings*, Springer, pp. 6–7, doi:10.1007/978-3-642-16961-8_4.

[6] J. Carmona, J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno & A. Yakovlev (2008): *A Symbolic Algorithm for the Synthesis of Bounded Petri Nets*. In: *Proceedings of the 29th International Conference on Applications and Theory of Petri Nets*, PETRI NETS '08, Springer-Verlag, Berlin, Heidelberg, p. 92–111, doi:10.1007/978-3-540-68746-7_10.

[7] Ero Carrera (2024): *Pydot*. Available at `https://github.com/pydot/pydot`. Accessed: 2024-04-20.

[8] Krishnendu Chatterjee & Laurent Doyen (2011): *Energy and mean-payoff parity Markov decision processes*. In: *International Symposium on Mathematical Foundations of Computer Science*, Springer, pp. 206–218, doi:10.1007/978-3-642-22993-0_21.

[9] Carlo Combi, Barbara Oliboni & Francesca Zerbato (2019): *A modular approach to the specification and management of time duration constraints in BPMN*. *Information Systems* 84, pp. 111–144, doi:10.1016/j.is.2019.04.010.

[10] Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Marius Mikučionis & Jakob Haahr Taankvist (2015): *Uppaal Stratego*. In Christel Baier & Cesare Tinelli, editors: *Tools and Algorithms for the Construction and Analysis of Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 206–211, doi:10.1007/978-3-662-46681-0_16.

[11] Massimiliano De Leoni, Wil MP Van Der Aalst & Boudewijn F Van Dongen (2012): *Data-and resource-aware conformance checking of business processes*. In: *Business Information Systems: 15th International Conference, BIS 2012, Vilnius, Lithuania, May 21-23, 2012. Proceedings 15*, Springer, pp. 48–59, doi:10.1007/978-3-642-30359-3_5.

[12] Remco M Dijkman, Marlon Dumas & Chun Ouyang (2008): *Semantics and analysis of business process models in BPMN*. *Information and Software technology* 50(12), pp. 1281–1294, doi:10.1016/j.infsof.2008.02.006.

[13] Marlon Dumas, Luciano García-Bañuelos & Artem Polyvyanyy (2010): *Unraveling Unstructured Process Models*. In Jan Mendling, Matthias Weidlich & Mathias Weske, editors: *Business Process Modeling Notation - Second International Workshop, BPMN 2010, Potsdam, Germany, October 13-14, 2010. Proceedings*, *Lecture Notes in Business Information Processing* 67, Springer, pp. 1–7, doi:10.1007/978-3-642-16298-5_1.

[14] Francisco Durán, Camilo Rocha & Gwen Salaün (2018): *Stochastic analysis of BPMN with time in rewriting logic*. *Science of Computer Programming* 168, pp. 1–17, doi:10.1016/j.scico.2018.08.007.

[15] Jorge Fernandes, João Reis, Nuno Melão, Leonor Teixeira & Marlene Amorim (2021): *The role of Industry 4.0 and BPMN in the arise of condition-based and predictive maintenance: A case study in the automotive industry*. *Applied Sciences* 11(8), p. 3438, doi:10.3390/app11083438.

[16] Nathanaël Fijalkow & Florian Horn (2010): *The surprizing complexity of generalized reachability games*. *arXiv preprint arXiv:1010.2420*, doi:10.48550/arXiv.1010.2420.

[17] Jerzy Filar & Koos Vrieze (2012): *Competitive Markov decision processes*. Springer Science & Business Media, doi:10.1007/978-1-4612-4054-9.

[18] Christoph Haase, Stephan Kreutzer, Joël Ouaknine & James Worrell (2009): *Reachability in Succinct and Parametric One-Counter Automata*. pp. 369–383, doi:10.1007/978-3-642-04081-8_25.

[19] Luke Herbert & Robin Sharp (2013): *Precise quantitative analysis of probabilistic business process model and notation workflows*. *Journal of Computing and Information Science in Engineering* 13(1), p. 011007, doi:10.1115/1.4023362.

[20] Plotly Technologies Inc. (2024): *Dash*. Available at `https://dash.plotly.com/`. Accessed: 2024-04-20.

[21] Richard E. Korf (1998): *A complete anytime algorithm for number partitioning*. *Artificial Intelligence* 106(2), pp. 181–203, doi:10.1016/S0004-3702(98)00086-1.

[22] M. Kwiatkowska, G. Norman & D. Parker (2011): *PRISM 4.0: Verification of Probabilistic Real-time Systems*. In G. Gopalakrishnan & S. Qadeer, editors: *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, *LNCS* 6806, Springer, pp. 585–591, doi:10.1007/978-3-642-22110-1_47.

[23] Matteo Magnani & Danilo Montesi (2007): *BPMN: How Much Does It Cost? An Incremental Approach*. In Gustavo Alonso, Peter Dadam & Michael Rosemann, editors: *Business Process Management*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 80–87, doi:10.1007/978-3-540-75183-0_6.

[24] Edi Muškardin, Bernhard Aichernig, Ingo Pill, Andrea Pferscher & Martin Tappler (2022): *AALpy: an active automata learning library*. *Innovations in Systems and Software Engineering* 18, pp. 1–10, doi:10.1007/s11334-022-00449-3.

[25] James L Peterson (1977): *Petri nets*. *ACM Computing Surveys (CSUR)* 9(3), pp. 223–252, doi:10.1145/356698.356702.

[26] Luise Pufahl, Francesca Zerbato, Barbara Weber & Ingo Weber (2022): *BPMN in healthcare: Challenges and best practices*. *Information Systems* 107, p. 102013, doi:10.1016/j.is.2022.102013.

[27] Wolfgang Thomas (1995): *On the synthesis of strategies in infinite games*. In: *Annual Symposium on Theoretical Aspects of Computer Science*, Springer, pp. 1–13, doi:10.1007/3-540-59042-0_57.

[28] Uri Zwick & Mike Paterson (1996): *The complexity of mean payoff games on graphs*. *Theoretical Computer Science* 158(1), pp. 343–359, doi:10.1016/0304-3975(95)00188-3.