Optimal Reachability in Cost Time Petri Nets

Hanifa Boucheneb[1], Didier Lime[2], Baptiste Parquier[2], Olivier H. Roux[2] and Charlotte Seidner[3]

[1] École Polytechnique de Montréal, Québec, Canada
[2] École Centrale de Nantes, LS2N UMR CNRS 6004
[3] Université de Nantes, LS2N UMR CNRS 6004

Abstract. In order to model resource-consumption or allocation problems in concurrent real-time systems, we propose an extension of time Petri nets (TPN) with a linear cost function and investigate the minimum/infimum cost reachability problem. We build on the good properties of the state class symbolic abstraction, which is coarse and requires no approximation (or k-extrapolation) to ensure finiteness, and extend this abstraction to symbolically compute the cost of a given sequence of transitions. We show how this can be done, both by using general convex polyhedra, but also using the more efficient Difference Bound Matrix (DBM) data structure. Both techniques can then be used to obtain a symbolic algorithm for minimum cost reachability in bounded time Petri nets with possibly negative costs (provided there are no negative cost cycles). We prove that this algorithm terminates in both cases by proving that it explores only a finite number of extended state classes for bounded TPN, without having to resort to a bounded clock hypothesis, or to an extra approximation/extrapolation operator. All this is implemented in our tool Romeo and we illustrate the usefulness of these results in a case study.

1 Introduction

Time Petri nets (TPN for short) have been introduced by Merlin in 1974 to extend the modelling and analysis powers of Petri nets to time dependent systems. They allow to specify different kinds of time constraints by means of intervals associated with transitions. Furthermore, they offer effective reachability analysis methods that take into account the time constraints of systems. These methods are generally based on the state space abstraction where all the firing sequences and reachable markings are represented. Even if the reachability problem is not decidable for TPN, there are some subclasses of TPN, such as bounded TPN, for which the reachability problem is decidable. Using reachability analysis methods, tools such as Tina and Romeo provide an interesting platform to verify various qualitative and quantitative properties of TPN.

Cost time Petri nets (cTPN for short) extend TPN with costs associated with transitions and markings. The cost of a transition represents its firing cost while the cost of a marking is the price per time unit for staying in the marking. As a run in the TPN is a succession of discrete transitions interspersed with time elapsing (delay transitions), the cost of the run is the accumulation of the costs of its discrete and delay transitions. Several runs may lead with different costs to a goal marking. These costs represent in general resource-consumptions such as memory and power consumptions. In such cases, it would be interesting to be able to determine the runs that yield the optimal cost. This problem, called the optimal-cost reachability, can be stated formally as follows in the context of cTPN: Given a goal marking m, what is the optimal (minimal/infimum) cost to reach m in the cTPN?

This paper deals with the optimal-cost problem for cTPN. It proposes a forward exploration of cost state classes that provides the optimal cost to reach a given goal marking for all bounded cTPN with no negative-cost cycles.

Related works

In the literature, the optimal-cost problem has been addressed for Priced Timed automata (PTA) in [2,3,4,9,12] and Priced Timed Petri nets (PTPN) in [1]. A PTA is a timed automaton where locations have *rate costs* and edges have *costs*. The rate cost of a location gives the cost per time unit for staying in the location, whereas the cost of a transition indicates its firing cost. A PTPN is a timed arc Petri net where each place has a rate cost, each transition has a firing cost and the firing semantics of its transitions is weak. It is well known that many verification problems such as reachability and coverability are undecidable under the strong semantics but decidable under the weak semantics. However, timed models based on strong semantics are more appropriate to specify urgency than those based on weak semantics. Moreover, they do not need to manage dead tokens or transitions.

For PTA with non negative integer costs, two different solutions based on priced regions and priced zones have been proposed, in [2] and [12], respectively for the optimal-cost problem. The solution proposed in [2] has allowed the authors to prove decidability of the optimal-cost problem. However, from a practical point of view, region graphs are less useful than zone graphs. In [3,4,9,12], the computation of the optimal-cost to reach a goal location is based on a forward exploration of priced zones, where an extra variable Cost gives the currently best known cost of reaching the goal location. A priced zone extends a zone with a linear cost function specifying the optimal cost to reach every state of that zone [3,4,9,12]. The optimal cost of a priced zone is obtained by minimising its cost function under the constraints of the zone. The priced zones of the discrete and continuous successors are computed by considering some zone facets[4]. The exploration is performed congruently with a "bigger and cheaper" inclusion relation over priced zones. The inclusion relation used in [4,12] and implemented in the UPPAAL-CORA tool ensures termination of the exploration for all *bounded* PTA (meaning all clocks are bounded). However, the terminaison is not guaranteed for PTA with unbounded clocks. Furthermore, if negative

[4] A zone facet is obtained by adding a constraint of the form $x = c$ (or $x - y = c$), where c is a constant and $x \prec c$ (or $x - y \prec c$) is an atomic constraint of the zone.

costs are allowed, the exploration does not necessarily provide the optimal-cost to reach the goal location. Indeed, if a path leading to the goal location goes through a priced zone belonging to some cycle with negative cost, the optimal cost to reach the goal location may be $-\infty$. In [9], the authors have improved the approach, developed in [3,4,12], by refining and combining the inclusion relation over priced zones with an over-approximation relation over clock valuations, by ignoring clock values that exceed some bound when priced zones are compared together. This improvement ensures termination of the forward exploration algorithm even when clocks are not bounded and costs are negative, provided that the PTA is free of negative cost cycles.

For PTPN, the optimal-cost reachability problem is also decidable, but only if all costs are non negative integers. The computation of the optimal-cost for reaching a goal marking is based on similar techniques to those of PTA [1].

Our contribution

While weighted, priced, or cost timed automata have been well-studied in the literature, very few comparable results exist for time Petri nets. Yet, beyond subjective preferences for one or another formalism, time Petri nets exhibit some interesting properties. In particular, while the symbolic techniques defined for timed automata can be adapted to TPN, the symbolic abstraction of choice remains the so-called state classes. They are naturally very coarse and have the great advantage of not requiring any further approximation (as in k-extrapolation, LU-extrapolation, etc.), or any boundedness hypothesis on the clock variables, to ensure their finite number. This has proven quite problematic and the restriction that clocks should be bounded to ensure termination has has been lifted only recently in [9], 15 years after the the initial approach of [12].

We therefore investigate here how this specific abstraction can be adapted to symbolically compute optimal costs. As we expected, the state class abstraction, even extended with costs, does not require any approximation. While the results we obtain are similar, in terms of what we can do in the end, to the results obtained for timed automata, the underlying techniques are quite specific. For instance, an important result is that we can partition the domains (encoded as Difference Bound Matrices or DBMs) to ensure that the constraints on the cost remain simple as in [12] but the notion of facet used in that paper does not apply to our model.

Outline The paper is structured as follows. Section 2 presents the TPN formalism, its extension with costs (cTPN) and their semantics. Section 3 extends the state class method to cTPN. In Section 4, we present the symbolic algorithm used to compute the optimal cost. In Section 5, by describing how cost state classes can be partitioned, we both improve the algorithm efficiency and provide a key result for the termination proof of the symbolic algorithm, given in Section 6. In Section 7, we present our implementation in the Romeo tool and a case study to illustrate how cTPN can be useful for the design of real-time systems. Finally, Section 8 concludes this paper.

2 Cost Time Petri Nets

2.1 Preliminaries

We denote the set of natural numbers by $\mathbb{N}$, the set of integers by $\mathbb{Z}$, the set of rational numbers by $\mathbb{Q}$, the set of real numbers by $\mathbb{R}$ and the set of non-negative real numbers by $\mathbb{R}_{\geq 0}$.

For $I \in \mathcal{I}_{\mathbb{Q}_{\geq 0}}$, $\underline{I}$ denotes its left end-point and $\overline{I}$ denotes its right end-point if I is bounded and ∞ otherwise. Moreover, for any $\theta \in \mathbb{R}_{\geq 0}$, we let $I \dot{\dot{-}} \theta$ be the interval defined by $\{x - \theta \mid x \in I \wedge x - \theta \geq 0\}$.

Let F and F' be two systems of linear inequalities over a set of variables X; $F \equiv F'$ denotes that both systems have the same set of solutions over X. Furthermore, $F_{|Y}$ (with $Y \subseteq X$) denotes the projection of F over Y obtained for instance by a Fourier–Motzkin elimination of all variables that are in X but not in Y.

2.2 Time Petri Nets

Definition 1 (Time Petri Net (TPN)). *A* Time Petri Net *is a sextuple* $\mathcal{N} = (P, T, {}^{\bullet}., .^{\bullet}, m_0, I_s)$ *where:*

- P *is a finite non-empty set of places,*
- T *is a finite set of transitions such that* $T \cap P = \emptyset$,
- ${}^{\bullet}. : T \rightarrow \mathbb{N}^P$ *is the backward incidence mapping,*
- $.^{\bullet} : T \rightarrow \mathbb{N}^P$ *is the forward incidence mapping,*
- $m_0 : P \rightarrow \mathbb{N}$ *is the initial marking,*
- $I_s : T \rightarrow \mathcal{I}_{\mathbb{Q}_{\geq 0}}$ *is a function assigning a firing interval to each transition.*

The distribution of tokens over the places of $\mathcal{N}$ is called a marking which is a mapping from P to $\mathbb{N}$. For a marking $m \in \mathbb{N}^P$, $m(p)$ denotes the number of tokens in place p. A Petri net $\mathcal{N}$ is said to be k-bounded or simply bounded if the number of tokens in each place does not exceed a finite number k for any marking reachable from m_0.

A transition $t \in T$ is said to be *enabled* by a given marking $m \in \mathbb{N}^P$ if m supplies t with at least as many tokens as required by the backward incidence mapping ${}^{\bullet}$. We define $En(m)$ as the set of transitions that are enabled by the marking m:

$$En(m) = \{t \in T \mid m \geq {}^{\bullet}(t)\}$$

A transition $t' \in T$ is said to be *newly enabled* by the firing of a transition t from a given marking $m \in \mathbb{N}^P$ if it is enabled by $m - {}^{\bullet}t + t^{\bullet}$ but not by $m - {}^{\bullet}t$. The set of transitions that are newly enabled by the firing of t from the marking m is:

$$\mathcal{N}ewlyEn(m,t) = \left\{t' \in En(m - {}^{\bullet}t + t^{\bullet}) \mid t' \notin En(m - {}^{\bullet}t) \text{ or } t = t'\right\}$$

Definition 2 (State). *A* state *of the net $\mathcal{N}$ is described by an ordered pair (m, I) in $\mathbb{N}^P \times \mathcal{I}^T_{\mathbb{Q}_{\geq 0}}$, where m is a marking of $\mathcal{N}$ and I is a function called the interval function. $I : T \to \mathcal{I}_{\mathbb{Q}_{\geq 0}}$ associates a temporal interval with every transition enabled by m.*

Definition 3 (Semantics of a TPN). *The semantics of a TPN is defined by a timed transition system $(Q, q_0, \to)$ where:*

- $Q \subseteq \mathbb{N}^P \times \mathcal{I}^T_{\mathbb{Q}_{\geq 0}}$
- $q_0 = (m_0, I_0)$ *s.t.* $\forall t \in En(m_0) \quad I_0(t) = I_s(t)$
- $\to$ *consists of two types of transitions:*
  - *discrete transitions:* $(m, I) \xrightarrow{t} (m', I')$ *iff*
    - $m \geq {}^{\bullet}t$, $m' = m - {}^{\bullet}t + t^{\bullet}$ *and* $\underline{I(t)} = 0$,
    - $\forall t' \in En(m')$
      - $I'(t') = I_s(t')$ *if* $t' \in \mathcal{N}ewlyEn(m, t)$,
      - $I'(t') = I(t')$ *otherwise*
  - *time transitions:* $(m, I) \xrightarrow{\theta \in \mathbb{Q}_{\geq 0}} (m, I \ddot{-} \theta)$ *iff* $\forall t \in En(m)$, $\overline{(I \ddot{-} \theta)(t)} \geq 0$.

A run of a time Petri Net $\mathcal{N}$ is a (finite or infinite) path starting in state q_0 and whose steps follow the semantics described above. The set of runs of a TPN $\mathcal{N}$ is denoted by $\mathsf{Runs}(\mathcal{N})$. A run is therefore a succession of time and discrete transitions; let us for instance consider the elapsing of a duration θ followed by the firing of a transition t: $(m, I) \xrightarrow{\theta} (m, I \ddot{-} \theta) \xrightarrow{t} (m', I')$. In the following, such a succession is denoted by $(m, I) \xrightarrow{t@\theta} (m', I')$.

Furthermore, *sequence*(ρ) denotes the projection of the run ρ over T. The sequence σ corresponding to the run $\rho = q_0 \xrightarrow{t_0@\theta_0} q_1 \xrightarrow{t_1@\theta_1} q_2 \xrightarrow{t_2@\theta_2} q_3$ is therefore $\sigma = sequence(\rho) = t_0 t_1 t_2$.

Definition 4 (Discrete state graph of a TPN). *The discrete state graph (DSG) of a TPN is the structure $DSG = (S, s_0, \hookrightarrow)$ where $S \in \mathbb{N}^P \times \mathcal{I}^T_{\mathbb{Q}_{\geq 0}}$, $s_0 = (m_0, I_s)$ and $s \overset{t}{\hookrightarrow} s'$ iff $\exists \theta \in \mathbb{Q}_{\geq 0} \mid s \xrightarrow{t@\theta} s'$*

Any state of the DSG is a state of the semantics of the TPN and any state of the semantics which is not in the DSG is reachable from some state of the DSG by a continuous transition. The DSG is a dense graph and a state may have infinite number of successors by $\overset{t}{\hookrightarrow}$. Finitely representing state spaces involves grouping some sets of states.

State Classes For an arbitrary sequence of transitions $\sigma = t_1 \ldots t_n \in T^*$, let C_σ be the set of all states that can be reached by the sequence σ from s_0: $C_\sigma = \{s \in S | s_0 \overset{t_1}{\hookrightarrow} s_1 \cdots \overset{t_n}{\hookrightarrow} s\}$. All the states of C_σ share the same marking and can therefore be written as a pair (m, D) where m is the common marking and D is the union of all points belonging to the set of firing intervals. D is called the *firing domain*.

$\cong$ denotes the relation satisfied by two such sets of states when they have both the same marking and the same firing domain.

Definition 5. *Let $C_\sigma = (m, D)$ and $C'_{\sigma'} = (m', D')$ be two sets of states; $C_\sigma \cong C_{\sigma'}$ iff $m = m'$ and $D \equiv D'$.*

If $C_\sigma \cong C_{\sigma'}$, any firing schedule firable from some state in C_σ is firable from state in $C_{\sigma'}$ and conversely. The state classes as defined in [6,5] are the equivalence classes of the $\cong$ relation defined on the set of classes C_σ.

Definition 6. *The state class graph (SCG) of [6,5] is defined by the set of state classes equipped with a transition relation: $C_\sigma \xrightarrow{t} X$ iff $C_{\sigma.t} \cong X$.*

Hence the SCG computes the smallest set C of state classes w.r.t. $\cong$. The SCG is finite iff the net is bounded. Moreover, the SCG is a complete and sound state space abstraction of the TPN.

Given a state class $C = (m, D)$, a point $x = (\theta_1, \theta_2, ..., \theta_n) \in D$ is composed of the values of variables $\theta_1, \theta_2, ..., \theta_n$ that refers to the firing instants in C of transitions $t_1, t_2 ... t_n$ that are enabled by m. The firing domain may be described by linear inequations of the form $\theta_j - \theta_i \leq c$ or $\theta_i \leq c$ where $c \in \mathbb{Q}$; therefore, they can be encoded as a Difference Bound Matrix (DBM) [6,10].

Let $\Theta = \{\theta_1 ... \theta_n\}$ and $\mathcal{C}$ a set of constraints over Θ. Let θ_0 a reference variable whose value is always 0 and $\Theta_0 = \Theta \cup \{\theta_0\}$. A DBM M representing $\mathcal{C}$ is a matrix of size $|\Theta_0| \times |\Theta_0|$ such that $M_{ij} = inf\{c | (\theta_j - \theta_i \leq c) \in \mathcal{C}\}$ where $inf(\emptyset) = +\infty$. A DBM has a unique canonical form which gives the tightest bounds on all differences between variables.

2.3 Cost Time Petri Nets

Definition 7 (Cost Time Petri Net (cTPN)). *A* Cost Time Petri Net *is a tuple $\mathcal{N}_c = (P, T, {}^\bullet., .^\bullet, m_0, I_s, \omega, cr)$ where:*

- *$\mathcal{N} = (P, T, {}^\bullet., .^\bullet, m_0, I_s)$ is a TPN,*
- *$\omega : T \rightarrow \mathbb{Z}$ is the discrete cost function,*
- *$cr : \mathbb{N}^P \rightarrow \mathbb{Z}$ is the cost rate function; as a matter of fact, cr is a linear function over markings.*

Definition 8 (Semantics of a cTPN). *The semantics of a cTPN $\mathcal{N}_c = (P, T, {}^\bullet., .^\bullet, m_0, I_s, \omega, cr)$ is the semantics of the TPN $\mathcal{N} = (P, T, {}^\bullet., .^\bullet, m_0, I_s)$.*

The cost state of a cTPN is $(m, I, c) \in \mathbb{N}^P \times \mathcal{I}^T_{\mathbb{Q}_{\geq 0}} \times \mathbb{R}$, where (m, I) is a TPN state and c is the accumulation, from the initial state, of the costs of the discrete and timed transitions of a run that leads to (m, I). More specifically:

- the cost of a discrete transition $(m, I, c) \xrightarrow{t} (m', I', c')$ is $c' - c = \omega(t)$;
- the cost of a timed transition $(m, I, c) \xrightarrow{d} (m, I', c')$ is $c' - c = d * cr(m)$.

Definition 9 (Cost of a run (Ω_r)). *The cost of a run $\rho = (m_0, I_0, c_0) \xrightarrow{t_0 @ \theta_0} (m_1, I_1, c_1) \xrightarrow{t_1 @ \theta_1} (m_2, I_2, c_2) \cdots \xrightarrow{t_{n-1} @ \theta_{n-1}} (m_n, I_n, c_n)$ is*

$$\Omega_r(\rho) = \sum_{i=0}^{n-1} \theta_i * cr(m_i) + \omega(t_i)$$

Definition 10 (Optimal cost of a sequence). *The optimal cost $\Omega(\sigma)$ of the sequence of transitions σ is*

$\Omega(\sigma) = \Omega_r(\rho)$ such that $sequence(\rho) = \sigma$ and $\nexists \rho' \in \mathsf{Runs}(\mathcal{N}) \mid \Omega_r(\rho') < \Omega_r(\rho)$.

Since C_σ is the set of all states that can be reached by the sequence σ, we also denote $\Omega(C_\sigma) = \Omega(\sigma)$.

3 Cost State classes

We now extend the notion of state class to additionally include an information on the cost of the corresponding runs. We call *cost state classes* these extended state classes.

Recall that the firing domain D of a classic state class $C_\sigma = (m, D)$ of [6,5] is a convex polyhedron constraining the firing times of the transitions enabled by m. Note that these firing times are relative to the absolute firing date of the last transition of σ (or 0 for the initial class). For an enabled transition t_i, we denote by θ_i the corresponding variable in D.

Cost state classes $L_\sigma = (m, F)$ extend the firing domain with an additional cost variable c, initially null, and evolving as described in the semantics above, and using the following observation: since firing dates are relative to the last fired transition, the time spent in a class before firing some transition t_i is exactly θ_i.

Computing the successive cost state classes then naturally extends the classic computation of [6,5] as follows:

- the initial cost state class is: $L_\varepsilon = (m_0, \{\theta_i \in I_s(t_i) | t_i \in En(m_0)\} \wedge \{c = 0\})$
- a transition t_f is firable from class $L_\sigma = (m, F)$ iff:
  - t_f is enabled by m;
  - $F \wedge \bigwedge_{i \neq f} \theta_f \leq \theta_i \neq \emptyset$.
- the successor $L_{\sigma.t_f}$ of cost state class L_σ by a transition t_f firable from L_σ is given by Algorithm 1.

Algorithm 1 Successor $L' = (m', F')$ of $L = (m, F)$ by firing t_f: $L' = Next(L, t_f)$

```
1: $m' \leftarrow m' = m - {}^\bullet t_f + t_f^\bullet$
2: $F' \leftarrow F \wedge \bigwedge_{i \neq f} \theta_f \leq \theta_i$
3: for all $i \neq f$, add variable $\theta'_i$ to $F'$, constrained by $\theta_i = \theta'_i + \theta_f$ to $F'$
4: add variable $c'$ to $F'$, constrained by $c' = c + \theta_f * cr(m) + \omega(t_f)$
5: eliminate (by projection) variables $c$, $\theta_i$ for all $i$, and $\theta'_j$ for all $t_j$ disabled by firing
   $t_f$, from $F'$
6: for all $t_j \in \mathcal{N}ewlyEn(m, t_f)$, add variable $\theta'_j$, constrained by $\theta'_j \in I_s(t_j)$
```

Remark that the only change to the classic successor computation in Algorithm 1 is the addition of line 4 (and of course the elimination of c in line 5).

By iteratively computing the extended state classes we obtain a possibly infinite graph with edges labeled by fired transitions and nodes by classes. The quotient of the graph by the equivalence relation $\equiv$ defined by $(m, F) \equiv (m', F')$ iff $m = m'$ and $F = F'$ (in the sense that the polyhedra contain the same points), provides a finite graph for regular state classes, when the net is bounded. This is however not necessarily the case with cost state classes since the cost variable c may increase or decrease unboundedly, and its relation to the other variables may be arbitrarily complex (though still linear).

Lemma 1 **($L_{\sigma|\theta} \cong C_\sigma$).** *Let σ a firable sequence from the initial state, $L_{\sigma|\theta} \cong C_\sigma$.*

A corollary of lemma 1 is that $Next(L_{\sigma|\theta}, t) \cong Next(L_\sigma, t)_{|\theta}$.

Lemma 2 **(Optimal cost of L_σ).** *$\Omega(\sigma) = inf(L_{\sigma|c})$.*

We will now denote $\Omega(L_\sigma) = \Omega(C_\sigma) = \Omega(\sigma)$.

4 Symbolic Algorithm

Now that we have a symbolic abstraction, we can reuse the symbolic algorithm from [12,15], originally designed for priced zones. The only property we need to ensure correctness and soundness is that we can extract the minimum cost for a given sequence of transitions. We have seen how to do that for cost state classes in the previous section.

So, given a target set of markings Goal, if Algorithm 2 terminates, it will provide the optimal cost to reach Goal.

Algorithm 2 Symbolic algorithm for optimal cost

```
 1: COST ← $\infty$
 2: PASSED ← $\emptyset$
 3: WAITING ← $\{(m_0, F_0)\}$
 4: while WAITING $\neq \emptyset$ do
 5:    select $L_\sigma = (m, F)$ from WAITING
 6:    if $m \in$ Goal and $\Omega(L_\sigma) <$ COST then
 7:       COST ← $\Omega(L_\sigma)$
 8:    end if
 9:    if for all $L' \in$ PASSED, $L_\sigma \not\preccurlyeq L'$ then
10:       add $L_\sigma$ to PASSED
11:       for all $t \in \mathcal{F}irable(L_\sigma)$, add $L_{\sigma.t}$ to WAITING
12:    end if
13: end while
14: return COST
```

The algorithm consists in a classic exploration of the symbolic state-space, updating the optimal cost whenever we visit a marking in Goal. It uses a passed list to store already visited symbolic states but since the cost is not bounded a priori there is no reason the same states will eventually repeat.

To overcome this difficulty the algorithm uses a dedicated comparison operator $\preccurlyeq$ between symbolic states that is easily adapted to cost state classes as follows.

For any cost state class $L = (m, F)$ and any point $\boldsymbol{\theta} \in F_{|\theta}$, the optimal cost of $\boldsymbol{\theta}$ in F is defined by $\Omega_F(\boldsymbol{\theta}) = \min_{(\boldsymbol{\theta},c)\in F} c$.

In the sequel, given a point $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n) \in F_{|\theta}$, we often write $(\boldsymbol{\theta}, c)$ instead of $(\theta_1, \ldots, \theta_n, c)$ for the corresponding point in F with cost value c.

Definition 11. *Let $L = (m, F)$ and $L' = (m', F')$ two cost state classes. We say that L is subsumed by L', which we denote by $L \preccurlyeq L'$ iff $m = m'$ and for all $F_{|\theta} \subseteq {F'}_{|\theta}$, and for all $\boldsymbol{\theta} \in F_{|\theta}, \Omega_{F'}(\boldsymbol{\theta}) \leq \Omega_F(\boldsymbol{\theta})$.*

Relation $\preccurlyeq$ can be checked for cost state classes in the same way proposed for priced zones in [15]: consider $(m, F) \preccurlyeq (m', F')$, then $m = m'$ and $F \subset F'$ are easy to check as polyhedral operations. To check the last condition, if c is the cost variable in F and c' the cost variable in F', we need only minimize $c - c'$ on F and check that it is non negative. This minimisation can again be done using classic polyhedral operations, here, for instance, the simplex method.

We can however also reduce $\preccurlyeq$ checking to standard inclusion on polyhedra.

Given a cost state class $L_\sigma = (m, F)$, we denote by $\uparrow F$ the convex polyhedron obtained from F by removing all upper bound constraints on cost variable c (or equivalently, by adding an extremal ray in the direction of c). By extension, we note $\uparrow L_\sigma = (m, \uparrow F)$.

It is easy to see that for all points $(\theta_1, \ldots, \theta_n, c')$ in $\uparrow F$ there exists a point $(\theta_1, \ldots, \theta_n, c)$, with $c \leq c'$ in F and therefore $\Omega(\uparrow L_\sigma) = \Omega(L_\sigma)$. This also implies that for any transition t firable from L_σ, the successor of $\uparrow L_\sigma$ by t (obtained with Algorithm 1) is equal to $\uparrow L_{\sigma.t}$. Furthermore, we have the following lemma.

Lemma 3. *Let L and L' be two cost state classes. We have $L \preccurlyeq L'$ iff $\uparrow L \subseteq \uparrow L'$.*

Now, to prove that the algorithm indeed always terminates, we first have to show that relaxed cost state classes can always be partitioned in a finite number of cost state classes with only one lower bound constraint on the cost variable.

Definition 12. *A* simple cost state class *is a cost state class such that its domain contains only one constraint over the cost variable and this constraint is a lower bound constraint.*

This will also give us a usually more efficient way to symbolically compute the optimal cost, using the efficient DBM data structure and, in particular, minimisation of a linear expression over a DBM, instead of the simplex or polyhedral inclusion, using the results of [15]. Suppose we have two simple cost state classes L and L': their firing domains F and F' can be decomposed as DBMs D and D', each with an additional constraint on the cost variable, $c \geq \ell(\boldsymbol{\theta})$ and $c' \geq \ell'(\boldsymbol{\theta})$. Then instead of minimizing $c - c'$ over F, we only need to minimize $\ell(\boldsymbol{\theta}) - \ell'(\boldsymbol{\theta})$ over D, which is usually much easier [15].

5 Computing the simple cost state classes

We now show how we can partition relaxed cost state classes into *simple* cost state classes. Note that the initial cost state class, once relaxed with $\uparrow$, is indeed a simple cost state class. We then focus on computing the successors of simple cost state classes.

Let us consider a simple cost state class $L = (m, F)$ where F is a combination of a classic firing domain D, written as a DBM, and of a linear inequality over variables θ_i constraining the cost c. To ease further reading, we also define sets $\mathcal{E}$ as $En(m)$ and $\mathcal{E}_f$ as $En(m) \setminus \{t_f\}$. The firing domain F is thus defined:

$$F : \begin{cases} D : \begin{cases} \forall t_i \in \mathcal{E} & \alpha_i \leq \theta_i \leq \beta_i \\ \forall t_i, t_j \in \mathcal{E} & \theta_i - \theta_j \leq \gamma_{ij} \end{cases} \\ c \geq \sum\limits_{t_i \in \mathcal{E}} a_i \theta_i + b \end{cases}$$

Let us compute its successor $L' = (m', F')$ by firing transition t_f following Algorithm 1 and show that L' can be written as a finite union of simple cost state classes.

Applying line 2 simply means that we modify D by adding the constraint $\theta_f \leq \theta_i$ for all t_i in $\mathcal{E}_f$. Following line 3, we then replace θ_i by $\theta'_i + \theta_f$; after simplification, we obtain the following domain:

$$F_3 : \begin{cases} D_3 : \begin{cases} \alpha_f \leq \theta_f \leq \beta_f \quad \textbf{(5.1)} \\ \forall t_i \in \mathcal{E}_f \quad \begin{cases} \alpha_i - \theta'_i \leq \theta_f \leq \beta_i - \theta'_i \ \textbf{(5.2)} \\ \max(0, -\gamma_{fi}) \leq \theta'_i \leq \gamma_{if} \end{cases} \\ \forall t_i, t_j \in \mathcal{E}_f \quad \theta'_i - \theta'_j \leq \gamma_{ij} \end{cases} \\ c \geq \sum\limits_{t_i \in \mathcal{E}_f} a_i \theta'_i + \left(\sum\limits_{t_i \in \mathcal{E}} a_i \right) \theta_f + b \end{cases}$$

We then compute the constraint on the new cost c', according to line 4 of the algorithm: $c' \geq \sum_{t_i \in \mathcal{E}_f} a_i \theta'_i + C * \theta_f + B$ **(5.3)** where $C = cr(m) + \sum_{t_i \in \mathcal{E}} a_i$ and $B = b + \omega(t_f)$.

Before proceeding to line 5 of the algorithm, in which we need to eliminate θ_f (amongst other variables) from the system, let us notice that only inequalities **(5.1)**, **(5.2)** and **(5.3)** involve θ_f. To eliminate θ_f by projection, we use Fourier–Motzkin elimination (FME): we keep all the inequalities in F_4 that don't involve θ_f and we add all the inequalities stating that any lower bound of θ_f should be lower than any of its upper bounds. We obtain the following system:

$$F_5 : \begin{cases} D_5 : \begin{cases} \forall t_i \in \mathcal{E}_f & \max(0, -\gamma_{fi}, \alpha_i - \beta_f \leq \theta'_i \leq \min(\gamma_{if}, \beta_i - \alpha_f) \\ \forall t_i, t_j \in \mathcal{E}_f & \theta'_i - \theta'_j \leq \min(\gamma_{ij}, \beta_i - \alpha j) \end{cases} \\ c' \geq \begin{cases} \max\left(\alpha_f, \max\limits_{t_i \in \mathcal{E}_f} (\alpha_i - \theta'_i)\right) * C + \sum\limits_{t_i \in \mathcal{E}_f} a_i \theta'_i + B & \text{if } C \geq 0 \\ \min\left(\beta_f, \min\limits_{t_i \in \mathcal{E}_f} (\beta_i - \theta'_i)\right) * C + \sum\limits_{t_i \in \mathcal{E}_f} a_i \theta'_i + B & \text{otherwise} \end{cases} \end{cases}$$

Again, D_5 is a DBM; following Lemma 1, it is indeed equal to the DBM obtained by a computation of the next state without considering the cost. On a side note, exact expressions for the bounds of the canonical form of this DBM can be found in [7,8]. We now consider that D_5 is defined by:

$$D_5 : \begin{cases} \forall t_i \in \mathcal{E}_f & \alpha'_i \leq \theta'_i \leq \beta'_i \\ \forall t_i, t_j \in \mathcal{E}_f & \theta'_i - \theta'_j \leq \gamma'_{ij} \end{cases}$$

In our aim to obtain an union of simple cost state classes, we shall now consider the constraints on the new cost c'. Let us suppose that $C \geq 0$; the constraint over c' can be split in two cases: either α_f is the largest coefficient, or one transition $t_I \in \mathcal{E}_f$ yields largest coefficient. Supposing that α_f is indeed the largest coefficient, we know that $\alpha_i - \theta'_i \leq \alpha_f$ for all t_i in $\mathcal{E}_f$ and that $c' \geq \alpha_f * C + \sum_{t_i \in \mathcal{E}_f} a_i \theta'_i + B$. By combining these constraints with F_5, we obtain the following simple cost state class:

$$F'_5 : \begin{cases} D'_5 : \begin{cases} \forall t_i \in \mathcal{E}_f & \max(\alpha'_i, \alpha_i - \alpha_f) \leq \theta'_i \leq \beta'_i \\ \forall t_i, t_j \in \mathcal{E}_f & \theta'_i - \theta'_j \leq \gamma'_{ij} \end{cases} \\ c' \geq \alpha_f * C + \sum_{t_i \in \mathcal{E}_f} a_i \theta'_i + B \end{cases}$$

All other cases (e.g. one of the $\alpha_I - \theta'_I$ is the greatest coefficient, and also the cases when $C < 0$) also lead to adding constraints preserving the DBM form, and we can thus show that F_5 can indeed be split as a finite union of simple cost state classes of the following form:

$$F'_5 : \begin{cases} D'_5 : \begin{cases} \forall t_i \in \mathcal{E}_f & \alpha''_i \leq \theta'_i \leq \beta''_i \\ \forall t_i, t_j \in \mathcal{E}_f & \theta'_i - \theta'_j \leq \gamma''_{ij} \end{cases} \\ c' \geq \sum_{t_i \in \mathcal{E}_f} a'_i \theta'_i + B' \end{cases}$$

In order to complete line 5 of the algorithm, we need to eliminate in all domains F'_5 all variables refering to transitions that have been disabled by the firing of t_f. Let t_k be such a transition; to eliminate θ'_k from F'_5, we apply the FME method again. Note that, to eliminate θ'_k in D'_5, provided D'_5 is in canonical form, we simply erase any inequality involving this variable, which gives us DBM D''_5; we therefore focus on inequalities over the cost c' and obtain the following domain:

$$F'_5 : \begin{cases} D''_5 \\ c' \geq \begin{cases} \max\left(\alpha''_k, \max\limits_{t_i \in \mathcal{E}_f \setminus \{t_k\}} (\theta'_i - \gamma''_{ik})\right) * a'_k + \sum\limits_{t_i \in \mathcal{E}_f \setminus \{t_k\}} a'_i \theta'_i + B' & \text{if } a'_k \geq 0 \\ \min\left(\beta''_f, \min\limits_{t_i \in \mathcal{E}_f \setminus \{t_k\}} (\theta'_i + \gamma''_{ki})\right) * a'_k + \sum\limits_{t_i \in \mathcal{E}_f} a'_i \theta'_i + B' & \text{otherwise} \end{cases} \end{cases}$$

Again, we can split the constraint on c' to obtain a finite union of simple cost state classes and iterate the process for all the transitions that have been disabled by the firing of t_f.

Finally, we add the constraints given by line 6 to finish the computation of F'. In the end, we indeed obtain the successor of our initial simple cost state class as a finite union of simple cost state classes.

Each of the elements of this finite union can then be considered as a standalone successor of that state class in Algorithm 2 much like in [12,15].

6 Termination of the algorithm

To prove the termination, we consider $\succcurlyeq$ the symmetric relation to $\preccurlyeq$, such that $x \succcurlyeq y$ iff $y \preccurlyeq x$, and prove that it is a well quasi-order (wqo), i.e., that for every infinite sequence of cost state classes, there are at least L and L' in the sequence, with L strictly preceding L' such that $L \succcurlyeq L'$. This implies that the exploration of children in Algorithm 2 will always eventually stop.

The idea is to first prove that $\succcurlyeq$ is a wqo on *simple* cost state classes, and then to lift this result to a certain quasi-order derived from $\succcurlyeq$ and defined on *sets* of simple cost state classes. To ensure the lifted order is indeed a wqo, $\succcurlyeq$ has to have a stronger property: indeed, we need to prove that it is a *better* quasi-order (bqo). The definition of bqo's is a bit involved and we actually do not need to use it explicitly so we refer the interested reader to [14] for instance.

Proposition 1. *Let $\mathcal{N}$ be a bounded TPN such that the cost of all runs is uniformly lower-bounded by some constant m, then relation $\succcurlyeq$ is a better quasi-order on the simple cost state classes of $\mathcal{N}$.*

The wqo on cost state classes and the termination of Algorithm 2 are rather direct consequences of Proposition 1.

Corollary 1. *Let $\mathcal{N}$ be a bounded TPN such that the cost of all runs is uniformly lower-bounded by some constant M, then relation $\succcurlyeq$ is a well quasi-order on the cost state classes of $\mathcal{N}$.*

Corollary 2. *When $\mathcal{N}$ is bounded and the cost of all runs is uniformly lower-bounded by some constant M, Algorithm 2 terminates.*

7 Practical results

We have implemented the above algorithms in Romeo[5], a tool for the verification of (parametric) time Petri nets [13]. In this section, we illustrate the above approaches with a practical example. It uses negative costs, the point here being to show how to obtain a scheduler using prediction about environmental features.

[5] http://romeo.rts-software.org

EPOC (Energy Proportional and Opportunistic Computing systems)

The EPOC project [11] focusses on energy-aware task execution in the context of a mono-site and small data Center which is connected to the regular electric grid and to local renewable energy sources (such as windmills or solar cells).

Given a reliable prediction model, it is possible to design a scheduling that aims at optimizing resource utilization and energy usage. A power-driven approach allows shifting or scheduling the postponable workloads to the time period when the electricity is available (from the renewable energy sources) or at the best price.

Description: We look here at a small system with four tasks: Task1 can be scheduled at any time with non-renewable energy whereas the other tasks must be computed using renewable energy. To run the four tasks, there are two processors: Task2, Task3 and Task4 can run on both, but Task1 must run on the first processor. Furthermore, the second processor, which can only use renewable energy, is twice as slow as the first processor.

The energy source is assumed to rely on solar cells and wind turbines; as illustrated by Fig. 2, the weather pattern used in the case study is the following:

– 10 a.m.–11:20 a.m.: windy, with a mix of sunny and cloudy;
– 11:20 a.m.–11:30 a.m.: calm and cloudy;
– 11:30 a.m.–11:40 a.m.: calm and weakly sunny;
– 11:40 a.m.–12 p.m. (noon): calm and sunny.

If a task is executed after the deadline, the cost rate is 100. Using non-renewable energy for Task1 has a cost rate of 40. If tasks 2, 3 and 4 are executed during a sunny period, the cost rate is -20; during a period of weak sun, it is -10; and during a windy period, it is -10. Evidently, costs add up: e.g. when the weather is sunny and windy, the cost rate is $-20-10=-30$.

The TPN model is presented in Fig 1. Proc1 and Proc2 stand for the processors (1 and 2).

The associated cost function is: $40 * R1\_1 + (DL) * (R1\_1 + R2\_1 + R2\_2 + R3\_1 + R3\_2 + R4\_1 + R4\_2) * 100 - (1 * Windy + 2 * (Sun1 + Sun2 + Sun3 + Sun4) + 1 * WeakSun) * (R2\_1 + R2\_2 + R3\_1 + R3\_2 + R4\_1 + R4\_2) * 10$.

Objective: We want to reach a marking corresponding to the situation where all tasks have been executed, which corresponds to all places in the upper net being empty except Proc1 and Proc2, which contains exactly one token.

Results: The minimal cost to reach a state such that all the tasks are executed is -1560 and from the associated trace (given by Romeo) we can derive the Gantt chart in Fig 2. As for Fig 3, it shows the evolution of the cost rate during the scenario proposed in the Gantt chart Fig 2.

Table 1 summarizes the performances of Romeo to reach the minimal cost using cost state classes (and polyhedral operations) or simple cost state classes (relying on DBMs). As a sanity check we can remark that both abstractions

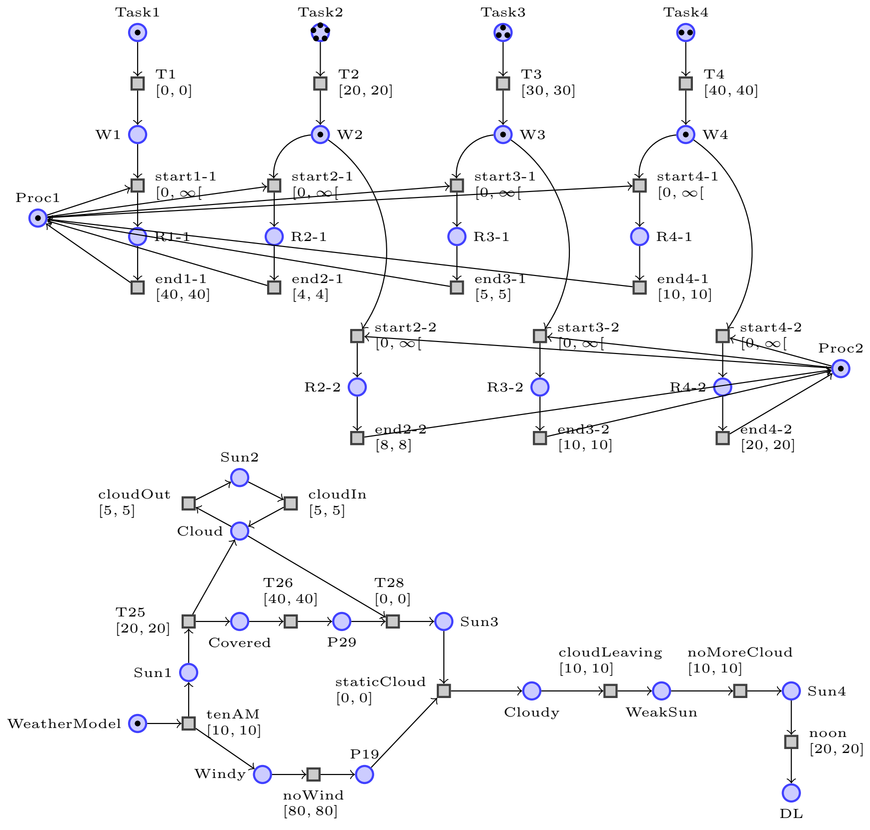

Fig. 1. EPOC example

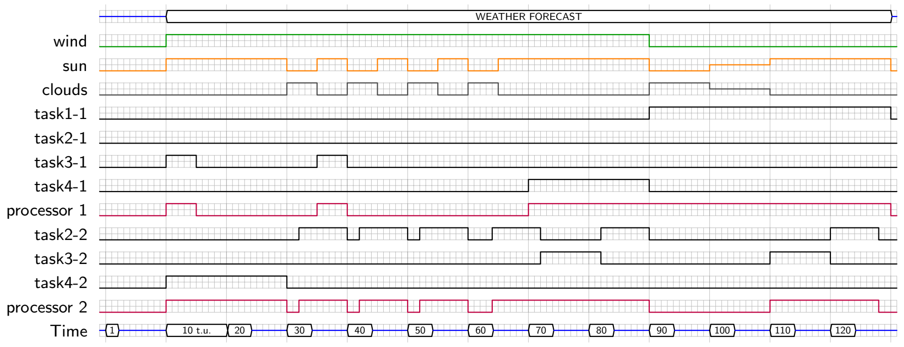

Fig. 2. EPOC: Gantt chart

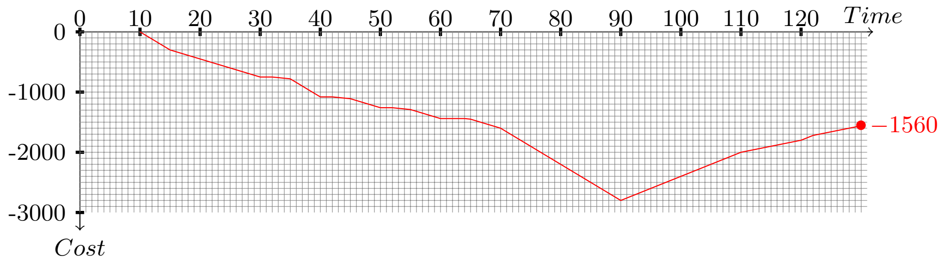

Fig. 3. EPOC: Cost evolution

indeed compute the same minimal cost. As we are still implementing Romeo with cost features, we are not yet able to get consistent data about the memory used for each computation.

For this example, we notice using simple state classes to reach the minimal cost is more efficient (almost 3 times faster): it is something we observed with other examples studied, but not exposed in this paper. Therefore, experiment urge us to favour this method over the use of cost state classes algorithms, even though both methods give correct results.

Table 1. Offline non-preemptive Scheduler: Romeo performances

Method	*Cost State classes*	*SimpleStateClasses*
Minimal cost	-1560	-1560
Computing time	$4856.3s$	$1696.3s$

8 Conclusion

In this paper, we have studied the optimal-cost reachability problem for time Petri nets, where both letting time elapse and firing transitions have costs. We have proposed a forward exploration algorithm based on the state class method that provides the optimal-cost to reach a marking, for all bounded TPN with no negative-cost cycles. We have first defined the reachability cost problem by means of time-dependent cost constraints integrated to state classes and then adapted consequently the firing rule. The optimal-cost to reach a state class from the initial state class is reduced to a linear programming problem. Unlike other approches [2,3,4,9,12,1], the one presented in this paper doesn't need any approximation/extrapolation nor handling dead tokens or transitions. Finally, we have confirmed the effectiveness and efficiency of our approach through a case study.

References

1. Parosh A. Abdulla and Richard Mayr. Priced Timed Petri Nets. *Logical Methods in Computer Science*, 9(4), 2013.
2. Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, 318(3):297 – 322, 2004.
3. Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. *Minimum-Cost Reachability for Priced Timed Automata*, pages 147–161. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
4. Gerd Behrmann, Kim G. Larsen, and Jacob I. Rasmussen. Optimal scheduling using priced timed automata. *SIGMETRICS Perform. Eval. Rev.*, 32(4):34– 40, March 2005.
5. Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time petri nets. *IEEE Trans. Software Eng.*, 17(3):259–273, 1991.
6. Bernard Berthomieu and Miguel Menasche. An enumerative approach for analyzing time petri nets. In *IFIP Congress*, pages 41–46, 1983.
7. Hanifa Boucheneb and John Mullins. Analyse des réseaux temporels : Calcul des classes en $O(n^2)$ et des temps de chemin en $O(m \times n)$. *TSI. Technique et science informatiques*, 22(4):435–459, 2003.
8. Pierre-Alain Bourdil, Bernard Berthomieu, Silvano Dal Zilio, and François Vernadat. Symmetry reduction for time petri net state classes. *Science of Computer Programming*, 132:209–225, 2016.
9. Patricia Bouyer, Maximilien Colange, and Nicolas Markey. Symbolic optimal reachability in weighted timed automata. In *Proceedings of the 28th International Conference on Computer Aided Verification (CAV'16)*, volume 9779 of *Lecture Notes in Computer Science*, pages 513–530, Toronto, Canada, July 2016. Springer.
10. D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proc. Int. Workshop Automatic Verification Methods for Finite State Systems (CAV'89)*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.
11. EPOC. Energy proportional and opportunistic computing systems. http://www.epoc.cominlabs.ueb.eu/fr.
12. Kim Larsen, Gerd Behrmann, Ed Brinksma, Ansgar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. *Lecture Notes in Computer Science*, 2102:493–505, 2001.
13. Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez. Romeo: A parametric model-checker for Petri nets with stopwatches. In Stefan Kowalewski and Anna Philippou, editors, *15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009)*, volume 5505 of *Lecture Notes in Computer Science*, pages 54–57, York, United Kingdom, March 2009. Springer.
14. Alberto Marcone. Fine analysis of the quasi-orderings on the power set. *Order*, 18(4):339–347, 2001.
15. Jacob I. Rasmussen, Kim G. Larsen, and K. Subramani. On using priced timed automata to achieve optimal scheduling. *Formal Methods in System Design*, 29(1):97–114, 2006.