Évaluation de stratégies de régénération dans un modèle de vente de services : une approche par réseaux de Petri à coût

Didier Lime[2], Pascale Marangé[1], Rémi Parrot[2], and Olivier H. Roux[2]

[1] Université de Lorraine, CRAN, UMR 7039, Campus Sciences BP 70239, 54506 Vandoeuvre-lès-Nancy, France
pascale.marange@univ-lorraine.fr
[2] Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France
{prenom.nom}@ec-nantes.fr

Abstract

Une mise en oeuvre d'une économie circulaire peut être basée sur la vente de services, au lieu du produit lui-même. Les entreprises qui mettent en place cette vente de service ont besoin de déterminer les moments opportuns pour la maintenance, la régénération du produit ou des composants, afin de maximiser la durabilité de vie du produit tout en minimisant les coûts d'entretien.

Nous proposons une modélisation de ces processus à l'aide des réseaux de Petri à coût, intégrant des aspects temporels, économiques et état de santé des produits. Les transitions temporisées permettent de planifier les interventions de maintenance et de fin de location. Les coûts associés aux places et aux transitions modélisent les dépenses liées à la maintenance, à la collecte, au tri et à la régénération, qui sont à la charge de l'entreprise dans ce modèle économique. Pour la modélisation et l'analyse, nous utilisons le logiciel Roméo, permettant la modélisation, la simulation et la vérification de réseaux de Petri temporels, paramétrés et à coût. Il offre des fonctionnalités avancées telles que la synthèse de paramètres pour le model-checking d'une sous-classe de la logique temporelle TCTL, la synthèse conjointe de paramètres et de contrôleurs, ainsi que la synthèse de paramètres pour l'accessibilité optimale avec un modèle de coût affine.

Cette approche permet d'évaluer différentes stratégies de régénération (réutilisation, remanufacturation, recyclage) en fonction de paramètres tels que la durée de location, les dates de maintenance et les critères d'état de santé des composants. Elle offre ainsi un outil d'aide à la décision pour les entreprises souhaitant optimiser leurs opérations dans le cadre d'une offre de services.

1 Introduction

L'économie circulaire [12] s'impose aujourd'hui comme un levier essentiel pour répondre aux défis environnementaux, économiques et sociétaux liés à l'épuisement des ressources, à la production de déchets et à la nécessaire réduction de l'empreinte carbone des activités industrielles. En rompant avec le modèle linéaire traditionnel "extraire-produire-consommer-jeter", elle vise à prolonger la durée de vie des produits, à favoriser leur réutilisation, leur réparation ou leur régénération, et à maximiser la valorisation des ressources.

La mise en œuvre concrète de l'économie circulaire soulève des problématiques complexes de décision. Elle implique de choisir entre différentes stratégies de régénération : réemploi, recyclage, reconfiguration..., en tenant compte de plusieurs paramètres : état de santé des produits, coûts de la logistique/ de la maintenance/de la régénération, durée d'usage, ou encore performance environnementale. Ces choix nécessitent des approches adaptées de modélisation, d'évaluation multicritère et parfois de synthèse ou de contrôle, afin d'orienter les décisions vers des compromis viables entre efficacité économique et impact environnemental.

Dans ce contexte, les réseaux de Petri temporisés à coût et à paramètres apparaissent comme un formalisme pertinent, car ils permettent d'articuler des aspects temporels, économiques et structurels, tout en offrant des capacités avancées d'analyse, de simulation et de vérification. Leur expressivité autorise une représentation de la dégradation des produits, des logiques de maintenance ou de pilotage et des processus de régénération.

L'objectif de ce papier est de montrer que les réseaux de Petri constituent une réponse aux problématiques posées par l'économie circulaire, en particulier lorsqu'il s'agit de piloter des stratégies sous contraintes. Pour cela, deux études de cas sont développées : la première concerne la vente de services dans un modèle d'économie de la fonctionnalité, la seconde explore les décisions de régénération en fonction de l'état de santé du produit. Ces cas d'étude démontrent la capacité des réseaux de Petri à modéliser, simuler et guider des choix opérationnels.

La suite de l'article est organisée comme suit. La section 2 présente deux études de cas illustrant des situations industrielles typiques de mise en œuvre de l'économie circulaire. La section 3 introduit le formalisme des réseaux de Petri temporels (TPN) à coût et à paramètres, ainsi que les outils utilisés pour la vérification et l'analyse. La section 4 détaille l'application de ce formalisme TPN à un système de vente de services. La section 5 présente la modélisation de différentes stratégies de régénération et l'évaluation du coût et de la durée de vie du produit. Enfin, la section 6 propose une discussion des apports et des limites de l'utilisation de l'outil de formalisation TPN et d'analyse, et ouvre des perspectives pour de futurs travaux.

2 Cas d'études

Cette section présente deux études de cas illustrant la mise en œuvre de l'économie circulaire. Le premier cas porte sur l'économie de la fonctionnalité, où l'entreprise vend un service d'usage plutôt qu'un produit. Le second analyse les stratégies de production liés à la régénération de produits collectés et à leurs états de santé variables. Ces cas permettent d'analyser les enjeux économiques et environnementaux associés à deux approches complémentaires : (i) l'optimisation de l'usage et de la maintenance et (ii) la revalorisation post-consommation.

2.1 Cas d'étude n°1 : Évaluation des stratégies de maintenance dans une économie de la fonctionnalité lors de l'usage du produit

Dans le cadre de la transition vers un modèle plus soutenable, certaines entreprises ont adopté une approche d'économie de la fonctionnalité, consistant à proposer un service d'usage en lieu et place de la vente de biens [14]. Cette transformation du modèle économique impose une redéfinition de l'équilibre entre performance économique et environnementale.

Les entreprises doivent repenser la conception de leurs produits, leurs relations contractuelles avec leurs clients, ainsi que la structuration de leurs chaînes logistiques et de maintenance. Le passage d'un modèle transactionnel à un modèle serviciel nécessite d'optimiser plusieurs paramètres sur le cycle de vie du service : durée de location, fréquence de maintenance, retours logistiques et renouvellement des composants... Dans cette étude de cas, une entreprise du secteur a mis en place une offre basée sur la location de produits avec un contrat de service associé : maintenance incluse et reprise en fin de contrat. Pour assurer la viabilité du modèle, l'entreprise souhaite analyser différents scénarios en fonction de plusieurs critères :

- Coût total du service sur la durée du contrat, en intégrant les coûts de maintenance.
- Empreinte environnementale, mesurée à l'aide d'indicateurs comme les émissions de GES, les kilomètres parcours, ou la durabilité des produits.

- Satisfaction client et continuité de service, incluant la fréquence des interventions de maintenance préventive et corrective.

Dans cette étude de cas, nous nous interesserons à la stratégie de maintenance pour analyser à quelle fréquence doivent être faites les maintenances préventives par rapport au délai d'indisponibilité du produit et aux couts des maintenance correctives

2.2 Cas d'étude n°2 : Évaluation de stratégies de pilotage de l'atelier lors de la régénération du produit

L'économie circulaire repose notamment sur la capacité à régénérer les produits en fin de vie, non pas uniquement par le recyclage, mais en privilégiant des stratégies à plus forte valeur ajoutée comme le réemploi ou le remanufacturing, regroupés sous le nom de régénération [15]. Cette seconde étude de cas s'inscrit dans le contexte d'une entreprise opérant des ateliers de régénération, dans lesquels les produits collectés sont évalués, triés et orientés vers différents parcours de régénération. Une difficulté pour les ateliers de régénération réside dans la variabilité de l'état de santé des produits collectés. Cette hétérogénéité a un double impact :

- Économique : plus un produit est dégradé, plus son coût de régénération est élevé, remettant parfois en question la rentabilité de l'opération.
- Organisationnel : les ateliers doivent être flexibles et reconfigurables, afin d'adapter les processus à la régénération, tout en gérant les contraintes logistiques et de stockage.

Cette deuxième étude de cas repose sur une modélisation des flux de produits à régénérer dans un atelier proposant deux parcours de régénération :

- Réemploi, après tests et reconditionnement léger. Il y a pas d'impact sur l'état de santé du produit,
- Recyclage, via la revalorisation de la matière. La vie du produit s'arrête à cette étape.

Chacun de ces parcours a un impact économique dépendant des paramêtres suivants :

- La taille des stocks, sachant que les produits peuvent continuer à se dégrader durant le stockage, affectant leur régénération possible.
- La taille des lots à lancer pour chaque type de régénération, en prenant en compte les coûts fixes de reconfiguration des ateliers, les ressources disponibles, et la prolongation effective de la durée de vie induite par chaque stratégie.
- Le moment de déclenchement des opérations de revalorisation, en arbitrant entre le regroupement et la rapidité.

En fonction de l'arrivée aléatoire des produits, l'évolution de leur état de santé, et les choix stratégiques associés aux lancements de lots de régénération, les performances sont évaluées selon des critères économiques (coût de traitement, valeur récupérée).

3 Réseaux de Petri temporels et extensions

Cette section introduit les différents formalismes utilisés Nous donnons une définition des réseaux de Petri temporels avec leur sémantique, puis nous définissons des extensions avec des coûts, des paramètres, des variables haut-niveau, ainsi que les jeux de réseaux de Petri.

Réseau de Petri temporel On note respectivement $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ et $\mathbb{R}$ les ensembles des entiers positifs, des entiers relatifs, des rationels et des réels. Les ensembles des rationels positifs et les réels positifs sont notés respectivement $\mathbb{Q}_+$ et $\mathbb{R}_+$. Pour un ensemble fini X, $|X|$ dénote son cardinal. Soit un ensemble de nombres X, on note $\mathbb{I}(X)$ l'ensemble des intervalles dont les bornes sont dans X.

Étant donnés deux ensembles V et X, une V-valuation (ou simplement une valuation si le contexte est clair) de X est une fonction de $X \mapsto V$. Lorsque X est fini, on confondra une V-valuation de X avec un vecteur de $X^{|V|}$.

Définition 1 (Réseau de Petri)**.** *Un réseau de Petri (PN) est un n-uplet* $\mathcal{N} = (P, T, {}^\bullet., .^\bullet, m_0)$*, où : P est un ensemble fini de* places*, T est un ensemble fini de* transitions*, ${}^\bullet.: T \mapsto \mathbb{N}^P$ est la fonction d'incidence en arrière, $.^\bullet: T \mapsto \mathbb{N}^P$ est la fonction d'incidence en avant et $m_0 \in \mathbb{N}^P$ est le* marquage *initial.*

Définition 2 (Réseau de Petri temporel)**.** *Un réseau de Petri temporel (TPN) est un n-uplet* $\mathcal{N} = (P, T, {}^\bullet., .^\bullet, m_0, I)$*, où :* $(P, T, {}^\bullet., .^\bullet, m_0)$ *est un PN et* $I : T \mapsto \mathbb{N}$ *est la fonction d'*intervalle de tir.

Un marquage est un vecteur de $\mathbb{N}^P$. Pour un marquage $m \in \mathbb{N}^P$, $m(p)$ represente le nombre de *jeton* dans la place p. Une transition $t \in T$ est dite *sensibilisée* par un marquage $m \in \mathbb{N}^P$ si $m \geq {}^\bullet t$. On note $\mathsf{enab}(m)$ l'ensemble des transitions sensibilisées par m : $\mathsf{enab}(m) = \{t \in T \mid m \geq {}^\bullet t\}$. Depuis un marquage m le tir d'une transition t mène au nouveau marquage $m' = m - {}^\bullet t + t^\bullet$. Une transition $t' \in T$ est dite *nouvellement sensibilisée* par le tir d'une transition t depuis un marquage m si elle est sensibilisée par le nouveau marquage mais qu'elle ne l'est pas par $m - {}^\bullet t$. On note $\uparrow \mathsf{enab}(m, t)$ l'ensemble des transitions nouvellement sensibilisées par le tir de t depuis m : $\uparrow \mathsf{enab}(m, t) = \{t' \in T \mid t' \in \mathsf{enab}(m - {}^\bullet t + t^\bullet)$ et $(t' \notin \mathsf{enab}(m - {}^\bullet t)$ ou $t' = t)\}$

L'état d'un TPN est une paire (m, v) où m est un marquage et $v \in \mathbb{R}_+^T$ est une *valuation d'horloges*. $v(t)$ représente le temps écoulé depuis que la sensibilisation de la transition t. $\mathbf{0}$ est la valuation d'horloge initiale, telle que $\forall t \in T, \mathbf{0}(t) = 0$. Soit $d \in \mathbb{R}_+$, on note $v' = v + d$ la nouvelle valuation d'horloge, telle que $\forall t \in T$, $v'(t) = v(t) + d$.

Définition 3 (Sémantique d'un TPN)**.** *La sémantique d'un TPN est un* système de transition temporisé $(Q, q_0, A, \to)$ *où : l'ensemble des états est $Q \subset (\mathbb{N}^P \times \mathbb{R}_+^T)$, l'état initial est $q_0 = (m_0, \boldsymbol{0})$, l'ensemble des actions est $A = T$ et $\to$ défini deux types de transitions :*

transition discrète

pour tout $t \in T$ *:* $(m, v) \xrightarrow{t} (m', v')$ *ssi*

$$\begin{cases} t \in \mathsf{enab}(m) \ \text{ et } m' = m - {}^\bullet t + t^\bullet \\ v(t) \in I(t) \\ \forall t' \in T v'(t') = \begin{cases} 0 & \text{si } t' \in \uparrow \mathsf{enab}(m, t) \\ v(t') & \text{sinon.} \end{cases} \end{cases}$$

transition continue

pour tout $d \in \mathbb{R}_+$ *:* $(m, v) \xrightarrow{d} (m', v')$ *ssi*

$$\begin{cases} m' = m \\ v' = v + d \\ \forall t' \in \mathsf{enab}(m), v(t) + d \in I(t) \end{cases}$$

Exemple 1. *Un exemple de TPN est dessiné dans la figure 1. C'est un graphe biparti dont les nœuds circulaires sont les places et les nœuds rectangulaires sont les transitions. Le marquage est représenté par des jetons dans les places et les intervalles de tir associés à chaque transition sont écrits à côté.*

Dans cet exemple, t_0 et t_1 modélisent deux processus parallèles avec des temporalités différentes, et t_2 modélise une synchronisation de ces deux processus.

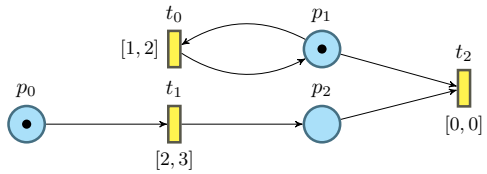

Figure 1: Le TPN $\mathcal{N}_1$ avec le marquage initial $m_0 = \begin{bmatrix}1 & 1 & 0\end{bmatrix}^\intercal$

L'exécution suivante est un exemple d'exécution possible qui termine par le tir de t_2 :

$$\left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}\right) \xrightarrow{1.1} \left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1.1\\1.1\\0\end{bmatrix}\right) \xrightarrow{t_0} \left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1.1\\0\end{bmatrix}\right) \xrightarrow{1.5} \left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1.5\\2.6\\0\end{bmatrix}\right) \xrightarrow{t_1} \left(\begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1.5\\2.6\\0\end{bmatrix}\right) \xrightarrow{t_2} \left(\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1.5\\2.6\\0\end{bmatrix}\right)$$

Extension avec des coûts Nous définissons maintenant une première extension des TPN avec des coûts. Nous verrons dans la suite que cette extension permet de modéliser des objectifs quantitatifs et d'optimisation.

Définition 4 (Réseau de Petri temporel à coûts)**.** *Un réseau de Petri temporel à coûts (cTPN) est un n-uplet $\mathcal{N} = (P, T, {}^\bullet., .^\bullet, m_0, I, \mathsf{cost}_t, \mathsf{cost}_m)$, où : $(P, T, {}^\bullet., .^\bullet, m_0, I)$ est un TPN, $\mathsf{cost}_t \colon T \mapsto \mathbb{Z}$ est la fonction de* coût discret *et $\mathsf{cost}_m \colon \mathbb{N}^P \mapsto \mathbb{Z}$ est la fonction de* coût continu*.*

L'état d'un cTPN est un n-uplet $(m, v, c) \in \mathbb{N}^P \times \mathbb{R}_+^T \times \mathbb{R}$ où m est un marquage, v est une valuation d'horloges et c un coût cumulé.

La sémantique d'un cTPN est similaire à celle d'un TPN, à ceci près que le coût cumulé c est mis-à-jour à chaque transition. Pour une transition discrète, $(m, v, c) \xrightarrow{t} (m', v', c')$ avec $t \in T$, le coût discret est simplement ajouté : $c' = c + \mathsf{cost}_t(t)$. Pour un transition continue, $(m, v, c) \xrightarrow{d} (m', v', c')$ avec $d \in \mathbb{R}_+$, le temps écoulé pondéré par le coût continu est ajouté : $c' = c + \mathsf{cost}_m(m) * d$.

Exemple 2. *Prenons, par exemple, le cTPN $\mathcal{N}_2$ formé du TPN $\mathcal{N}_1$ de la figure 1 et de la fonction de coût discret $\mathsf{cost}_t = \begin{bmatrix}1 & -10 & -3\end{bmatrix}^\intercal$ et de la fonction de coût continue telle que $\forall m \in \mathbb{N}^P$, $\mathsf{cost}_m(m) = 2 * m(p_0) * m(p_1)$. Nous pouvons jouer la même exécution que précédemment :*

$$\left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, 0\right) \xrightarrow{1.1} \left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1.1\\1.1\\0\end{bmatrix}, 2.2\right) \xrightarrow{t_0} \left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1.1\\0\end{bmatrix}, 3.2\right) \xrightarrow{1.5} \left(\begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1.5\\2.6\\0\end{bmatrix}, 6.2\right)$$
$$\xrightarrow{t_1} \left(\begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1.5\\2.6\\0\end{bmatrix}, -3.8\right) \xrightarrow{t_2} \left(\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1.5\\2.6\\0\end{bmatrix}, -6.8\right)$$

Ce coût peut ensuite permettre de formuler un objectif. Par exemple, on peut se demander s'il est possible d'atteindre le marquage $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\intercal$ avec un coût positif. La réponse est non : le coût maximal que l'on peut atteindre est -4 et il est obtenue par la séquence $1t_0 1t_0 1t_0 t_1 t_2$.

Extension avec des paramètres Nous définissons maintenant une extension des TPN avec des paramètres. Les extensions paramètres permettent d'explorer un espace de conception de manière systématique dans l'objectif de synthétiser un contrôleur.

Définition 5 (Réseau de Petri paramétrique temporel)**.** *Un réseau de Petri paramétrique temporel (pTPN) est un n-uplet $\mathcal{N} = (P, T, {}^\bullet., .^\bullet, m_0, I_p, \mathbb{P})$, où : $(P, T, {}^\bullet., .^\bullet, m_0)$ est un PN, $\mathbb{P}$ est un ensemble fini de* paramètres *et $I_p \colon T \mapsto (\mathbb{N} \cup \mathbb{P})$ est la fonction d'intervalle* paramétrique *de tir.*

L'état d'un pTPN est un n-uplet $(m, v, \gamma) \in \mathbb{N}^P \times \mathbb{R}_+^T \times \mathbb{Q}^{\mathbb{P}}$ où m est un marquage, v est une valuation d'horloges et γ une *valuation de paramètres*.

Soit un objet paramétré x de paramètres $\mathbb{P}$ et une valuation de ses paramètres $\gamma \in \mathbb{Q}^{\mathbb{P}}$. On note $\gamma(x)$ l'objet non-paramétré dans lequel tous les paramètres $p \in \mathbb{P}$ ont été remplacés par leur valeur $\gamma(p)$.

Dans l'état (m, v, γ), $\gamma(I_p)$ est la fonction d'intervalle de tir dans laquelle tous les paramètres $p \in \mathbb{P}$ ont été remplacés par la valeur $\gamma(p)$, et $\gamma(\mathcal{N})$ est le TPN $(P, T, {}^\bullet., .^\bullet, m_0, \gamma(I_p))$ Ainsi, chaque valuation de paramètres défini un nouveau TPN.

Exemple 3. *Un exemple de pTPN avec deux paramètres a et b est dessiné dans la figure 2.*

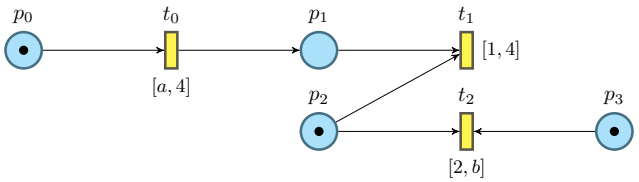

Figure 2: Le pTPN $\mathcal{N}_2$ avec deux paramètres a et b

Sur les modèles paramètrés, on s'intéresse à des problèmes de synthèse. C'est-à-dire qu'on cherche les valeurs de paramètres qui vérifient certaines propriétés.

Par exemple, on peut se poser la question : pour quelles valeurs de paramètre est-il possible d'atteindre le marquage $\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^\intercal$ *? Autrement dit, pour quelles valeurs de paramètre est-il possible de tirer t_1 avant t_2. Au plus tôt, on peut tirer t_0 au temps a et t_1 au temps $a + 1$. Au plus tard, on doit tirer t_2 au temps b. La réponse est donc l'ensemble des valeurs de paramètres qui vérifient $a + 1 \leq b$.*

Jeu de réseau de Petri Nous définissons ici les jeux construits à partir de TPN. Les jeux permettent de modéliser des interactions entre le système et un environnement incontrôlable. Comme pour l'extension paramétrique, ce formalisme a pour objectif de synthétiser un contrôleur.

Définition 6 (Jeu de réseau de Petri temporel)**.** *Un jeu de réseau de Petri temporel (gTPN) est un n-uplet $\mathcal{N} = (P, T_c, T_u, {}^\bullet., .^\bullet, m_0, I)$, où : $T_c \cap T_u = \emptyset$, $(P, T, {}^\bullet., .^\bullet, m_0, I)$ est un TPN avec $T = T_c \cup T_u$, T_c est l'ensemble des transitions contrôlables et T_u est l'ensemble des transitions incontrôlables.*

Dans un gTPN, on ne peut tirer que les transitions contrôlables. Les transitions incontrôlables sont tirés de manière non-déterministe par l'environnement.

Sur un tel modèle, on s'intéresse à la synthèse de *stratégie*. Une stratégie est une fonction qui donne pour toute exécution *passée*, le prochain *coup* à jouer, qui peut-être soit de l'attente (action continue), soit le tir d'une transition (action discrète).

Exemple 4. *Un exemple de gTPN est dessiné dans la figure 3. Les transitions incontrôlables sont représentées en pointillés, ici il n'y a que la transition t'_0.*

Dans un jeu, on s'intéresse à la synthèse de stratégie qui répond à certains critères. On appelle ces dernières stratégies gagnantes.

On peut par exemple se poser la question : quelle stratégie nous assure d'atteindre le marquage $\begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}^\intercal$ *? Pour cela, il faut pouvoir tirer t_2 avant que t_1 soit forcée de tirer. Si on tire t_0 avant une unité de temps, la transition t_1 sera sensibilisée trop tôt et devra tirer avant t_2. Si on attend plus de une unité de temps :*

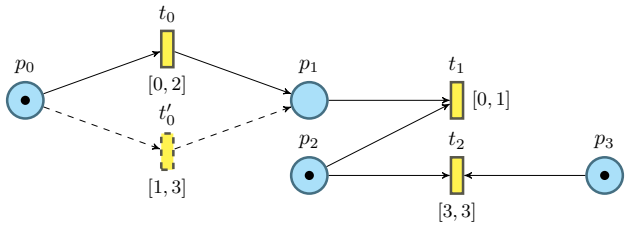

Figure 3: Le gTPN $\mathcal{N}_3$ avec une transition incontrôlable t'_0

- *dans le meilleur cas t'_0 ne tire pas, on attend deux unités de temps avant de tirer t_0, puis on peut tirer t_2 après une unité de temps.*

- *dans le pire cas t'_0 va tirer le plus tôt possible, c'est-à-dire au temps 1, et alors t_1 devra tirer au temps 2.*

Il n'existe donc pas de stratégie gagnante pour cet objectif, c'est-à-dire de stratégie qui nous assure d'atteindre l'objectif.

Extension avec des couleurs et des variables haut-niveau Il existe enfin une extension des TPN avec de la manipulation de variables haut-niveau [7] (hTPN). Dans le cas des réseaux bornés, c'est-à-dire avec un marquage borné, les variables haut-niveau permettent d'introduire du sucre syntaxique qui rend le modèle plus lisible en évitant les motifs compliqués. Lorsque ces variables sont définis dans un ensemble fini, leur utilisation ne change rien à la décidabilité des problèmes classiques tel que l'accessibilité. Nous ne définissons pas formellement ici cette extension, mais donnons simplement son intuition.

La première extension introduite consiste à associer à chaque jeton du réseau une *couleur*. Cette dernière permet alors de différencier les jetons présents dans une place. Ainsi, le marquage d'une place devient un vecteur qui a pour taille le nombre de couleurs. Les transitions sont ainsi multisensibilisées au sens multiserveurs avec un serveur par couleur. Un exemple de réseau de Petri coloré est dessiné dans la figure 6. La place `User` est initialement marquée par trois jetons, un de chaque couleur (donc un vecteur $\left[1,1,1\right]$). Les transitions peuvent ensuite consommer ou produire des jetons sélectivement suivant leur couleurs. Dans l'exemple de la figure 6, les couleurs sont utilisées via des variables haut-niveau que nous présentons juste après.

La seconde extension consiste à ajouter des *variables* au modèle et du *code* pour manipuler ces variables. Il est possible de *mettre à jour* la valeur de ces variables par le tir d'une transition (incrémenter, décrémenter, remettre à zéro, assigner une valeur). Il est également possible de conditionner le tir d'une transition par une *garde* sur ces variables (comparer avec une constante ou une autre variable). Des exemples de hTPN sont représentés dans la suite de l'article (ex: figure 4 et 6). Les gardes sont représentées en vert et les mises à jour sont représentées en bleu.

Ces variables peuvent être directement le marquage d'une place. Par exemple, dans la figure 4, le tir de la transition `Vente` remet à zéro le marquage de la place P_4. Ce modèle s'étend naturellement avec des objets plus haut-niveau comme des tableaux et des fonctions (voir figure 6). Pour finir, toutes ces extensions des TPN peuvent être combinées pour donner des modèles plus expressifs. Nous utiliserons dans la suite un TPN paramétrique avec des variables haut-niveau (figure 4), un jeu de TPN avec des variables haut-niveau (figure 5) et un TPN à coûts avec des variables haut-niveau (figure 6).

Analyse de ces modèles Différents problèmes peuvent être traités par un parcours de l'espace d'états [2] sur ces classes de réseaux de Petri à commencer par le problème du model-checking (vérification) de propriétés telles que l'accessibilité ou des propriétés exprimées dans une logique telle que TCTL [3].

Sur les modèles à paramètres temporels, en plus de l'existence de valeurs de paramètres, il est possible de synthétiser les contraintes sur ces paramètres qui garantissent la satisfaction des propriétés attendues [8]. Concernant les modèles à coût, il est soit possible de prendre en compte une contrainte de coût dans une démarche de model-checking soit de calculer une stratégie (un chemin dans l'espace d'état) permettant d'obtenir le coût minimal pour atteindre un objectif spécifié [4]. Ce même problème peut être traité dans le cas paramètré conduisant à la synthèse des contraintes sur les paramètres permettant d'obtenir ce coût minimal [10].

Enfin concernant le problème de synthèse de contrôleur, les méthodes basées sur les jeux temporisés [13] permettent de synthétiser une stratégie gagnante (i.e. garantissant d'atteindre un objectif donné) en jouant sur les actions contrôlables à partir des états du système. Ces approches ont été étendues au cas paramétré [9]. Toutes ces méthodes sont impémentées dans l'outil Roméo [11].

4 Modélisation de systèmes de vente de services

Considérons le scooter électrique présenté dans la section 2 modélisé par le réseau de Petri temporel paramétré de la figure 4 qui comporte 2 paramètres temporels a et b et une variable entière `money`.

La boucle principale constituée des places P_1 et P_2 représente le cycle annuel avec un entretien d'une durée b. La boucle constituée par les places P_4 et P_5 modélise des pannes éventuelles qui peuvent arrivées après une durée a. La durée de la réparation est de 10 jours. Les `update` des transitions `Vente`, `cycleAnnuel` et `entretien` manipulent directement les marquages des places P_4 et P_5 ce qui est équivalent à des arcs de reset pour les 2 premières et à un arc de type `post` pour la transition `entretien` mais permet d'alléger la figure.

La variable `money` permet de représenter le budget. La location annuelle est de 500€ et le coût de la panne est de 50€. Au bout de 10 ans, le scooter est vendu 500€ après entretien.

Dans un premier temps, nous cherchons à synthétiser les valeurs des paramètres a et b sachant que la valeur du scooter est de 4000€ et que l'on verifie si le modèle d'entreprise est valable. Nous imposons $a \geq 10$ et $b \geq 10$ et nous vérifions donc la propriété AF (P3==1 and money $\geq$ 4000)

Nous obtenons en 5.4 s et avec 122 Mo le résultat suivant : $\begin{cases} a \in [10, inf[\\ b \in [10, 365] \\ 4 * a + b \geq 335 \end{cases}$

Nous modifions les contraintes de départ en considérant que le temps d'entretien doit être inférieur à 20 jours avec $a \geq 10$ et $b \leq 20$

Nous obtenons en 11.6 s et avec 97 Mo : $\begin{cases} a \in [315/4, inf[\\ b \in [0, 20] \\ 4 * a + b \geq 335 \end{cases}$

Coût discret L'utilisation des TPN à coût permet de supprimer la variable `money` en associant un coût discret de -50 à la transition `panne` et de $+500$ aux transitions `cycleAnnuel` et `vente`. Les algorithmes dont nous disposons sur les TPN à coût impose des formules de type `EF` avec des contraintes de type $cost \sim k$ avec $\sim \in \{<, \leq\}$.

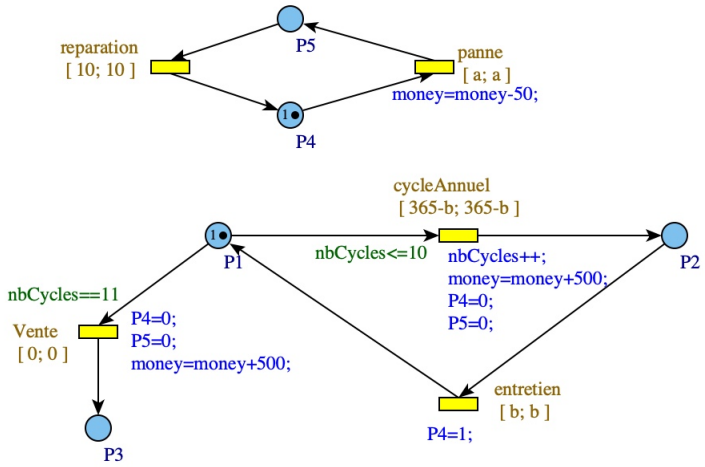

Figure 4: Modèle du scooter

Cependant, nous souhaitons vérifier une formule AF ϕ_1 and ϕ_2 avec ϕ_1= (P3==1) et ϕ_2=(cost $\geq$ 4000) mais nous savons que ϕ_1 arrivera immanquablement et que ϕ_2 ne peut plus changer une fois que ϕ_1 est vrai. Par conséquent nous pouvons vérifier de manière équivalent la formule not EF (ϕ_1 and not ϕ_2) (qui est aussi équivalent à AG (not ϕ_1 or ϕ_2)).

Nous vérifions donc la propriété suivante : not EF(P3==1 and cost $<$ 4000).

Avec les contraintes $a \geq 10$ et $b \geq 10$, nous retrouvons en 82 s et avec $145Mo$ le même résultat suivant : $\begin{cases} a \in [10, inf[\\ b \in [10, 365] \\ 4 * a + b \geq 335 \end{cases}$

Avec les contraintes $a \geq 10$ et $b \leq 20$, on obtient en 59 s et avec 149 Mo le résultat suivant : $\begin{cases} a \in [315/4, inf[\\ b \in [0, 20] \\ 4 * a + b \geq 335 \end{cases}$

Coût continu Avec un coût continu, on peut avoir un coût qui dépend uniquement du temps pendant lequel le scooter est réellement disponible pour le client en donnant une dérivée du coût par rapport au temps égale par exemple à 500/365 lorsqu'il y a un jeton à la fois dans P_1 et dans P_4. Comme les dérivées du coût doivent être entière et que $500/365 = 100/73$ nous multiplions la valeur de coût souhaité (4000) par 73 et nous prenons : Timed_cost = P1*P4*100.

Nous obtenons en 69 s et avec 148 Mo le résultat suivant :

$\begin{cases} a \in]292/11, inf[\\ b \in [0, 105/11[\end{cases}$ *or* $\begin{cases} a \in]2920/99, inf[\\ b \in [0, 215/11[\end{cases}$ *or* $\begin{cases} a \in]365/11, inf[\\ b \in [0, 20] \end{cases}$

À noter que si l'on fait la même vérification en paramètre entier nous obtenons :

$\begin{cases} a \in [34, inf[\\ b \in [0, 20] \end{cases}$ *or* $\begin{cases} a \in [30, inf[\\ b \in [0, 19] \end{cases}$ *or* $\begin{cases} a \in [27, 30] \\ b \in [0, 9] \\ 5 * a + b \leq 154 \end{cases}$

Coûts continu et discret La durée de réparation en cas de panne est de 10, donc dans l'étape précédente avec un coût continu, le coût de la panne n'est que de 10. Nous pouvons mixer les coûts continus et discrets en ajoutant alors un coût discret de -40 (donc $-40 * 73$) à la transition `panne`.

Nous obtenons en 65 s et avec 151 Mo le résultat suivant : $\begin{cases} a \in]325/3, inf[\\ b \in [0, 20] \\ 3 * a + b > 345 \end{cases}$

Controle Nous souhaitons maintenant faire de la synthèse de stratégies, ici dans le but d'un gain final supérieur à 4000€ grâce à un modèle de scooter fiable et réparable. Nous revenons sur le modèle de la figure 4 mais en instanciant les paramètres a et b. Nous nous basons sur les valeurs $a = 85$ et $b = 20$ qui sont des valeurs acceptées dans le cas "Coût discret" précédemment étudié. Nous ajoutons un peu d'indéterminisme dans les durées. Nous considérons que les transitions `panne` et `entretien` sont incontrôlables, la première car n'étant pas un choix et la seconde à cause de la durée qui est soumise à des aléas. Ces transitions sont en pointillé dans la figure. De plus la vente peut être réalisée à n'importe quel cycle donc nous supprimons sa contrainte mais considérons qu'à 11 cycles avec moins de 4000€, la transition $T6$ met une fin définitive à l'expérience. Enfin nous imposons que `money` doit être positif. Le modèle est donné Figure 5.

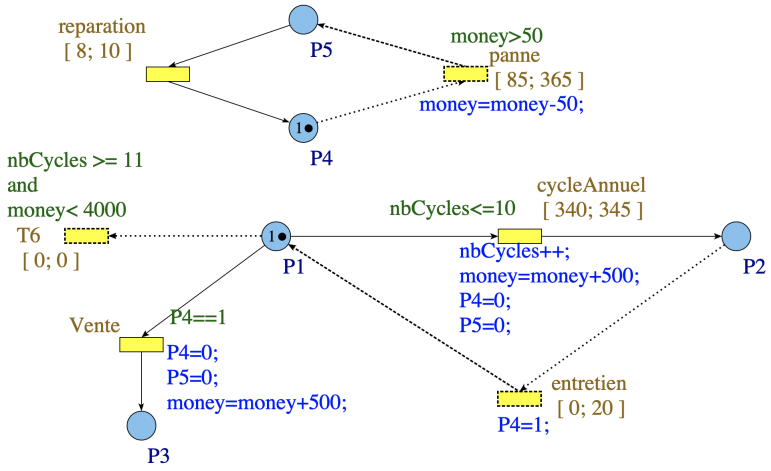

Figure 5: Modèle du scooter pour le contrôle

La propriété est `control (P3==1 and money >4000)`. Le résultat est qu'il existe une stratégie gagnante en ne jouant que sur les actions contrôlables pour atteindre l'objectif. Ce résultat est obtenu en $0.3s$ avec $6.7Mo$ utilisé.

Cette stratégie est donnée pour chaque état symbolique de l'espace d'état ce qui est trop volumineux pour être intégralement montré.

Nous illustrons cette stratégie sur deux états.

Etat 1 : Deux pannes se sont produites dans la première année. On obtient la stratégie :

`when P5=1 P4=0 P3=0 P2=0 P1=1 money=400 nbCycles=1`

- **with:** `cycleAnnuel` $\in$ `[340, 345] and reparation` $\in$ `[0, 10]`
  do `cycleAnnuel`
- **with:** `cycleAnnuel` $\in$ `[186, 340[ , reparation` $\in$ `[8, 10] ,`
  `reparation - cycleAnnuel` $\leq$ `-178`
  do `reparation`
- `otherwise wait`

Cette stratégie montre que lorsque le scooter tombe en panne à la fin de la période annuel il est plus intéressant de l'envoyer en entretien annuel que de faire seulement la réparation.

Etat 2 : money=3900 après 8 cycles On obtient la stratégie :

`when P5=0 P4=1 P3=0 P2=0 P1=1 money=3900 nbCycles=8`

- **with:** `cycleAnnuel` $\in$ `[0, 345] , panne` $\in$ `[0, 345] , Vente` $\in$ `[0, inf[ ,`
  `panne - cycleAnnuel = 0 , 0` $\leq$ `Vente - cycleAnnuel, 0` $\leq$ `Vente - panne`
  do `Vente`
- **with:** `cycleAnnuel` $\in$ `[340, 345] , panne` $\in$ `[0, 252] , Vente` $\in$ `[0, 252] ,`
  `panne - cycleAnnuel` $\leq$ `-93 , Vente - cycleAnnuel` $\leq$ `-93 , 0` $\leq$ `Vente - panne`
  do `cycleAnnuel`
- **with:** `cycleAnnuel` $\in$ `[93, 340[ , panne` $\in$ `[0, 247[ , Vente` $\in$ `[0, 247[ ,`
  `panne - cycleAnnuel` $\leq$ `-93 , Vente - cycleAnnuel` $\leq$ `-93 , 0` $\leq$ `Vente - panne`
  do `Vente`

La vente fait partie de la stratégie mais refaire un cycle annuel est aussi proposé car la vente restera possible en respectant l'objectif après un cycle supplémentaire. Cela montre que la stratégie générée ne cherche pas l'optimalité en temps.

5 Modélisation d'un système de régénération

Considérons maintenant le système de la figure 6. Il s'agit d'un réseau de Petri coloré, avec trois couleurs représentant chacune un lot de pièces.

Initialement, seule la place `User` est marquée avec trois jetons de couleurs différentes.

Les transitions `very_bad`, `bad`, et `medium` représentent le niveau de dégradation du lot que l'on recevra pour la régénération. Elles sont sensibilisées séparément par chacune des couleurs. Lors du tir d'une transition, la variable `$any` représente le numéro de la couleur pour laquelle la transition tire.

Le tableau health, indexé par les numéros des couleurs, représente l'état de santé des pièces des différents lots. Le tableau `repairs` le nombre de réparations autorisées pour ce lot (ici, maximum deux fois).

Pour les lots dans la place `Damaged`, il faut choisir entre démanteler qui met effectivement fin à la vie du produit, ou réparer qui permet de le remettre en circulation en bon état.

On ne peut ici réparer ou démanteler que deux lots en même temps et la réparation n'est possible que si le lot n'est pas trop détérioré : ici, elle n'est pas possible après `very_bad`.

La fonction `size` permet d'avoir le nombre de jetons (de n'importe quelle couleur) dans une place; la fonction `move` déplace tous les jetons de son deuxième argument vers son premier.

Nous associons un coût à l'exécution de ce système: premièrement, un coût par unité de temps égal à 10 fois le nombre de jetons dans `ToBeDismantled` plus 20 fois le nombre de jetons dans `Damaged` et `ToBeRepaired`. Ce coût modélise à la fois la dégradation des produits en attente de traitement et le coût de gestion des stocks.

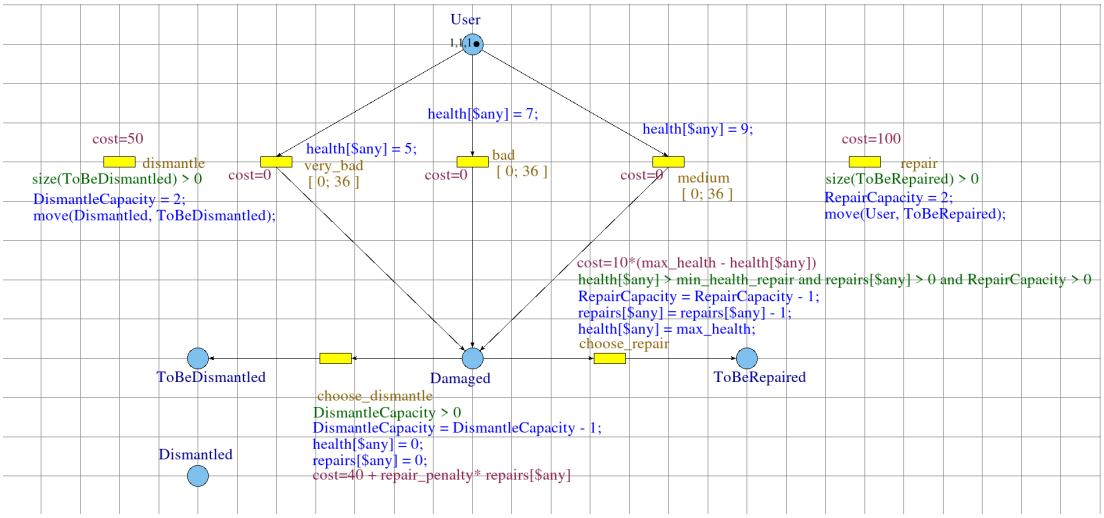

Figure 6: Réseau de Petri pour le système de régénération

Deuxièmement un coût discret, lié à la réparation de 10 fois la différence entre l'état de santé du produit et son maximum (ici 10). Chaque lot qui part en démantèlement engendre également un coût fixe de 40 plus une pénalité qui nous allons faire varier et qui dépend du nombre de fois que le lot pouvait encore être réparé.

Enfin, un coût fixe est associé aux transitions `repair` et `dismantle`, indépendant du nombre de lots traités.

L'objectif est de minimiser le coût jusqu'à ce que tous les lots soient démantelés. Les solutions varient entre tout démanteler immédiatemment et toujours tout réparer tant que c'est possible puis finalement démanteler. Dans Roméo, la propriété correspondante est `mincost (size(Dismantled) == _romeo_colors)` où `_romeo_colors` est une constante prédéfinie qui donne le nombre de couleurs dans le modèle (ici 3).

Avec la valeur 0 pour `repair_penalty`, on trouve un coût de 220 et la stratégie observée est de toujours démanteler immédiatemment. Ce comportement persiste en augmentant la valeur de la pénalité jusqu'à 60, le coût optimal étant alors 580.

À 61, le coût optimal est toujours 580 mais la stratégie optimale bascule à toujours tout réparer tant que c'est possible. Bien entendu cette stratégie reste la même pour des valeurs plus grande de la pénalité, le coût optimal restant le même (580) puisque avec cette stratégie, on démantèle quand toutes les réparations ont été faites, ce qui annule le terme de pénalité.

On peut observer que dans la stratégie optimale proposée avec une pénalité supérieure à 60 c'est toujours la transition `medium` qui est choisie avant réparation, puisque c'est celle qui va minimiser le coût de la réparation.

On peut néanmoins modéliser un pire cas en vérifiant une propriété de contrôle optimal, avec les transitions depuis `User` qui sont incontrôlables. Nous avons implémenté l'approche de [6] et nous vérifions la propriété `control (size(Dismantled) == _romeo_colors)`.

Avec une pénalité de 0 nous obtenons un coût optimal de 270, qui correspond à démanteler les pièces une par une, et coûte donc $40 + 50 = 90$ pour chacune d'elle, car maintenant l'environnement peut (i) toujours prendre la transition `very_bad` qui ne permet pas la réparation et (ii) faire en sorte que les lots arrivent avec une délai de 18 unités de temps entre chaque dans `Damaged`. Si on devait attendre, pour démanteler deux lots à la fois et gagner 50, cela coûterait

donc $18 \times 20 = 360$, ce qui n'est pas avantageux.

La pénalité ne permet clairement pas de changer ce comportement, et pour permettre la réparation, nous supprimons la transition `very_bad`. Avec une pénalité de 0, le comportement est inchangé car le coût de réparation est plus important que le démantèlement immédiat.

Il faut augmenter la pénalité à 130 pour que la stratégie optimale consiste à tout réparer le plus possible, avec un coût total de 1050, qui correspond aux 270 précédents pour tout démanteler au final, plus $(30+100)$ pour chacune des deux réparations de chacun des trois lots. La valeur 30 correspond à la réparation après la transition `Bad` $(10 \times (10-7))$, et non plus `medium`; et comme précédemment chaque lot est réparé ou démantelé séparément.

La stratégie fournie par Roméo, faisant de l'ordre de 23000 lignes, nous avons inféré ces stratégies à partir du modèle et de la valeur optimale fournie par le logiciel mais des techniques existent pour rendre ces stratégies plus exploitables [1].

Performances Sur une machine avec un processeur de type intel i7 à 2.7GHz et 16Go de RAM, la vérification des propriétés `mincost` prend de l'ordre d'une seconde et 12Mo de mémoire.

Le contrôle n'avait pas fini au bout de 10 minutes, en utilisant plus de 10Go de RAM.

Le système ayant une certaine symétrie au niveau des couleurs, nous avons donc implémenté la sémantique *First Enabled First Fired* (FEFF) introduite dans [5] pour les TPNs avec multi-sensibilisation. Elle consiste à ne rendre tirable que l'instance de chaque transition multi-sensibilisée qui a été sensibilisée en premier, reduisant drastiquement les possibilités. Cette sémantique simule néanmoins la sémantique prenant en compte toutes les possibilités de tirs [5] et préserve donc les propriétés qui nous intéressent. Ici la multi-sensibilisation des transitions sort légèrement du cadre de [5] dans lequel les jetons n'étaient pas colorés, mais l'adaptation du résultat est directe pour peu que la sensibilisation des transitions ne dépende pas d'une valeur particulière des couleurs, ce qui est le cas ici. Avec la sémantique FEFF, les propriétés `mincost` ne prennent plus que de l'ordre de 0.1s avec 1.5Mo de mémoire environ. Les propriétés de contrôle prennent jusqu'à 100s et 1.5Go de RAM. Ces résultats démontrent l'intérêt de cette sémantique en pratique.

6 Conclusion

Le travail présenté dans ce papier, visait à explorer l'apport des réseaux de Petri temporisés à coûts et à paramètres pour représenter et analyser des stratégies industrielles relevant de l'économie circulaire. Deux cas d'usage ont été étudiés : le premier dans le cadre d'un modèle d'économie de la fonctionnalité, le second dans un atelier de régénération. Chacun de ces cas a été volontairement limité à l'analyse d'un paramètre unique (durée de location et stratégie de maintenance ou coût), afin de poser les bases d'une démarche progressive.

Au-delà de la simple modélisation des scénarios, l'approche s'est appuyée sur deux apports du formalisme TPN : (i) la vérification formelle de propriétés temporelles, économiques ou structurelles (par exemple, atteindre un état sous certaines contraintes de coût ou de temps),(ii) la synthèse de stratégies ou de contrôleurs permettant de déterminer automatiquement des configurations de paramètres garantissant la satisfaction de ces propriétés.

Ces résultats montrent que les TPN offrent un cadre structurant pour explorer différents choix industriels sous contraintes, et qu'ils permettent non seulement de simuler mais aussi de guider la décision à travers des outils formels de vérification et de contrôle.

Ce travail constitue une première étape. À terme, il s'agira d'étendre cette approche à la prise en compte simultanée de plusieurs paramètres interdépendants, d'introduire des incertitudes sur les flux ou les états. L'application à des systèmes circulaires multi-acteurs ou territori-

alisés représente également une perspective riche, non seulement pour accompagner la transition vers des modèles plus durables, mais aussi pour faire évoluer les approches de modélisation, de vérification et de synthèse autour des réseaux de Petri, en stimulant le développement de nouveaux outils et méthodes adaptés à des systèmes dynamiques, incertains et multicritères.

References

[1] P. Ashok, J. Kretínský, K. G. Larsen, A. L. Coënt, J. H. Taankvist, and M. Weininger. SOS: safe, optimal and small strategies for hybrid markov decision processes. In *Quantitative Evaluation of Systems, 16th International Conference, QEST*, LNCS, Glasgow, UK, Sept. 2019. Springer.

[2] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE trans. on soft. eng.*, 17(3):259–273, 1991.

[3] H. Boucheneb, G. Gardey, and O. H. Roux. TCTL model checking of time Petri nets. *Journal of Logic and Computation*, 19(6):1509–1540, Dec. 2009.

[4] H. Boucheneb, D. Lime, B. Parquier, O. H. Roux, and C. Seidner. Optimal reachability in cost time Petri nets. In *15th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2017)*, LNCS, Berlin, Germany, Sept. 2017. Springer.

[5] H. Boucheneb, D. Lime, and O. H. Roux. On multi-enabledness in time Petri nets. In J. M. Colom and J. Desel, editors, *The 34th International Conference on Application and Theory of Petri Nets and other models of concurrency (Petri Nets 2013)*, LNCS, Milano, Italy, June 2013. Springer.

[6] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *FSTTCS'04*, volume 3328 of *LNCS*. Springer, 2004.

[7] I. Haur, J. Béchennec, and O. H. Roux. High-level Colored Time Petri Nets for true concurrency modeling in real-time software. In *International Conference on Control, Decision and Information Technologies (CODIT 2022)*, Istanbul, Turkey, May 2022.

[8] A. Jovanović, D. Lime, and O. H. Roux. Integer Parameter Synthesis for Real-Time Systems. *IEEE Transactions on Software Engineering (TSE)*, 41(5):445–461, 2015.

[9] A. Jovanović, D. Lime, and O. H. Roux. Control of Real-time Systems with Integer Parameters. *IEEE Transactions on Automatic Control*, 67(1):75–88, Jan. 2022.

[10] D. Lime, O. H. Roux, and C. Seidner. Cost problems for parametric time petri nets. *Fundamenta Informaticae*, 183(1-2):97–123, 2021.

[11] D. Lime, O. H. Roux, C. Seidner, and L.-M. Traonouez. Romeo: A parametric model-checker for Petri nets with stopwatches. In S. Kowalewski and A. Philippou, editors, *15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009)*, LNCS, York, United Kingdom, Mar. 2009. Springer.

[12] E. MacArthur, K. Zumwinkel, and M. R. Stuchtey. Growth within: a circular economy vision for a competitive europe. *Ellen MacArthur Foundation*, 100, 2015.

[13] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS'95*, volume 900 of *LNCS*, pages 229–242. Springer, 1995.

[14] A. Tukker. Product services for a resource-efficient and circular economy–a review. *Journal of cleaner production*, 97:76–91, 2015.

[15] G. Vanson, P. Marange, and E. Levrat. A technical and systematic characterization of circular strategy processes. In *IFIP International Conference on Product Lifecycle Management*, pages 3–13. Springer, 2023.