République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Aboubekr Belkaïd – Tlemcen
Faculté de Technologie
Département de Génie-électrique et électronique
Laboratoire MELT

THÈSE

Pour obtenir le diplôme de doctorat LMD

Spécialité : Génie Industriel

Présentée par :

BERRICHI Asma

TITRE :

Contrôle et ordonnancement d'un réseau d'alimentation de produits pétroliers basés sur les réseaux de Petri hybrides étendus

Soutenue le, 09-09-2024, devant le jury :

Président	Pr. DALI YOUCEF Malika	Univ. Abou-Bekr BELKAID – Tlemcen
Examinateurs	Pr. HAFFAF Hafid	Univ. Ahmed Ben-Bella – Oran 1
	Pr. KOULOUGHLI Sihem	Univ. Abou-Bekr BELKAID – Tlemcen
Directeurs de thèse	Pr. GHOMRI Latéfa	Univ. A. BELKAID, Tlemcen
	Pr. ALLA Hassane	Univ. Grenoble – Alpes
Invités	Dr. BENNACEUR Djamel	Ingénieur à l'entreprise NAFTAL
	Dr. Maliki Fouad	École sup. de sciences appliquées - Tlemcen

Résumé

Dans cette thèse, nous étudions le problème du transport de produits pétroliers (carburants) par des pipelines multi-produits. Nous nous concentrons particulièrement sur les problèmes de modélisation et de contrôle de ce mécanisme de transport. Ce domaine attire de plus en plus l'attention des scientifiques, de sorte qu'une grande quantité de recherches lui a été consacrée. La particularité du travail proposé ici est de considérer les réseaux de Petri hybrides pour la phase de modélisation et les automates hybrides pour la phase de contrôle. Nous nous focalisons sur un cas réel d'un système pipeline multi-produit situé en Algérie, qui approvisionne la région ouest en différents dérivés pétroliers. Nous développons un algorithme de synthèse de contrôleur en temps réel combiné à une programmation linéaire en nombre entier mixte (MILP) pour répondre efficacement aux exigences de notre système.
Pour valider les résultats obtenus, nous comparerons les résultats de contrôle avec ceux de l'optimisation par l'approche MILP réalisée dans nos travaux antérieurs.

Mots-clés
Pipelines multi-produits, Modélisation et contrôle, Réseaux de Petri hybrides, Automates hybrides, Programmation Linéaire en nombre entier mixte (MILP).

Abstract

In this paper, we study the problem of transporting oil products (fuels) using multiproduct pipelines. More especially, we consider the modeling and control problems of this transport mechanism.
This field attracts more and more the attention of scientists therefore a lot of research has been devoted to this subject. The particularity of the work proposed here is to consider the hybrid Petri nets for the modeling phase and the hybrid automata for the control phase. We focus on a real case of a multiproduct pipeline system located in Algeria that supplies the western region with different oil derivatives.
We develop a real-time controller synthesis algorithm combined with a mixed integer linear pogramming (MILP) formulation to respond efficiently to the requirements of our system.
To validate the results obtained, we will compare the control results with those of the optimization by the MILP approach achieved in our previous work.

Keyword
Multiproduct Pipelines, Modeling and Control Problems, Hybrid Petri Nets, Hybrid Automata, mixed integer linear pogramming (MILP).

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les pipelines sont d'immenses installations, permettent le transport de produits, qu'ils soient liquides ou gazeux, sur de longues distances de manière sécurisée. Dans le cadre de notre étude, nous nous intéressons aux réseaux de pipelines transportant du pétrole brut et ses produits dérivés depuis une raffinerie ou une source, vers des centres de stockage et de distribution. Il s'agit donc d'oléoducs, mais nous les désignerons de manière générale sous le terme de pipelines

Étant donné que l'investissement initial pour un pipeline est très coûteux, et qu'il engendre des frais d'exploitation tels que l'énergie consommée par les stations de pompage ainsi que les frais d'entretien, un seul pipeline est souvent utilisé pour transporter plusieurs types de carburants. Par ailleurs, comme le fonctionnement d'un pipeline nécessite qu'il soit constamment sous pression, l'injection de ces produits se fait successivement sans séparation physique entre eux, grâce à une technique appelée "batching". Ce type de pipeline est appelé "pipeline multi-produit".

Bien que ce type de pipeline offre de nombreux avantages en termes de coûts d'utilisation, il présente un important un inconvénient majeur. Une zone de mélange appelée contaminât ou interface se forme entre deux produits en contact dans le pipeline. Ces produits circulent séquentiellement dans le pipeline, ce qui fait que les longues séquences de plusieurs lots de produits (batches) engendrent de nombreuses interfaces.

Plusieurs investigations et études sur les pipelines ont été introduites dans différents domaines tels que la sécurité, la rhéologie, la régulation des différents paramètres liés à ce réseau par les automates programmables, et enfin la planification et l'ordonnancement, que ce soit pour les processus de production en établissant des calendriers au sein de la raffinerie, ou en optimisant le processus de transport pour minimiser le nombre d'interfaces issues du transport successif des batches dans le pipeline tout en répondant aux demandes des clients. Dans le cadre de ce travail, nous nous intéressons à ce dernier aspect, à savoir le transport via le pipeline multi-produit d'une raffinerie vers les dépôts.

La planification d'un pipeline multi-produit est une tâche cruciale et complexe dans l'industrie pétrolière. Plusieurs approches ont été développées pour résoudre ce problème de manière optimale, telles que la simulation, la programmation en nombres entiers mixtes (MILP et MINLP), des heuristiques et des métaheuristiques. Ces méthodes d'optimisation se focalisent sur la modélisation mathématique du fonctionnement de pipeline multi-produit en utilisant des

variables de décision binaires et continues. L'avantage est qu'elles permettent de travailler sur des réseaux complexes. Cependant, d'un autre côté, le planning optimal n'est pas assez flexible face aux changements ou aux conditions imprévues. Si l'on prend, par exemple, la commutation de produits, passant de l'injection du produit A au produit B, le programme indique un instant précis. Et si des situations nécessitent l'arrêt du pipeline, comme des opérations de maintenance ou autres, elles doivent être programmées à l'avance et intégrées dans le modèle.

Face à ces limites, nous avons choisi dans notre étude de nous orienter vers la commande optimale de ce système et notre approche sera appliquée sur un cas réel : le multi-produit pipeline ASR. Cette approche nous a permis de déduire des intervalles optimaux pour la commutation entre les évènements discrets. Nous nous basons sur la modélisation par les réseaux de Petri hybrides (RdPH) qui offrent une représentation fidèle du comportement du système, qu'il s'agisse d'événements discrets ou de dynamiques continues.

Dans un système de pipeline, l'écoulement des fluides correspond à une dynamique continue d'un système positif, et la décision de transporter chaque produit (ouverture et fermeture des électrovannes) correspond à des événements discrets. Les réseaux de Petri hybrides peuvent modéliser avantageusement ces systèmes. Cet outil fournit un modèle proche du système physique et décrit explicitement les aspects continus et discrets.

L'approche dans cette thèse concerne les réseaux de pipelines qui doivent fournir des produits pétroliers en quantité et à des dates prédéfinies posant ainsi un problème de contrôle sur la commutation des vannes d'alimentation. Nous avons utilisé les réseaux Petri hybrides étendus où on associe des intervalles de commutation aux transitons discrètes. Un même pipeline peut transporter des produits différents, entraînant un coût à chaque commutation et posant le problème de la meilleure séquence pour minimiser le contaminât.

Par la suite, après la modélisation, nous avons envisagé de coupler la puissance de la modélisation par les Réseaux de Petri à la capacité d'analyse des automates. Nous traduisons ainsi le modèle de réseau de Petri hybride, qui décrit le fonctionnement du pipeline multi-produit, en automate hybride. Cette étape permet de calculer formellement les espaces d'états atteignables. Cela va ensuite nous aider à synthétiser un contrôleur qui respecte les spécifications établies en utilisant une méthode axée sur la modification des gardes des transitions contrôlables de notre automate. Cette méthode s'inspire des travaux de Ghomri(Ghomri, 2012) , où elle a développé un calcul formel en modifiant la garde des transitions contrôlables. Ces formules algébriques permettent de quitter un sommet de l'automate avant que l'espace atteignable dans ce sommet n'atteigne des régions indésirables où les spécifications sont violées.

Dans le cadre de notre recherche, les exigences ou les spécifications agissent uniquement sur les variables d'état continues, représentées par les m_i. Une des particularités de notre modèle est que la partie discrète pilote la partie continue. Cependant, ce sont les variables discrètes qui vont définir et encadrer la dynamique du système. Nous nous concentrerons donc sur le

moment d'occurrence de l'événement discret contrôlable, symbolisé par des transitions discrètes temporelles, pour élaborer ce contrôleur.

Nous avons abordé deux méthodes de calcul de contrôleur. La différence entre elles se situe dans la manière de formuler le temps de séjour dans un sommet de l'automate atteignable. La première méthode se focalise sur les valeurs minimales et maximales des variables continues à l'entrée d'un sommet (l'espace d'état d'une variable m_i), qui sont calculées en se basant sur les conditions initiales et le temps de séjour dans les sommets prédécesseurs du sommet en question. Bien que nous ayons réussi à calculer un contrôleur pour notre système en utilisant cette méthode, nous avons rencontré des difficultés lors de son application en raison de la dépendance d'état dans notre système de pipeline multi-produit. Nous avons eu besoin, au fur et à mesure, de modifier l'espace atteignable dans les sommets prédécesseurs déjà explorés, ce qui nécessite un effort de calcul intensif et peut même limiter le fonctionnement autorisé du système.

Cette situation nous a poussés à envisager de travailler en temps réel, en considérant le temps de séjour comme une variable de décision. Ainsi, à chaque occurrence d'un événement discret, nous résolvons un programme linéaire en nombres entiers mixtes. Cependant, avant d'adopter cette approche, nous avons élaboré notre modèle mathématique en développant des contraintes sur l'espace atteignable des sommets afin de les contraindre à respecter les spécifications imposées à notre système. Cette méthode nous a également permis de sélectionner des trajectoires optimales qui minimisent le nombre d'interfaces.

L'approche de synthèse du contrôleur pour le système de pipeline multi-produit, peut se résumer en quatre étapes essentielles. La dernière étape se décompose également en trois démarches :

1. L’utilisation les RdPH D-élémentaire pour décrire le comportement du pipeline multi-produit ASR.

2. translation du modèle RdPH D-élémentaire en automate hybride.

3. la modélisation des spécifications de façon formelle.

4. L’analyse du modèle d'automate hybride avec PHAVer et déduire l'automate atteignable. Le contrôleur sera élaboré en utilisant ce modèle d'automate atteignable.

4.1. Développer des contraintes sur l'espace d'état atteignable.

4.2. Éliminer les chemins interdits.

4.3. Trouver l'intervalle de commande optimal de chaque sommet en résolvant un programme linéaire en nombres entiers mixtes (MILP), cette étape devant être réalisée en temps réel.

Cette thèse comprend quatre chapitres:

Dans le premier chapitre, nous décrivons le système de pipeline multi-produit. Une revue de la littérature est présentée pour situer notre recherche dans le contexte des travaux antérieurs dans le domaine des pipelines. Nous nous concentrons sur l'optimisation du transport à travers le pipeline et concluons par une description détaillée du pipeline multi-produit ASR, en illustrant les différentes spécifications imposées sur ce système.

Dans le deuxième chapitre, nous explorons les principes fondamentaux des systèmes dynamiques hybrides. Nous examinons également les outils de modélisation utilisés dans notre étude, à savoir les réseaux de Petri hybrides et les automates hybrides. Nous détaillons leurs caractéristiques et les diverses extensions de ces outils.

Dans le troisième chapitre, nous présentons le modèle de réseau de Petri hybride D-élémentaire décrivant le comportement du système multi-produit ASR. Nous traduisons ensuite ce modèle en un automate hybride linéaire pour une analyse plus formelle. Une analyse d'atteignabilité est effectuée à l'aide de l'outil PHAVer. À la fin nous déduisons le modèle automate atteignable pour notre système.

Enfin, le quatrième chapitre porte sur la conception du contrôleur. S'inspirant des travaux de Ghomri, nous y explorons deux méthodes distinctes pour la synthèse du pipeline multi-produit ASR, en veillant au respect des spécifications et à l'atteinte de l'objectif d'optimalité.

Chapitre 1

Les Pipelines Multi-Produits : Vue Générale

Dans ce premier chapitre, nous introduisons les pipelines multi-produits. Nous commençons par détailler le fonctionnement des divers composants du réseau de pipeline ainsi que les diverses configurations possibles pour ce type de réseau. Ensuite, nous offrons une revue des recherches antérieures liées aux pipelines pour contextualiser notre travail. Nous nous concentrons principalement sur la section de transport, mettant l'accent sur l'ordonnancement et l'optimisation du transport à travers le réseau. Nous décrivons en détail notre cas d'étude : le pipeline multi-produit ASR, en précisant les spécifications requises pour ce système afin d'atteindre les objectifs établis. Nous présentons comment le pipeline, dans notre étude de cas ASR, peut être défini comme un système dynamique hybride, justifiant ainsi notre méthodologie orientée vers la synthèse de contrôleurs de ce système. De plus, nous discutons notre travail antérieur sur ce réseau, basé sur l'optimisation de transport des carburants dans ce pipeline, en utilisant l'approche MILP. Cela nous aide ensuite à comparer ces résultats avec ceux obtenus par l'approche de synthèse de contrôleurs développée dans le cadre de ce travail.

1.1 Généralités sur les pipelines

Le pipeline est utilisé dans le monde entier depuis 1862. L'idée originale derrière un pipeline provient de Samuel Duncan Karns, un jeune ouvrier au sein du puits Phillips No2 en Pennsylvanie (Brice, 2003). Le tout premier pipeline était conçu en bois, et il servait à transporter de l'huile en exploitant la force de la gravité. Au fil du temps, cette idée novatrice a

évolué, donnant naissance à des systèmes plus sophistiqués, utilisant des tuyaux en acier et des stations de pompage.

Les pipelines sont un moyen de transport très efficace pour acheminer de grandes quantités de produits pétroliers vers des dépôts ou des stations éloignés. Ils offrent de nombreux avantages en termes de sécurité, étant considérés comme l'outil le plus fiable et économique pour transporter d'importants volumes de différents produits pétroliers vers des dépôts éloignés. Bien que la vitesse d'approvisionnement soit plus lente par rapport à d'autres modes de transport tels que les chemins de fer et les autoroutes, les pipelines sont économiquement avantageux, car ils permettent une réduction des frais de transport par rapport aux autres modes. De plus, ils offrent une fiabilité et une sécurité supérieures, ainsi qu'un impact écologique moindre en raison de leur résistance aux changements climatiques. Cela explique pourquoi les pipelines sont largement utilisés dans le monde entier. Selon Energy Information Administration en 2019, les pipelines représentaient 65% du transport utilisé pour le pétrole et ses dérivés, tandis que les chemins de fer représentaient 20%, les camions citernes 13% et les barges 2%.

1.1.1 Définition du réseau de transport par pipeline

Le réseau de pipelines est composé d'une canalisation et de plusieurs stations qui garantissent l'approvisionnement et la distribution des carburants. Le processus de transport des produits via le pipeline commence par la partie après l'extraction du pétrole et sa transformation en produits dérivés, puis se poursuit jusqu'à la livraison. Les zones opérationnelles sont réparties le long de la canalisation pour assurer le transport, la distribution et le stockage des produits pétroliers. Parmi les principales stations en fonctionnement, on peut citer ce qui suit :

a) Les stations d'injection : La raffinerie

La raffinerie est le point de départ du pipeline où le pétrole est transformé en carburants tels que l'essence, le gasoil, etc., grâce à un processus complexe comprenant la distillation, le craquage et le raffinage. On l'appelle également une station d'injection, car une fois les produits transformés et stockés, ils doivent être injectés dans le pipeline pour être transportés vers les zones de stockage et de distribution à l'aide des stations de pompage situées à l'intérieur de la raffinerie.

b) Le pipeline ou la canalisation

Les canalisations sont formées de gros tubes utilisés pour transporter du carburant sur de longues distances. Elles sont généralement composées de tuyaux d'acier qui sont soudés

ensemble. Après leur assemblage, ces tuyaux sont souvent enterrés et soumis à un processus de détection des défauts et de protection pour réduire la corrosion le moins possible.

c) Stations de pompage

Ce sont des installations localisées à des emplacements spécifiques le long du pipeline, connues sous le nom de stations. Elles sont principalement composées de pompes et de moteurs qui garantissent que la pression, la vitesse et le débit des produits pétroliers circulant dans le pipeline respectent les normes requises.

d) Les centres de distribution et de stockage (Dépôts)

C'est un lieu dédié au stockage et à la distribution des carburants. Le stockage est effectué dans de grands réservoirs (bacs de stockage) spécialement conçus pour maintenir chaque type de carburant en bon état. Il existe deux méthodes de distribution: de pipeline à pipeline et de pipeline au client final, la livraison étant généralement effectuée par des camions citernes. Chaque dépôt comprend différents départements tels que la sécurité, la maintenance, le contrôle qualité, le commercial, la logistique et les opérations.

1.1.2 Différentes Formes de Réseaux de Pipeline et de canalisations

1.1.2.1 Selon la forme

a- Les pipelines droits unidirectionnels et bidirectionnels

Dans un pipeline droit, les segments sont reliés sans courbures ni angles importants, ce qui en fait la méthode la plus efficace pour le transport des fluides. Cette configuration maintient une pression constante et un débit élevé, car il y a moins de restrictions liées aux changements de direction. En général, les produits sont transférés des raffineries vers des dépôts via des pipelines droits et unidirectionnels, comme l'exemple illustré dans la figure 1-1. Cependant, il peut arriver que certaines situations nécessitent l'utilisation de pipelines bidirectionnels pour le transfert des produits dans deux directions différentes. Dans ce cas, le nombre d'interfaces (c'est-à-dire le mélange entre deux produits adjacents dans le pipeline multi-produit) est très élevé, ce qui limite l'utilisation de ce type de pipeline à des besoins spécifiques importants (Yongtu et al., 2012).

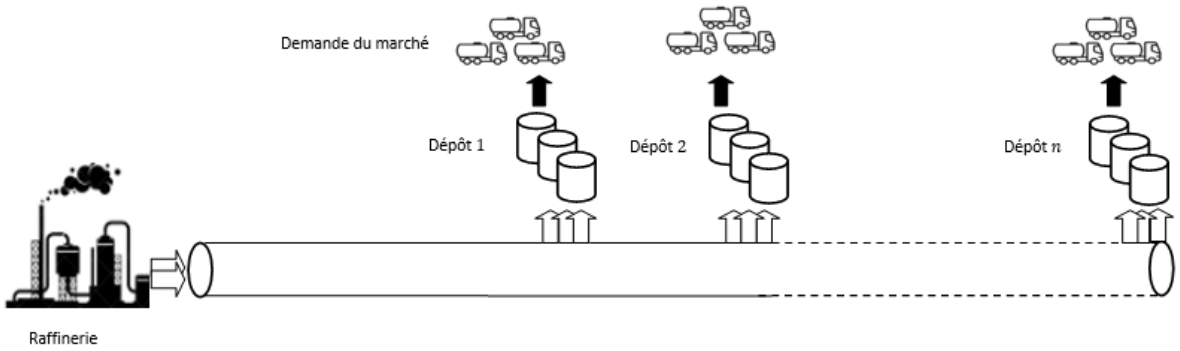


Figure 1-1 Réseau de pipeline droit et unidirectionnel

b- Pipeline arborescent

Dans ce type de réseau le pipeline se divise en plusieurs conduits plus petits qui sont connectés à la canalisation principale, qui a un diamètre plus large que les pipelines latéraux. Cette configuration crée un réseau en forme d'arbre avec plusieurs branches. Ce type de pipeline est conçu pour réduire les coûts d'investissement en raccourcissant la distance nécessaire pour acheminer les produits pétroliers vers différents centres de distribution ou raffineries. Ce système est flexible et peut répondre efficacement aux demandes des clients. La figure 2-1 donne un aperçu de la structure arborescente du pipeline.

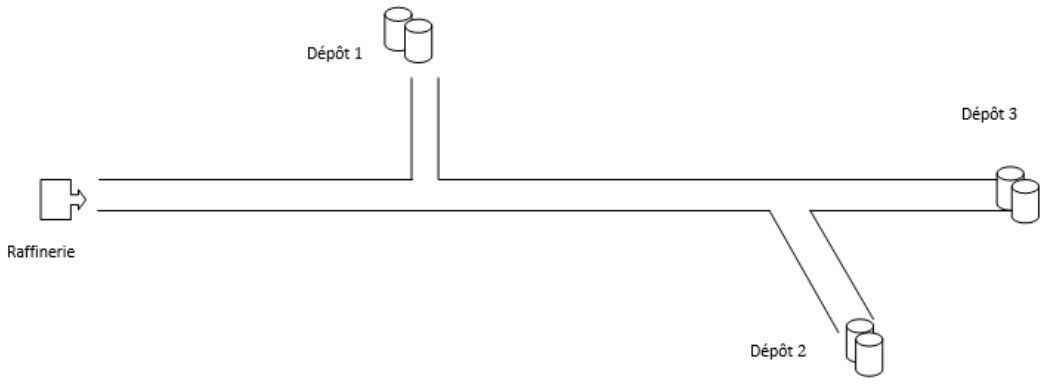


Figure 1-2 Un pipeline arborescent

c- Pipeline a structure maillée

Le pipeline à structure maillée représente la forme la plus complexe parmi les différentes configurations des pipelines existantes. Il relie plusieurs réseaux de pipelines les uns aux autres pour former un réseau plus vaste et plus flexible. Les pipelines sont connectés entre eux à l'aide de vannes et de stations de pompage, ce qui permet la redirection des produits vers différentes destinations. Cette structure est mise en place lorsqu'il existe plusieurs sources et plusieurs centres de distribution, ce qui permet d'optimiser la flexibilité du réseau. Même en cas de blocage ou de panne dans une section du pipeline, cette configuration permet une continuité du flux. Cependant, il convient de noter que cette méthode est également la plus coûteuse en termes d'entretien.

1.1.2.2 Selon le fonctionnement

a) Pipeline simple

Dans un système de pipeline simple, il est interdit de transporter simultanément différents types de produits. Le réseau de transport relie une seule raffinerie à un seul dépôt. Son but est d'acheminer un seul type de produit, depuis la raffinerie jusqu'à un centre de stockage et de distribution avec des canalisations exclusivement dédiées au transport d'un produit pétrolier spécifique, comme le gasoil par exemple. Ce genre de système est généralement utilisé pour les dépôts situés plus ou moins loin de la raffinerie, répondant à une demande significative pour un carburant spécifique préalablement déterminé.

b) Pipeline multi-produit

Le pipeline multi-produit permet le transport simultané de plusieurs types de carburant dans une même conduite qui alimente un ou plusieurs dépôts. L'avantage de ce système réside dans sa capacité à réduire les coûts d'investissement initiaux en permettant le transport de divers produits dans un même pipeline, au lieu de le réserver exclusivement à un seul type de carburant.

Un inconvénient majeur est la formation d'une zone de mélange indésirable entre les produits qui circulent successivement dans le pipeline. Ce mélange est considéré comme une perte, car il n'est pas commercialisable. Certaines entreprises utilisent une séparation physique entre les produits, mais cela entraîne des coûts d'entretien importants et peut parfois être inefficace en raison des différentes caractéristiques chimiques, telles que la densité des produits, ainsi que des variations de température et de pression qui impactent cette séparation au niveau du pipeline.

Cette zone de mélange apparaît entre deux produits adjacents, ce qui entraîne des coûts supplémentaires, car elle doit être stockée séparément dans des bacs en fonction de la nature des produits mélangés et renvoyée à la raffinerie pour être traitée. La quantité de ce mélange

augmente lorsque le nombre de lots de produits envoyés dans une courte durée augmente. C'est pourquoi il est nécessaire d'avoir une planification d'approvisionnement optimale dans cette situation.

1.1.3 Le fonctionnement du système pipeline multi-produit

Les pipelines sont des réseaux utilisés pour transporter des marchandises sous forme liquide ou gazeuse, depuis le point de réception jusqu'au point de livraison. Les systèmes de pipelines sont composés principalement de zones opérationnelles et de segments de pipeline. Les zones opérationnelles peuvent être des centres de distribution, des ports ou des raffineries. Il existe trois types principaux de zones opérationnelles : les stations d'injection situées généralement à l'entrée du système de pipeline au niveau des raffineries, les stations de pompage qui sont régulièrement réparties le long du système pour maintenir la pression et la vitesse du produit dans le pipeline, et enfin, les stations de livraison qui correspondent aux centres de stockage et distribution (dépôts).

Le processus d'approvisionnement en produits pétroliers à travers un pipeline multi-produit prend en compte toute la chaîne de transport, depuis la raffinerie jusqu'aux clients. Une raffinerie fournit des produits pétroliers qui doivent être transportés aux clients via un pipeline. Le pipeline est divisé en segments avec un dépôt à la fin de chaque segment. Les différents produits dans chaque dépôt sont stockés et distribués pour répondre aux demandes des clients. Les pipelines multi-produits permettent de transporter plusieurs produits simultanément dans une même ligne, augmentant ainsi le taux d'utilisation du pipeline et réduisant les coûts d'investissement. Cependant, ils ont également l'inconvénient que les produits entrent en contact direct à l'intérieur du pipeline multi-produit, ce qui crée une zone de mélange entre deux produits successifs (appelée interface ou contaminât).

Il existe deux différents types de livraison, la livraison à un seul dépôt à la fois ou la livraison simultanée à plusieurs dépôts. Cette dernière alternative permet d'éviter les interruptions et les commutations fréquentes des pompes, ce qui économise à la fois l'énergie, notamment la puissance réactive des pompes, et les frais d'entretien.

1.1.3.1 Le pipeline multi-produit et le concept de contaminât

- **Le batching**

Le batching est une technique de pompage séquentiel de consistant à expédier successivement deux ou plusieurs produits pétroliers différents dans un ordre spécifique via le même pipeline, la quantité de produit spécifique injectée à la fois est appelée lot ou batch. Cette pratique est souvent utilisée lorsque les volumes de produits transportés ne justifient pas la construction de pipelines distincts pour chaque produit. Elle permet notamment d'optimiser l'utilisation des

pipelines, de réduire les coûts d'investissement et surtout d'améliorer la flexibilité du réseau (Maire, 2011).

- **Le Contaminât ou interface :**

Lorsque différents produits sont injectés successivement, ils entrent en contact direct à l'intérieur de la canalisation multi-produit, provoquant forcément une dégradation de la qualité du produit dans la région de mélange (l'interface entre deux produits) comme le montre dans la figure 1-3. Ces pertes d'interface posent de grands problèmes opérationnels. Les mélanges ne peuvent pas simplement être reclassés, ils doivent subir des analyses physico-chimiques spéciales qui nécessitent généralement d'être renvoyés à une raffinerie ou stockés dans des réservoirs spéciaux. Tout cela entraîne des coûts supplémentaires pour le stockage, le transport et le traitement.

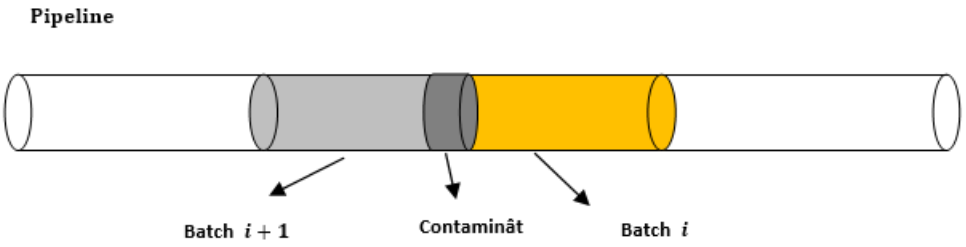


Figure 1-3 Pipeline : deux batches avec une zone de mélange

1.2 Études et recherches sur les pipelines multi-produits

L'étude des pipelines a suscité l'intérêt de nombreux chercheurs, que ce soit dans le domaine de la sécurité ou d'autres domaines. Selon la littérature, on peut les classer en quatre domaines principaux :

Le premier domaine concerne la sécurité des pipelines, où des efforts importants sont déployés pour étudier la sûreté et la sécurité, car les pipelines peuvent être très vulnérables aux accidents. Une analyse systématique de la sûreté et de la sécurité des pipelines est présentée dans (C. Chen et al., 2021). Par conséquent, de nouvelles méthodes ont été proposées pour évaluer et prévenir les défauts, les pertes et détecter les fissures lors de l'utilisation du pipeline à long terme (C. Chen et al., 2021; Priyanka & Thangavel, 2020).

Le deuxième domaine est la rhéologie, une étude de la littérature dans ce domaine est réalisée par (Martínez-Palou et al., 2011). L'objectif principal de plusieurs types de recherche dans ce domaine était de présenter des méthodes pour réduire la viscosité afin de prévenir la corrosion et d'optimiser l'efficacité des stations de pompage (Souas et al., 2021; Wang et al., 2020).

La troisième catégorie est la planification et l'ordonnancement du système de pipeline. Cela est considéré comme un élément crucial de la chaîne d'approvisionnement en carburants, que ce soit dans le cadre de la planification des programmes de production en raffinerie ou pour optimiser les pertes lors du transport tout en répondant efficacement à la demande du marché (Li et al., 2021; Pinto et al., 2000).

L'ordonnancement des pipelines est généralement établi de manière empirique, en se basant sur des prédictions. Les travaux dans ce domaine ont attiré l'attention de plusieurs chercheurs, dont l'objectif était d'optimiser le transport par pipeline. Ils ont introduit des approches de programmation linéaire en nombre entier mixte (MILP) et de programmation non linéaire en nombre entier mixte (MINLP)(Berrichi et al., 2021; Cafaro et al., 2012; Ghaffari-Hadigheh & Mostafaei, 2015), ainsi que l'utilisation des heuristiques et des métaheuristiques pour trouver un planning optimal qui satisfait les exigences du marché et les contraintes de fonctionnement du pipeline.

Le dernier domaine concerne la commande, où les travaux se concentrent sur la régulation de différents paramètres tels que la température, la pression, le débit, etc., afin d'assurer la sécurité du pipeline (Priyanka et al., 2018, 2019). De nouvelles méthodes et algorithmes sont implémentés dans les automates programmables industriels pour garantir l'efficacité et la robustesse du contrôle du pipeline(Możaryn et al., 2021).

Dans notre travail, nous nous intéressons particulièrement à la troisième catégorie qui concerne la planification et l'ordonnancement du système de pipelines, plus précisément dans la partie transport.

1.2.1 Planification d'un pipeline multi-produit

Afin de garantir un approvisionnement sûr et de répondre aux besoins du marché des produits pétroliers, ainsi que de minimiser la contaminât causée par le nombre d'interfaces dépendant principalement du nombre de batches transporté dans le pipeline, il est essentiel d'établir une planification adéquate. Cette planification doit permettre d'identifier les séquences optimales de produits à transporter sur une période donnée, allant de la raffinerie aux dépôts. Elle vise à satisfaire les demandes tout en respectant les exigences et contraintes opérationnelles et en minimisant les pertes telles que la

contaminât. Le calendrier établi doit inclure tous les détails nécessaires pour le transport, tels que l'heure d'injection du produit, le débit d'injection, le volume du produit, l'heure de réception du produit, et le dépôt par lequel il sera reçu, etc.

1.2.2 Les différentes approches utilisées dans la planification d'un pipeline multi-produit

La planification d'un pipeline multi-produit est une tâche cruciale et complexe dans l'industrie pétrolière. Des méthodes variées ont été développées pour faire face à ce défi et garantir une gestion sécurisée et efficace du transport. Dans ce contexte, nous examinerons les différentes approches adoptées par les chercheurs dans ce domaine.

MILP, MINLP : La programmation linéaire et non linéaire en nombres entiers mixtes (MILP et MINLP) est la méthode privilégiée par les chercheurs pour la planification des pipelines. Dans cette approche, nous choisissons des variables binaires pour prendre des décisions telles que l'affectation des produits aux batches, ainsi que des variables continues comme la durée de pompage et le débit. L'objectif principal est de minimiser le contaminât et d'autres coûts tels que les coûts de pompage et de stockage.

Il existe deux méthodes de travail : l'approche continue et l'approche discrète (Castro & Mostafaei, 2019; Rejowski Jr & Pinto, 2003). Dans l'approche continue, les batches sont perçus comme l'ensemble principal au sein de notre modèle MILP ou MINLP, nous permettant de suivre le temps en fonction du volume injecté et de la vitesse d'écoulement. Par contre, l'approche discrète divise le pipeline en petits segments, tous identiques en volume et appelés "paquets", qui sont traités comme l'ensemble principal dans le modèle où nous attribuons les produits (figure 1-4, figure 1-5).

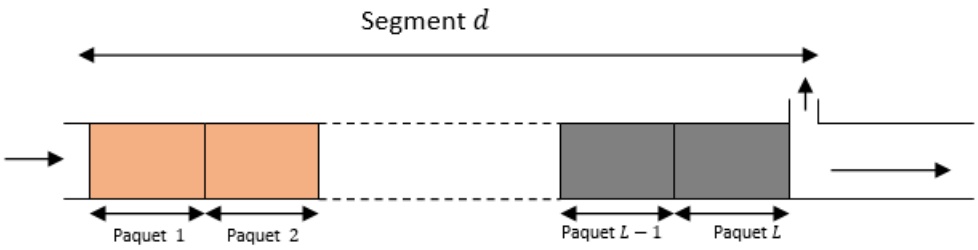


Figure 1-4 Un segment du pipeline dans l'approche discrète

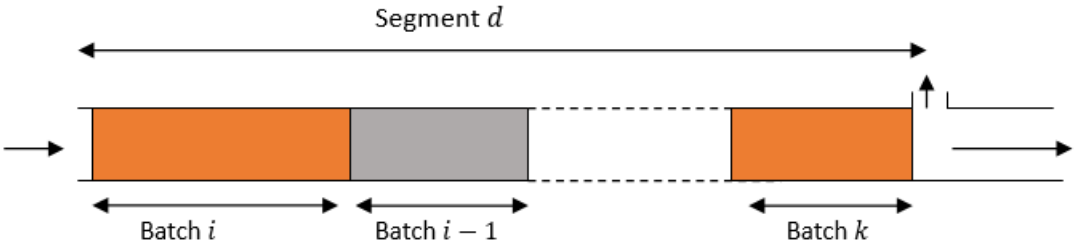


Figure 1-5 Un segment du pipeline dans l'approche continue

Le modèle peut être formulé soit comme un modèle linéaire en nombres entiers mixtes (MILP), soit comme un modèle non linéaire (MINLP), selon la formulation mathématique choisie. La résolution se fait à l'aide de solveurs tels que Lingo et Cplex, qui utilisent des heuristiques telles que comme branch and bound et cutting plans method. Ces heuristiques parcourent tout l'espace de calcul pour garantir un résultat optimal.

Cependant, si nous avons un système complexe avec plusieurs dépôts et une perspective à moyen ou long terme où de nombreuses variables sont impliquées, le temps de réponse augmente de manière exponentielle. Il devient alors difficile d'obtenir un résultat raisonnable dans un laps de temps donné.

LES HEURISTIQUES

D'autres auteurs ont introduit l'utilisation des heuristiques dans la planification des pipelines multi-produits (H. Chen et al., 2017; Liao, Zhang, Wang, et al., 2018). Ils ont exploité divers algorithmes existants tels que l'algorithme glouton (greedy algorithm), l'algorithme de priorité (priority algorithm), et l'algorithme de recherche en profondeur d'abord (depth-first search algorithm). Par ailleurs, ils ont développé de nouveaux algorithmes spécifiquement pour la planification des opérations de pipeline. Ces algorithmes sont capables de réduire efficacement l'espace des solutions et offrent une solution faisable avec une performance computationnelle abordable. Ainsi, même dans un système complexe à grande échelle, ils peuvent fournir des résultats dans un délai raisonnable par rapport aux solveurs. Cependant, bien que les heuristiques aient leurs avantages pour résoudre les problèmes complexes de planification des pipelines multi-produits à grande échelle, leur application peut varier selon le type de problème rencontré, ce qui limite l'universalité des algorithmes.

LES METAHEURISTIQUES :

Les métaheuristiques les plus couramment utilisées par les chercheurs pour aborder ce problème incluent l'optimisation par essaims de particules (PSO), l'algorithme génétique (GA),

et le recuit simulé (SA Algorithm)(Herrán et al., 2012; Khalili Goudarzi et al., 2021). L'application de ces algorithmes dans la planification du pipeline se divise en deux étapes principales :

Décomposition du problème : Initialement, la stratégie consiste à décomposer le problème initial en plusieurs sous-problèmes. Ces sous-problèmes sont ensuite résolus en utilisant les métaheuristiques mentionnées afin d'obtenir certaines variables clés, telles que les séquences des batches à injecter dans un horizon de temps donné, et les horaires d'arrivée des batches aux dépôts. En se basant sur ces variables, le reste du problème peut être résolu par des solveurs.

Utilisation du solveur : Après plusieurs itérations, jusqu'à ce que le modèle converge, on obtient la planification optimale. La fonction objective du modèle représente la “fitness” de l'algorithme métaheuristique utilisé. Ce résultat est ensuite utilisé pour modéliser un problème plus simple avec moins de variables.

Bien que les métaheuristiques puissent résoudre des problèmes à grande échelle, elles présentent l'inconvénient de ne pas garantir une planification optimale. Les solutions peuvent converger vers un minimum local plutôt qu'un minimum global, ce qui limite leur efficacité dans certains contextes.

Les méthodes basées sur les données (Data-Driven methods)

L’approche axée sur les données représente un progrès significatif dans la planification et l'ordonnancement d'un pipeline, car elle permet d'améliorer les méthodes de résolution de ce problème complexe. Cette méthode se base sur l'utilisation des données historiques, intégrant ainsi la science des données dans le processus de planification du pipeline (Haoran et al., 2018; Liao, Zhang, Xu, et al., 2018).

Si les schémas d'approvisionnement, les demandes et l'état initial du pipeline ne montrent pas de différences significatives, le nouveau planning à développer ne sera pas très différent du planning archivé. Si, par exemple, il y a une grande différence entre la solution initiale et la solution optimale dans l'utilisation d'une métaheuristique, le temps de résolution peut être allongé. Dans ce cas, le planning historique peut être utilisé comme solution initiale pour l'algorithme métaheuristique.

Donc en résumé en enregistrant chaque planning établi dans une base de données, il est possible de trouver le planning le plus similaire lors du développement d'un nouveau planning. En comparant les paramètres du modèle, ce planning peut ensuite être utilisé comme solution initiale pour l'algorithme métaheuristique. La création et l'utilisation d'une base de données jouent un rôle essentiel dans l'efficacité de cette méthode.

Elle permet ainsi d'accélérer la vitesse de convergence de l'algorithme et d'assurer de bons résultats. Comparée aux méthodes traditionnelles, elle utilise parfaitement l'historique du pipeline et réduit considérablement le temps de résolution.

La simulation

L'importance de valider un planning préétabli ne peut être sous-estimée, surtout dans le domaine du transport de carburant via des réseaux de pipelines maillés, où la structure est plus complexe. Dans des cas simples, où la durée est courte (2 à 4 jours), il est possible de valider un planning par des expériences pratiques. Cependant, pour des systèmes plus complexes ou des plannings s'étendant sur le moyen à long terme, les méthodes de simulation s'avèrent cruciales.

Prenons l'exemple de travail de l'auteur dans(Csontos et al., 2022), une nouvelle méthode de simulation basée sur des événements discrets est proposée. Cette approche est particulièrement adaptée pour valider des plannings sur des durées moyennes, en particulier pour des réseaux de pipelines multi sources à structure maillée.

1.3 Présentation du pipeline multi-produit ASR

1.3.1 Le Pipeline Multi-produit ASR : Objectif et spécifications

Le pipeline multi-produit ASR (l'abréviation d'Arzew, Sidi Bel Abbès, et Remchi) se situe à l'ouest de l'Algérie. S'étendant sur une longueur d'environ 168 km, il transporte les carburants depuis la Raffinerie d'Arzew vers les centres de stockage et de distribution de Sidi Bel Abbès (SBA) et Remchi. Ces derniers desservent les régions ouest et sud-ouest de l'Algérie en carburants. (Figure 1-6)

Figure 1-6 Profil en long du pipeline multi-produit ASR (Bennacer et al., 2016)

Le pipeline ASR transporte quatre produits pétroliers : gasoil, essence super, essence normale, et essence sans plomb via un pipeline multi-produit unidirectionnel. (Figure 1-7)

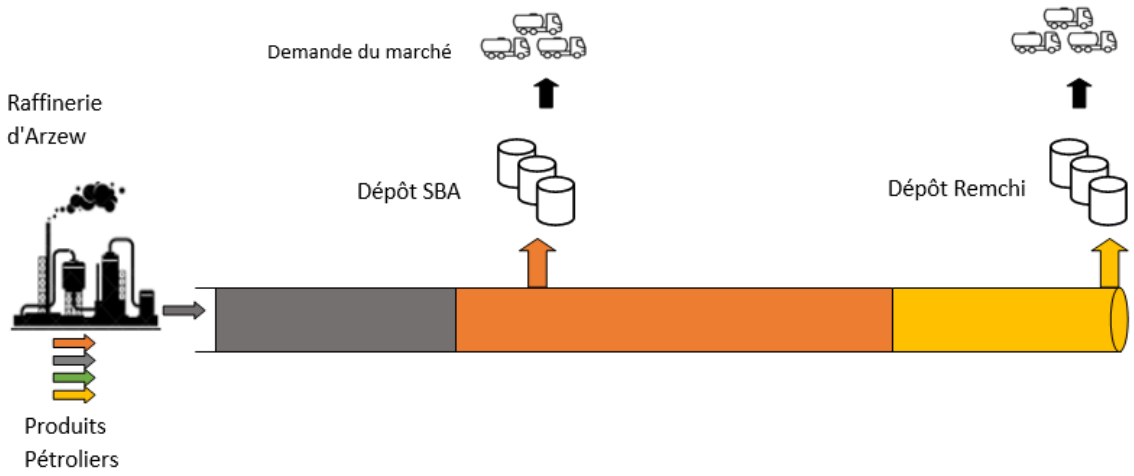


Figure 1-7 Pipeline multi-produit ASR

Dans le cadre de ce travail, nous envisageons d'effectuer une étude de cas réduite, en ne conservant que le dernier dépôt de Remchi, qui possède un important parc de stockage comparé au dépôt de Sidi Bel Abbès (SBA). Nous garderons également seulement deux

produits, le gasoil et l'essence super, qui sont les produits les plus consommés, et nous considérerons au maximum trois tronçons (batches) dans la canalisation à la fois. (Figure 1-8)

Nous avons opté pour un système simplifié afin de réduire la taille du modèle et de clarifier les étapes de l'approche de synthèse, notamment lors de la modélisation, de l'analyse et du calcul du contrôleur. Néanmoins, cette approche peut être étendue à des réseaux de pipelines plus élaborés.

L'objectif principal de la commande de ce système est de satisfaire les demandes des clients. Ainsi, nous devons assurer une autonomie de stock pour pallier toute rupture, tout en respectant la contrainte de capacité des produits dans les dépôts.

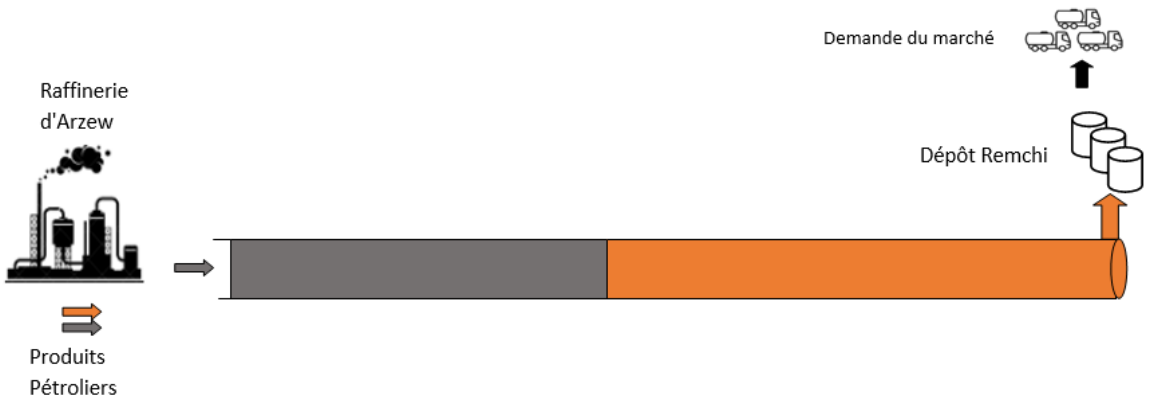


Figure 1-8 Une configuration simplifiée du pipeline multi-produit ASR

La figure 1-9 montre les différentes configurations possibles dans le pipeline multi-produit ASR (une source - un dépôt - et au maximum trois batches dans le pipeline).

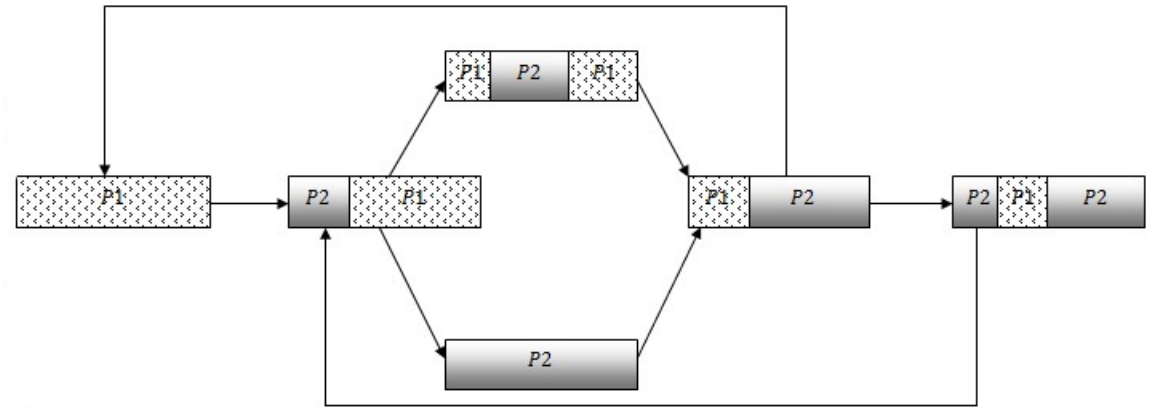


Figure 1-9 Les différentes configurations possibles dans le pipeline multi-produit ASR (une source - un dépôt - et au maximum trois batches dans le pipeline).

Le tableau suivant (table 1-1) récapitule les exigences en matière de demande journalière, de capacité et de stock de sécurité du dépôt pour chaque type de produit.

Produit	Capacité (m^3)	Stock de sécurité (m^3)	Niveau du stock à $t = 0$ (m^3)
Pr_1	22000	4400	5572
Pr_2	9500	1900	4350

Table 1-1 Capacité, stock de sécurité et niveau de stock initial dans le dépôt

Remarque1.1 les données que j'ai utilisées dans ce travail datent de l'année 2021

1.3.2 Optimisation du Transport de carburants dans le pipeline multi-produit ASR en utilisant l'approche MILP

Dans cette partie, nous présentons notre travail antérieur dédié à l'optimisation du pipeline multi-produit ASR (Berrichi et al., 2021). L'objectif était de planifier le transport des produits pétroliers en termes de quantité et de dates prédéfinies pour approvisionner efficacement les dépôts, tout en minimisant les pertes telles que le contaminât, en utilisant l'approche MILP. Le but est de proposer des solutions, c'est-à-dire des calendriers de transport optimisés, afin de pouvoir ensuite comparer ces résultats avec ceux obtenus par le contrôle. Notre article (Berrichi et al., 2021) détaille l'approche de résolution et le modèle mathématique complet du système pipeline ASR. Cependant, nous présentons ici le planning optimal pour une étude de cas réduite du pipeline ASR tel que défini dans la section précédente. La différence fondamentale entre les deux approches réside dans la manière dont les demandes journalières sont gérées : le modèle de Réseau de Petri (RdP) utilise un débit fixe pour répondre aux demandes, alors que le modèle de Programmation Linéaire en Nombres Entiers Mixtes (MILP) se base sur la demande cumulée jusqu'à l'injection du dernier batch, nécessitant que les demandes journalières pour chaque produit soient satisfaites avant la fin de la journée.

1.3.2.1 Problématique et modèle mathématique

Problématique

Le problème d'optimisation de transport des carburants par pipeline multi-produit considère tout le processus de transport depuis de la raffinerie jusqu'aux clients.

Une raffinerie fournit des produits pétroliers qui doivent être transportés aux clients par l'intermédiaire d'un pipeline, ce dernier est divisé en segments, un dépôt est mis à la fin de

chaque segment assurant ainsi le stockage et la distribution des différents produits, afin de satisfaire les demandes des clients.

Un pipeline doit être exploité de la meilleure façon possible, ceci revient à minimiser les coûts d'exploitation associés (les coûts d'interface entre les différents produit, les coûts de pompage…).

La minimisation des coûts cités précédemment nécessite la définition d'un programme permettant de spécifier :

- Le volume et le temps de début de pompage du chaque batch dans le pipeline.
- La distribution des batch dans les différents dépôts.
- L'endroit et l'ordre des différents batches dans le pipeline.
- Les niveaux d'inventaire de tous les réservoirs impliqués dans la raffinerie et les dépôts.

Tout en considérant les contraintes opérationnelles suivantes :

- Les capacités de stockage des produits dans les différents dépôts.
- la capacité de la canalisation et la cadence de fabrication maximum de la raffinerie.
- Un et un produit peut être pompé dans la canalisation à la fois.

L'hypothèse principale impose que le pipeline reste complètement plein à tout moment, et les produits sont incompressibles.

Modèle mathématique

Nous avons utilisé l'approche continue pour modéliser le problème, où nous avons considéré les batches comme un ensemble principal. L'approche continue signifie que nous ne divisons pas le pipeline en paquets, mais elle n'est pas continue en fonction du temps, car celui-ci est discrétisé. Ainsi, nous ne pouvons pas suivre et prévoir l'état du pipeline et du stock à chaque instant, mais plutôt à la fin de l'injection de chaque nouveau batch.

Chaque nouveau batch i qui est planifié pour être injecté dans le pipeline est représenté par les variables suivantes:

- Allocation du produit p au batch $\mathrm{Y}_{i,\mathrm{p}}$ (variable binaire) ;
- La taille initiale du batch (Qi);
- Le temps du pompage (L_i) ;
- Le temps du départ de l'injection (S_i);
- Le temps de fin de l'injection (S_i+L_i) ;
- La période du temps où l'injection du batch est terminé (ω (i, t) variable binaire).

Les ensembles:

$\boldsymbol{I}$	L'ensemble des batches ($I^{ancien} \cup I^{nouveau}$)
$\boldsymbol{I^{ancien}}$	L'ensemble des batches qui sont dans le pipeline à l'état initial (à $t = 0$)
$\boldsymbol{I^{nouveau}}$	L'ensemble des nouveaux batches qui seront injectés dans l'horizon de planification
$\boldsymbol{J}$	L'ensemble des dépôts
$\mathbf{P}$	L'ensemble des produits pétroliers
$\boldsymbol{T}$	L'ensemble des périodes du temps dans l'horizon de planification

La fonction objective

La fonction objectif consiste à minimiser le volume total du contaminât qui résulte entre les différents batches dans le pipeline. La variable de décision $inv_{i,p,p'}$ représente le volume d'interface entre le batch i et le batch $i+1$ contenant respectivement les produits p et p.

$$min \sum_{i \in I} \sum_{p \in P} \sum_{p' \in P} INV_{i,p,p'}$$

Contraintes

Dans cette section, l'accent est mis sur deux contraintes clés : le niveau de stock et la satisfaction des demandes journalières. Pour une compréhension approfondie des variables de décision et du modèle mathématique associé, je vous invite à consulter la publication de référence (Berrichi et al., 2021) pour des explications détaillées.

Contraintes sur le niveau de stock dans les dépôts

$$nvs_{p,j,i} = nvs_{p,j,i-1} + \sum_{h \in I, h \leq i} vmp_{h,p,j,i} - vom_{p,j,i}$$

$$\forall p \in P, j \in J, i \in I^{nouveau}$$

le niveau de stock du produit p dans le dépôt j à la fin du pompage du batch i est égale au niveau de stock de ce produit à la fin du pompage du batch précédent $i-1$ plus la somme du volume du produit p transféré au dépôt j durant l'injection du batch i moins la quantité livrés aux clients pendant le temps de pompage de ce batch ($vom_{p,j,i}$).

$vom_{p,j,i} \leq nvs_{p,j,i} \leq cap_{p,j} \forall \mathrm{p} \in \mathrm{P}, \mathrm{j} \in \mathrm{J}, \mathrm{i} \in I^{nouveau}$

Le niveau de stock du produit p dans le dépôt j doit être toujours inférieur ou égale à la capacité de ce produit dans ce dépôt et supérieure ou égale à le stock de sécurité.

Contrainte de la satisfaction des demandes journalières

$$\sum_{l=1, l \in I^{nouveau}}^{i} vom_{p,j,l} \geq \sum_{k=1, k \in T}^{t} dp_{p,j,k} * (\omega_{i,t} - \omega_{i+1,t})$$

$$\forall p \in P, j \in J, t \in T, \mathrm{i} \in I^{nouveau}$$

Supposons que le batch i est de dernier nouveau batch qui est injecté dans la période t donc $\omega_{i,t} = 1$ et $\omega_{i+1,t} = 0$,par conséquent la quantité du produit p qui est transféré à partir du dépôt j à l'ensembles des clients durant l'injection des nouveaux batches $(1,2,3, \ldots., i-1, i)$ doit être supérieure ou égale à la quantité demandé accumulé $(t1..t)$ du produit p.

1.3.2.2 Résultats et analyse

Notre étude de cas se concentre sur un segment spécifique du pipeline multi-produit ASR, reliant la raffinerie d'Arzew au centre de stockage et de distribution de Remchi. Ce segment approvisionne deux types de produits pétroliers : le gasoil P1 (GO) et l'essence super P2 (SCA). Les paramètres et les données utilisés dans notre modèle d'optimisation sont détaillés dans le tableau 1-1.

Lors de l'horizon d'optimisation, fixé à 384 heures, notre étude a déterminé la nécessité d'injecter quatre batches pour atteindre les objectifs fixés. En conséquence, la séquence optimale obtenue est:
$P2\ (9600\ m^3)\ - P1(43942\ m^3) - P2(4058\ m^3) - P2(19200\ m^3)$.
La figure suivante (figure 1-10) détaille le planning optimal établi pour notre période d'optimisation, incorporant tous les paramètres nécessaires à la gestion efficace des opérations d'injection et de réception des produits.

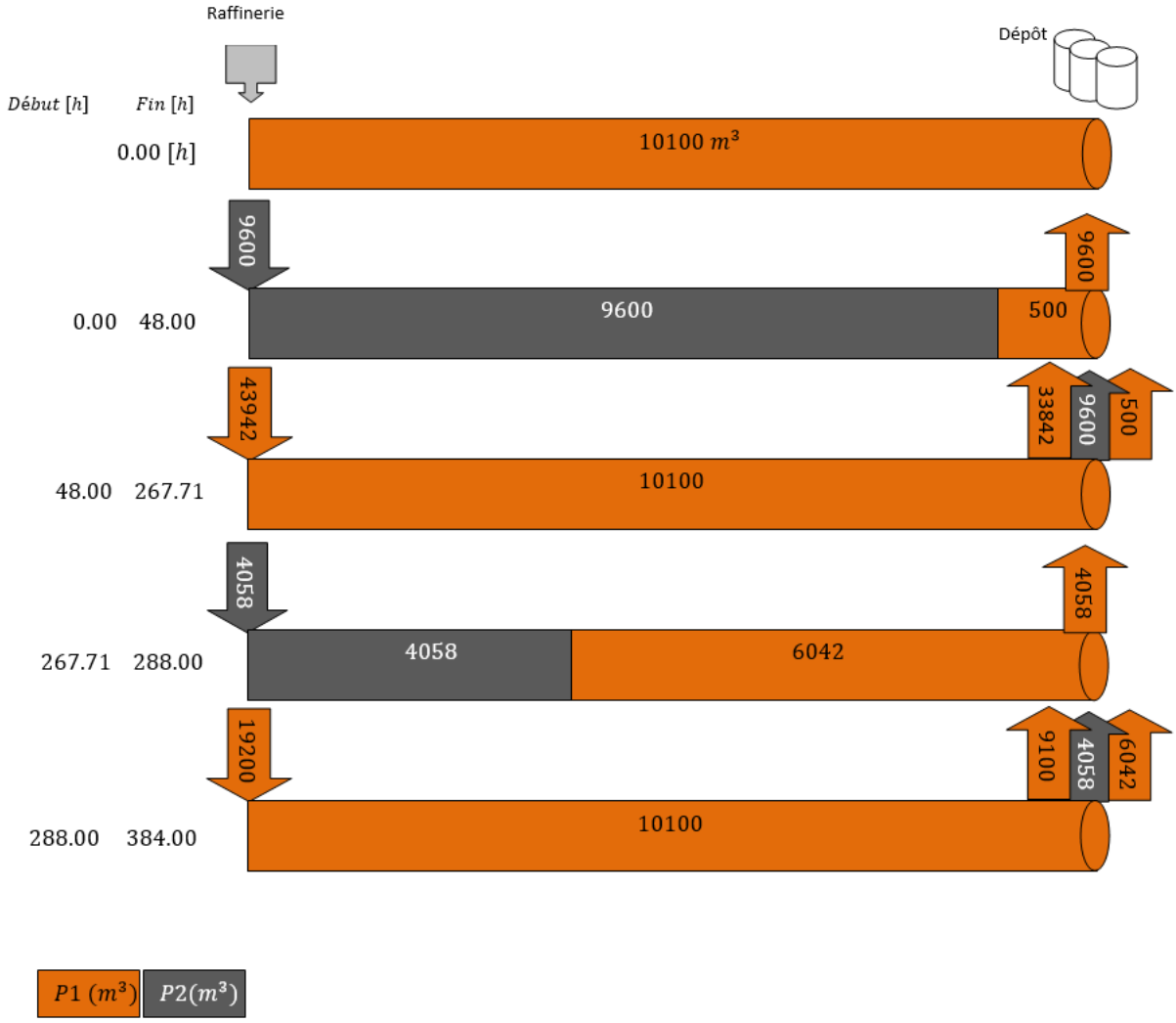


Figure 1-10 Planification optimale du transport par pipeline multi-produits (Approche MILP)

La première opération de pompage débute par l'injection du batch 1 (SCA) avec un volume de $9600\ m^3$. Cette opération commence à l'instant 0 et se termine après $48h$.
Durant cette période, le dépôt reçoit $9600\ m^3$ de produit 1 (GO) provenant de l'ancien batch 1.
A la fin de l'injection du batch 1, le contenu de la canalisation est :$9600\ m3$ de SCA du nouveau batch 1 et $500\ m3$ de GO de l'ancien batch 1.
Les opérations suivantes se déroulent de manière similaire, en suivant cette même logique.

1.3.3 Présentation du pipeline multi-produit ASR en tant que système dynamique hybride

L'écoulement de fluides dans les pipelines correspond à une dynamique continue propre aux systèmes positifs, avec des variables continues telles que le débit d'écoulement des produits et le volume transporté. Par ailleurs, l'ouverture et la fermeture des vannes d'alimentation, ainsi que la décision de transporter un produit particulier, correspondent à des événements discrets. Cela crée un système dynamique hybride où ces variables continues

interagissent avec des événements discrets. Ces événements discrets sont générés par les opérateurs qui agissent sur les électrovannes à la raffinerie pour injecter un carburant spécifique ou approvisionner les produits dans les dépôts.

La planification du transport dans un réseau de pipelines multi-produits est une préoccupation majeure au sein de l'industrie pétrolière. Ce genre de problème est complexe en raison des exigences opérationnelles liées au fonctionnement des pipelines, ainsi que d'autres contraintes telles que la satisfaction de la demande des clients. Étant donné que les pipelines représentent un système dynamique hybride (SDH), nous avons envisagé, dans le cadre de notre travail, d'utiliser des approches de modélisation et d'analyse adaptées à ce type de système. L'objectif consiste à élaborer une approche de commande pertinente qui permettra de répondre aux exigences spécifiées pour le réseau de pipelines (raffineries, canalisations et dépôts), tout en satisfaisant les demandes journalières des clients.

1.4 Conclusion

Dans ce premier chapitre, nous avons étudié les systèmes de pipelines pour comprendre leur fonctionnement, en présentant les diverses configurations de réseau existantes. Nous avons abordé les problèmes potentiels comme le contaminât, qui peut survenir lors de l'injection successive de batches dans le pipeline, ainsi que des défis comme la satisfaction des clients. La gestion de ces contraintes et la réalisation des objectifs nécessitent une planification et une optimisation du système de pipeline, ce qui implique une commande optimale dans notre contexte. Une revue de la littérature a été réalisée sur les approches existantes en matière de planification de pipelines multi-produits, ce qui a permis de positionner notre travail par rapport à d'autres études traitant de problématiques similaires. Enfin, nous avons introduit en détail notre cas d'étude, le pipeline multi-produit ASR, en mettant en évidence les spécifications requises pour ce système afin d'atteindre les objectifs fixés.

Dans le prochain chapitre, nous nous concentrerons sur les systèmes dynamiques hybrides, catégorie à laquelle appartient le système de pipeline. Nous explorerons les méthodes de modélisation et d'analyse adaptées à ce type de système, en mettant un accent particulier sur les réseaux de Petri et les automates.

Chapitre 2

Systèmes dynamiques hybrides: Modélisation, Analyse et Commande

Dans ce chapitre, nous abordons d'abord les fondements des systèmes dynamiques hybrides et examinons diverses méthodologies de modélisation, d'analyse et de commande propres à ces systèmes. Ensuite, nous mettons l'accent sur les outils spécifiques que nous utilisons dans notre étude, les réseaux de Petri et les automates. Les réseaux de Petri sont privilégiés pour leur flexibilité dans la modélisation, offrant une visualisation claire et compréhensible du système. Quant aux automates, ils sont choisis pour leur capacité à effectuer une analyse formelle et approfondie, permettant une exploration exhaustive de tous les états atteignables du système. Cette approche combinée enrichit notre compréhension du comportement du système pipeline multi-produit dans notre travail de recherche.

2.1 Systèmes dynamiques hybrides : concepts et théories

Les recherches dans le domaine des systèmes dynamiques hybrides sont à la fois passionnantes et complexes, car elles combinent des comportements continus et discrets

Avant d'approfondir ce sujet particulier, il est important de comprendre ce qu'est réellement un système. Bien qu'il existe diverses définitions d'un système dans la littérature, un système peut généralement être décrit comme un ensemble d'éléments qui interagissent pour former une entité unifiée qui remplit une fonction spécifique. Ces

systèmes peuvent être naturels, tels que les écosystèmes, ou créés par l'intervention humaine, comme les systèmes informatiques.

Dans ce contexte, nous allons détailler les fondements théoriques de ces systèmes, tout en soulignant leurs principaux principes et les défis spécifiques qu'ils engendrent.

2.1.1 Dynamique des systèmes : Une abstraction

Pour étudier et potentiellement contrôler ou optimiser un système manufacturier ou un procédé en général, plusieurs étapes sont nécessaires. Tout d'abord, il est essentiel de bien comprendre les évolutions et les comportements du système. Cette compréhension permet ensuite de modéliser et d'analyser le système, puis de le commander afin d'atteindre des objectifs spécifiques tels que la sécurité, la satisfaction des spécifications imposées par le cahier des charges, ou l'optimisation de critères donnés.

Dans la littérature, plusieurs modèles et approches d'abstraction des systèmes sont disponibles. Le choix parmi eux varie selon la nature du procédé, le but de l'étude, et la facilité d'application de l'approche. Il faut également prendre en compte la qualité d'approximation et la facilité de calcul, y compris le temps de convergence.

On peut distinguer trois différents types de systèmes dans la littérature : système à évènement discret (SED), système continu, système hybride.

Systèmes à Événements Discrets (SED)

Les systèmes à événements discrets (SED) ont des caractéristiques uniques où les états sont décrits par des variables discrètes, et les changements d'état se produisent par l'occurrence des évènements instantanés à un intervalle de temps généralement irréguliers(Sava, 2001). Dans ce type de système, les variables d'état évoluent de manière continue entre les événements et sont modifiées de manière discontinue lorsqu'un événement spécifique se produit (figure 2-1).

Un exemple fréquent de modélisation SED est une “file d'attente", qui sert souvent l'élément de base pour représenter différents types de DES (Cassandras & Lafortune, 2008). Des exemples concrets de systèmes SED incluent la circulation des véhicules dans le trafic urbain, le fonctionnement des machines dans un atelier flexible, etc.

Le comportement d'un système à événements discrets peut être soit prévisible ou déterminé, où l'on sait à l'avance comment le système va réagir ou évoluer, et peut être décrit par une simple séquence, soit non déterminé. Dans ce dernier cas, un état donné peut conduire à plusieurs possibilités d'événements, entraînant ainsi des états différents. L'évolution est ici décrite par un ensemble de séquences d'événements qui constitue un langage couvrant l'ensemble des événements possibles dans le système(Hopcroft et al., 2001).

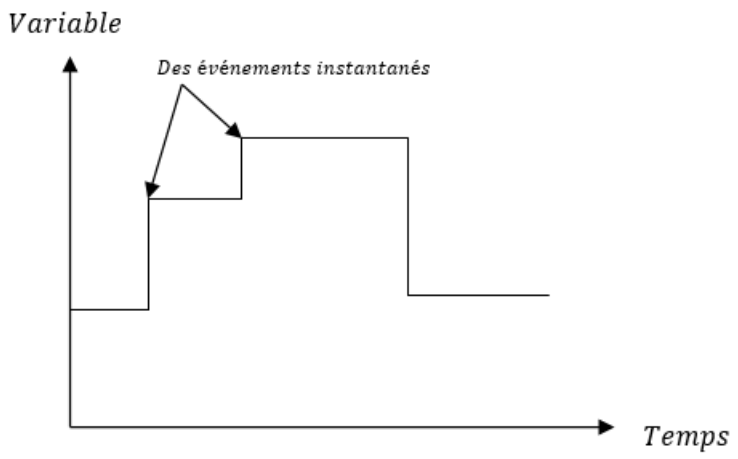


Figure 2-1 Évolution d'une Variable d'état dans un Système à Événements discrets (SED)

Les systèmes continus

Dans un système continu, le comportement du système évolue de manière continue dans le temps (figure 2-2). Ce type de système se caractérise par le fait que les variables d'état peuvent prendre des valeurs réelles au fur et à mesure que le temps (t) évolue où $t \in R$.

Des exemples de telles variables incluent le débit, la tension, la température, etc. Pour étudier ce système, nous utilisons des outils mathématiques qui permettent de décrire l'évolution continue des variables d'état. Habituellement, cela est représenté par des équations différentielles, et des méthodes d'état sous forme matricielle.

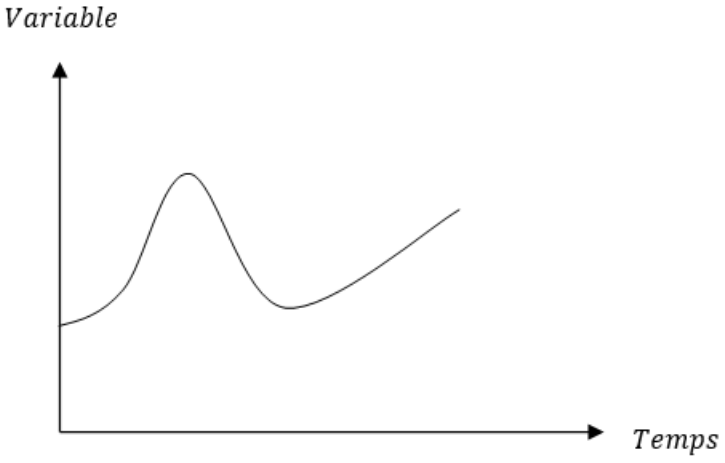


Figure 2-2 Évolution d'une Variable d'état dans un Système continu

Les systèmes dynamiques hybrides (SDHs)

Un système hybride se caractérise par l'interaction du comportement continu et discret. Il s'agit d'une combinaison de dynamiques telles que le flux continu et les événements instantanés discrets qui changent l'état du système (figure 2-3). Les variables qui décrivent le comportement du système sont également classifiées par nature continues ou discrètes.

On peut trouver les systèmes dynamiques hybrides dans divers domaines manufacturiers, tels que la robotique, l'industrie chimique, la gestion du trafic, les systèmes de communication, etc. L'exemple le plus couramment cité dans la littérature est celui d'un thermostat. Prenons l'exemple d'une pièce où nous souhaitons maintenir une température de 20 degrés Celsius pendant l'hiver. Le thermostat allume le chauffage si la température est inférieure au point de consigne et l'éteint s'il est supérieur. Ici, l'évolution de la température représente une dynamique continue tandis que la mise en marche et l'arrêt du thermostat sont des événements discrets, le passage d'un état discret à l'autre dépend de la température, donc de la partie continue. Il y a donc une interaction entre ces deux dynamiques qui forme un système hybride.

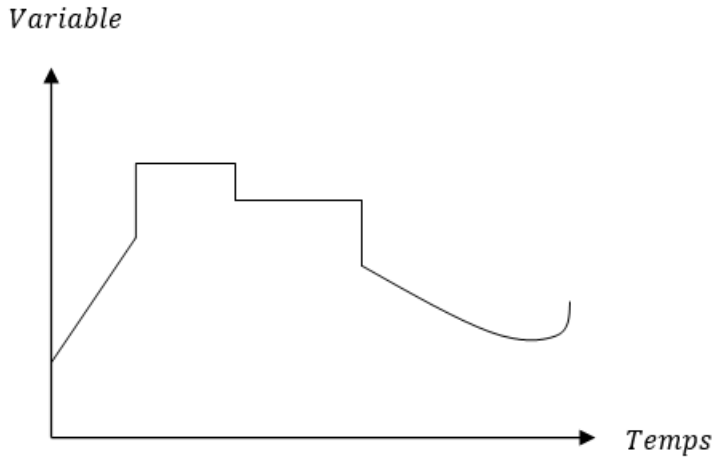


Figure 2-3 Évolution d'une variable d'état dans un SDH

2.1.2 Caractéristiques et comportement des SDHs

Tel qu'il a été abordé auparavant, un système hybride est caractérisé par l'interaction de dynamiques continues et discrètes. Dans les études scientifiques, il existe quatre formes principales de discontinuité qui s'appliquent à l'état continu.(Branicky, 1995) a proposé des classifications pour ces phénomènes hybrides. Dans cette section, nous allons brièvement présenter ces classifications.

Les commutations autonomes

La commutation autonome se produit lorsque le système hybride bascule entre des états ou des modes discrets sans avoir besoin d'un signal ou d'une action externe, la transition d'état est déterminée par la dynamique interne du système et les interactions entre les composantes discrètes et continues. Prenons l'exemple classique d'une balle rebondissante: la balle passe d'un mode de chute libre à un mode de rebond en fonction de sa position et de sa vitesse, sans aucun contrôle externe. Mathématiquement, cela se produit lorsque l'état continu atteint des seuils spécifiques, provoquant ainsi un changement discontinu des champs vectoriels.

Les commutations contrôlées

La commutation contrôlée fait référence au passage entre deux modes ou états discrets en réponse à une entrée ou un signal de commande externe. En d'autres termes, le comportement ou le champ vectoriel change de manière discontinue en réaction à une action extérieure (commande contrôlée), ce qui se traduit par une commutation entre différents champs vectoriels. Ce comportement dans un système est généralement initié par un changement des transitions de mode effectué par un opérateur ou un algorithme de contrôle. Par exemple, un contrôleur de jeux de signalisation peut changer les états selon une séquence de synchronisation prédéfinie ou un algorithme de contrôle qui prend en compte les conditions de circulation.

Les sauts autonomes

Un saut autonome se produit lorsqu'un changement instantané des variables d'état continues est provoqué par la dynamique interne du système sans aucune intervention externe. Cela se manifeste lorsque la variable d'état atteint une certaine région de l'espace d'état et effectue un changement brutal de valeur, souvent appelé saut ou impulsion de la variable. L'exemple le plus classique qui décrit cette situation est la collision entre deux corps, comme deux balles ; après la collision, la vitesse change brutalement et peut même changer de signe.

Les sauts commandés

C'est le phénomène qui se produit à la suite d'une commande contrôlée, changeant l'état continu de manière discrète. Il se produit lorsqu'un changement instantané des variables d'état continues est induit par une entrée ou un signal de commande externe. Les sauts contrôlés sont généralement utilisés lors de la conception de stratégies de contrôle pour les systèmes

dynamiques hybrides, où l'opérateur externe ou le contrôleur génère des impulsions pour modifier le comportement du système. Prenons l'exemple d'un oscillateur commandé en tension, où une tension de commande externe est utilisée pour modifier brusquement la fréquence de cet oscillateur. Dans ce cas, l'impulsion est contrôlée, car elle est provoquée par une entrée externe.

2.1.3 Approches de Modélisation des SDHs

De nombreux travaux se sont concentrés sur le développement de méthodes de modélisation des SDH, reflétant une grande diversité d'approches. Cette hétérogénéité résultait en grande partie de la nature intrinsèquement variée des SDH étudiés et, parfois, de l'application spécifique qui était envisagée. Cette diversité en matière de modélisation a été illustrée dans différents articles de la littérature (Kurovszky, 2002; Pettersson & Lennartson, 1995).

Un modèle d'un système dynamique hybride (SDH) doit intégrer à la fois des dynamiques discrètes et continues. Pour bien comprendre l'évolution de ce système, il est nécessaire de disposer d'un cadre de modélisation unifié. Cependant, d'autres approches dans la littérature tentent d'approximer soit la dynamique continue, soit la dynamique discrète, afin d'obtenir un modèle entièrement continu ou un modèle basé sur des événements discrets. Néanmoins, l'approche mixte qui intègre les deux dynamiques au sein du même modèle apparaît comme étant la plus précise pour interpréter le comportement hybride. (Lin et al., 2022) indiquent qu'il y a principalement deux groupes de modèles suivant cette dernière approche. Le premier groupe, relevant de l'informatique, perçoit les SDH comme des programmes discrets (informatiques) interagissant avec un environnement continu. Tandis que le second groupe, constitué d'automaticiens, le considère comme un ensemble d'équations différentielles ou de différence avec des variables discontinues ou à valeur multiple. Par la suite, nous décrirons brièvement chaque approche pour mettre en évidence les différences entre elles. Nous préciserons également quelle approche sera adoptée dans le cadre de notre travail.

L’approche continue

Cette approche vise à modéliser le système hybride en se concentrant uniquement sur la dynamique continue, de manière à éliminer ou à approximer l’évolution discrète à l'aide d'équations différentielles. Elle se concentre sur l'étude du comportement du modèle continu malgré la présence de discontinuités. Dans la littérature, deux méthodes principales permettent d'approximer la dynamique discrète : la première définit un ensemble des transitions pour suivre l'évolution discrète du système(Branicky et al., 1994; Witsenhausen, 1966), tandis que la seconde décrit le fonctionnement hybride du système en introduisant des variables booléennes ou entières dans le modèle continu, décrit par des équations différentielles. L'un des avantages de cette approche est qu'elle permet d'analyser la stabilité du système en utilisant les théories

appropriées aux systèmes continus. Cependant, un inconvénient majeur est qu'elle ne représente pas explicitement l'évolution discrète pour l'utilisateur. De plus, la complexité des équations produites peut poser des problèmes. Dans les travaux de (Dobbe & Tomlin, 2015; Stéphanou & Volpert, 2015), les auteurs ont mis en évidence les limites des approches continues, telles que les équations différentielles ordinaires pour modéliser un système biologique, et ont privilégié l'utilisation de la modélisation mixte des systèmes hybrides, où la partie continue et discrète s'intègrent explicitement dans le même cadre.

L'approche événementielle

Cette approche vise à approximer le système dynamique hybride (SDH) en négligeant ou en éliminant les dynamiques continues de sorte que le système soit principalement représenté par ses événements discrets. Dans les études qui adoptent cette approche, le travail (Puri et al., 1996) se démarque. Sa méthode divise essentiellement l'espace d'état continu en région, attribuant à chaque région un état discret. Cette technique permet d'appliquer la théorie traditionnelle des systèmes à événements discrets (SED) pour la commande, comme démontré dans diverses recherches (Lunze & Raisch, 2002). La modélisation basée sur les événements discrets s'avère particulièrement utile lorsque les transitions d'états sont plus significatives que les variations continues.

L'approche mixte

Cette approche offre une représentation claire et explicite des dynamiques discrètes et continues d'un système dynamique hybride (SDH), en les combinant dans un seul modèle où elles interagissent entre elles. La dynamique continue, est souvent formulée par des équations différentielles, est intégrée avec la dynamique discrète, généralement représentée par des automates à états finis ou d'autres structures similaires. Cette stratégie offre une vision précise de l'interaction entre les deux dynamiques, ce qui rend le modèle plus facile à comprendre. De plus, elle offre une grande flexibilité en termes de modélisation et peut être utilisée pour représenter différentes sortes de systèmes hybrides. L'avantage majeur de cette approche réside dans le fait qu'elle permet d'observer explicitement comment ces deux aspects du système interagissent, ce qui en fait un outil puissant et accessible pour comprendre et travailler avec les systèmes dynamiques hybrides complexes. Dans ce cadre, divers outils de modélisation ont été développe tels que les automates hybrides (Alur et al., 1993), qui sont une extension des automates à états finis, et les extensions des Réseaux de Petri, comme les réseaux de Petri hybrides (Alla & David, 1998). D'autres méthodes dans la même approche consistent à incorporer des variables binaires au sein du modèle continu, à l'image du formalisme MLD (Mixed Logical Dynamic) (Bemporad & Morari, 1999).

Notre choix s'est porté sur les réseaux de Petri hybrides pour ce travail. Cette décision s'explique par la manière explicite dont ils décrivent les aspects continus et discrets d'un

système, et par leur clarté visuelle. De plus, ces réseaux offrent une grande flexibilité et facilité lorsqu'il s'agit de modéliser un système. Le deuxième chapitre apportera plus d'informations détaillées sur cet outil spécifique de modélisation.

2.1.4 L'analyse des SDH

Après avoir modélisé un système dynamique hybride (SDH), l'étape cruciale qui suit est l'analyse. Cette étape s'initie généralement par une simulation du modèle, permettant ainsi de valider, vérifier et interpréter les propriétés à analyser du système hybride. La simulation offre une grande flexibilité et permet d'analyser une variété de modèles. Cependant, elle ne garantit pas la couverture exhaustive de toutes les évolutions possibles du comportement du modèle. En d'autres mots, malgré la précision apportée par la simulation, elle ne garantit pas l'absence d'erreurs.

D'autres approches, comme la simulation symbolique, offrent une alternative puissante pour surmonter les problèmes de la simulation classique (Alur et al., 2008; Nanez et al., 2014). Dans la simulation symbolique, plutôt que d'utiliser des valeurs numériques spécifiques, des symboles représentent un ensemble de valeurs possibles, permettant ainsi d'explorer de nombreux scénarios simultanément. Cependant au lieu d'obtenir une trajectoire unique, on obtient une représentation symbolique de diverses trajectoires possibles. Ces techniques peuvent être très gourmandes en ressources, surtout en ce qui concerne le temps de calcul et la mémoire. Ce défi est amplifié pour des systèmes comportant de nombreuses variables ou présentant une vaste diversité de comportements.

Face à cette limitation, des méthodes formelles ont été développées. Ces méthodes permettent d'analyser tous les trajectoires ou comportements du modèle d'un SDH et de vérifier les propriétés principales telles que la sûreté, la vivacité, etc.

L'approche de modélisation utilisée pour un SDH reflète généralement la vision des chercheurs lors de l'étape d'analyse. En effet, si l'approche de modélisation est basée sur une extension des formalismes des SED, comme les automates hybrides, l'analyse tend à se concentrer davantage sur la vérification des propriétés et l'analyse d'atteignabilité. D'un autre côté, les chercheurs qui privilégient l'aspect continu modélisent le système à l'aide d'équations différentielles ordinaires en intégrant des variables discrètes dans ces équations. Lors de la phase d'analyse, l'accent est principalement mis sur l'étude de la stabilité des systèmes.

Dans le cadre de notre travail, nous analyserons notre système en utilisant les automates hybrides. Par conséquent, nous portons un intérêt particulier aux travaux axés sur l'étude des propriétés et l'analyse d'atteignabilité, notamment ceux spécifiquement consacrés aux automates hybrides.

Pour plus de détails sur les propriétés structurelles des modèles, telles que la stabilité, vous pouvez consulter les travaux de (Branicky, 1998; Ye et al., 1998), ainsi que (Johansson &

Rantzer, 1997; Pettersson & Lennartson, 1996). Ces travaux utilisent principalement les fonctions de Lyapunov .

Parmi les approches qui favorisent l'approche discrète, on trouve les principales approches suivant : l'analyse d'atteignabilité symbolique basée sur le modèle checker, l'abstraction et les méthodes de vérification déductives :

L'analyse d'atteignabilité symbolique

Dans le cadre de l'analyse d'atteignabilité symbolique des systèmes hybrides, des outils informatiques ont été développés. HyTech a été le premier modèle checker à implémenter cette analyse pour les systèmes hybrides (Henzinger et al., 1997). Il est conçu pour la vérification formelle des systèmes hybrides, il est possible de vérifier des propriétés essentielles telles que la sûreté et la vivacité du modèle. Pour un automate hybride doté de n variables réels, chaque ensemble atteignable est représenté par une union finie de polyèdres n-dimensionnels associés à chaque mode. Ces polyèdres sont décrits par une conjonction d'inégalités linéaires sur les variables. La représentation basée sur les polyèdres est largement utilisée, en partie grâce à la disponibilité de bibliothèques open source facilitant leur manipulation et aussi à cause de leur application dans divers domaines informatiques.

Cependant, les modèles sont restreints aux automates hybrides linéaires où seules des expressions linéaires sont utilisées. La dynamique continue est définie par des inégalités différentielles qui posent des contraintes linéaires sur les dérivées du premier ordre. Comme de nombreux outils de modèle checking, HyTech peut être confronté au problème d'explosion d'états. Cette explosion rend la vérification ardue, en particulier pour les modèles où le nombre de variables réelles est élevé.

De nombreuses recherches ont visé à surmonter ce défi, voire à remplacer les polyèdres par d'autres représentations. Parmi celles-ci, les zonotopes et les fonctions de support se sont avérés être les plus adaptés pour analyser les systèmes hybrides linéaires. Cette évolution a conduit à l'émergence de l'outil SpaceEx (Frehse et al., 2011), capable d'analyser des systèmes à la fois complexes et de grandes dimensions.

L'abstraction

La méthode d'analyse par l'abstraction du système, l'objectif de l'abstraction dans ce contexte est d'obtenir un modèle simplifié à partir d'un modèle principal. Cette simplification vise à améliorer la capacité des outils de vérification. En effet, prouver la sûreté et les propriétés temporelles du modèle simplifié suffit pour démontrer ces mêmes propriétés pour le système principal. Les techniques d'abstraction pour un système hybride sont diverses : le modèle simplifié peut être discret tandis que le système principal est continu, ou le premier peut présenter une dynamique linéaire quand le second est non-linéaire. De nombreuses recherches

portent sur l'abstraction des systèmes hybrides, tels que l'approximation de phase-portrait (Henzinger et al., 1998) ou l'abstraction par prédicat (Alur et al., 2006; Clarke et al., 2003).

La vérification déductive

La vérification déductive des systèmes hybrides vise à prouver leur justesse, notamment dans les secteurs où l'assurance de la sécurité est primordiale, telle que la robotique et les véhicules autonomes. Cette approche utilise principalement le concept d'invariants inductifs. Ces invariants, conçus à l'origine pour les systèmes discrets (propriétés qui demeurent vraies pendant toute l'exécution d'un système), ont été adaptés aux systèmes dynamiques grâce aux certificats-barrières, qui servent à prouver la sûreté et d'autres propriétés des systèmes. Ces certificats garantissent que les états initiaux sont séparés des états non sécurisés en utilisant une fonction ψ, qui doivent respecter certaines conditions. La vérification basée sur ces invariants permet d'éviter de recalculer constamment les états atteignables. Elle assure que la propriété est satisfaite pour toutes les conditions initiales et toutes les entrées possibles, sans nécessiter de simulation ou d'analyse exhaustive de toutes les trajectoires potentielles du système. L'outil Keymaera supporte cette approche déductive. La vérification déductive peut gérer des problèmes non linéaires et est considérée comme une preuve formelle de la justesse d'un système. Toutefois, le concepteur ou l'analyste doit avoir une connaissance approfondie du comportement du système pour pouvoir proposer des invariants appropriés.

2.1.5 La commande d'un SDH

Le but de la commande dans un système est de superviser son comportement afin d'éviter des problèmes liés à la sécurité ou à l'exploitation. Dans ce contexte, les états interdits correspondent aux états indésirables, où le comportement du procédé ne respecte pas les spécifications ou le cahier des charges. Il s'agit donc du rôle de la commande qui limite ces comportements sans entraver le fonctionnement général du système.

L'ensemble des événements générés par le procédé se divise en deux catégories : les événements contrôlables et les événements incontrôlables. Les événements contrôlables sont ceux qui résultent des actions appliquées sur le procédé, telles que l'ouverture et la fermeture d'une vanne. Les événements incontrôlables, quant à eux, sont le résultat de sorties de capteurs, comme un capteur qui détecte le volume d'un liquide dans un réservoir lorsqu'il atteint un niveau maximal ou minimal.

Le superviseur ou le contrôleur peut intervenir lorsqu'il s'agit d'un événement contrôlable indésirable et l'interdire. En revanche, face à un événement incontrôlable, il ne dispose pas de la capacité d'intervenir ou de l'interdire.

Avant d'entamer la commande d'un système hybride, il est indispensable de parler de la commande des systèmes à événements discrets (SED), car le concept de supervision a d'abord été appliqué aux SED et a ensuite évolué pour inclure d'autres types de systèmes.

2.1.5.1 La supervision des systèmes à évènements discrets

Un système à événements discrets est un système où l'espace d'état est discret. Lorsqu'on est dans un état précis, le système reste stable et n'évolue pas avec le temps. Ses trajectoires d'état sont constantes par morceau, et l'état change seulement lorsqu'un événement physique est produit dans un temps généralement quelconque et irrégulier.

Le concept de la commande par supervision a été introduit dans le travail de Ramadge et Wonham (Ramadge & Wonham, 1987), où ils ont modélisé le comportement d'un SED à l'aide des automates et des langages formels. Les spécifications ont été modélisées de la même manière.

L'idée de base est de calculer ou de construire un superviseur qui interagit avec le modèle du système pour empêcher le procédé ou le système d'atteindre des états indésirables. Le superviseur réagit en interdisant certains événements et en autorisant d'autres. Le principe de fonctionnement de la commande par supervision est décrit dans la figure 2-4.

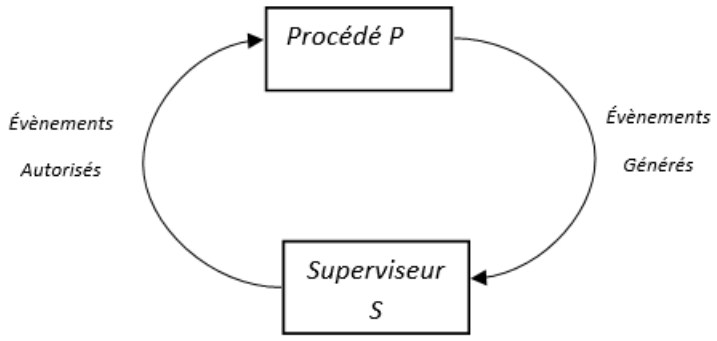


Figure 2-4 le concept de la commande par supervision

Le superviseur reçoit en entrée l'état du système courant et l'événement généré par le procédé, puis analyse ces informations et les compare aux lois de contrôle prédéfinies qui définissent les événements autorisés et interdits depuis un état précis.

Dans la pratique, la construction d'un superviseur est réalisée à l'aide d'un automate à état fini, où le comportement du procédé et les spécifications à imposer sont modélisés. L'optimal est de créer un modèle de superviseur qui commande le comportement du système et laisse le système interagir le plus possible sans limiter les états autorisés et interdits sauf les états indésirables.

Dans ce contexte, Kumar (Kumar et al., 1991) a développé un algorithme pour la commande par supervision d'un système à événement discret, déterminant les états interdits d'un automate qui modélise le comportement d'un procédé. Le chercheur a nommé l'état indésirable qu'il peut atteindre l'automate par l'occurrence d'un événement incontrôlable par un état faiblement interdit. Il a développé un superviseur qui permet d'éliminer les états interdits et faiblement interdits du fonctionnement désiré d'un automate, permettant ainsi d'obtenir un superviseur le plus permissif.

2.1.5.2 Synthèse et commande des SDHs

Les systèmes hybrides, qui comportent deux dynamiques différentes continue et discrète, présentent une synthèse de commande plus vaste et plus complexe, comparativement à un système qui n'aurait qu'une seule dynamique, comme les systèmes à événements discrets (SED) et les systèmes continus. Certains chercheurs ont mis l'accent sur la partie continue, tandis que d'autres se sont concentrés sur la partie discrète.

Dans le domaine de la synthèse et commande des Systèmes dynamiques hybrides (SDH), il existe deux approches principales qui sont largement utilisées. La première approche consiste à chercher une stratégie discrète afin de limiter le comportement du système, dans le but d'éviter des problèmes de sécurité ou de non-respect des spécifications. La deuxième approche se concentre sur la formulation et la résolution d'un problème de commande optimale (Shaikh & Caines, 2007; Zhu & Antsaklis, 2015), ainsi que sur les extensions de la théorie de Lyapunov.

Notre stratégie de commande pour notre procédé vise à définir une stratégie discrète pour calculer un contrôleur à notre système; à cet égard, nous offrons une brève revue du domaine. Parmi les méthodes les plus répandues, nous trouvons celle d'Antsaklis (Antsaklis et al., 1993). Dans ce travail, l'auteur introduit une nouvelle démarche pour la modélisation et la commande des systèmes hybrides, cette méthode se focalise sur la synthèse d'un superviseur discret destiné à un système continu. Elle se divise en trois étapes principales: la modélisation du système continu à travers un ensemble d'équations différentielles, la représentation du contrôleur discret par un automate à états finis et la mise en place d'une interface assurant la communication entre la partie continue et la partie discrète. Cette interface traduit les variables continues en symboles pour le contrôleur via un générateur et convertit les événements discrets en signaux continus par morceau à l'aide d'un actionneur.

Le contrôleur, représenté par un automate à états finis, reçoit des événements traduits par le générateur de l'interface. Cette méthode a permis l'application des techniques de commande et d'analyse développées pour les SED aux systèmes hybrides, s'appuyant sur l'approche de

Ramadge et Wonham. Se basant sur l'approche d'Antsaklis, (Stiver et al., 1996) a introduit une méthode pour la conception du contrôleur d'un système hybride , dans le même but dans (Stiver et al., 2001) ont développé une approche de contrôle, en devisant un automate, en sous-automates. Chaque sous-automate correspond à un objectif de contrôle spécifique. Par la suite, il a assemblé ces automates pour former le contrôleur complet.

Dans le travail de (Ghomri, 2012) une méthode de synthèse de contrôleur pour des systèmes hybrides à flux continu a été élaborée. Cette méthode est centrée sur l'automate atteignable résultant de l'analyse du système hybride considéré. Les spécifications sont uniquement focalisées sur les variables d'état continu du système. L'approche est détaillée pour un sommet spécifique de l'automate atteignable étudié. Dans ce contexte la partie discrète du système qui commande la partie continue, signifiant ainsi que la stratégie de commande est intrinsèquement basée sur les variables discrètes. Le travail consiste à calculer de nouveaux grades de transition contrôlables de façon à quitter le sommet avant que les spécifications ne soient violées, c'est-à-dire avant que la dynamique continue n'atteigne des régions indésirables. La méthode repose sur des formules algébriques qui déterminent les durées de séjour maximal et minimal au sein d'un sommet tout en respectant les spécifications imposées au système.

Dans les travaux de (Kurovszky, 2002), elle a proposé une méthode d'analyse d'atteignabilité pour un automate hybride ainsi qu'une synthèse de superviseur pour le système dynamique hybride, en tenant compte de l'abstraction de la dynamique continue. Son approche vise à modéliser les événements discrets à l'aide d'un automate à état fini et à représenter de manière discrétisée la dynamique continue par un système linéaire discrétisé. Le résultat de cette représentation est un automate hybride à temps discret. Suite à l'analyse d'atteignabilité, elle obtient l'automate atteignable qui modélise toutes les évolutions possibles du système étudié. À ce stade, elle réalise ce qu'elle nomme le dépliage temporel de la dynamique du système par l'abstraction de l'automate atteignable pour obtenir un automate à état fini. Dans la partie supervision, elle synthétise le superviseur grâce à des extensions de l'approche de supervision des SED, inspirées de Ramadge et Wonham, adaptées pour les systèmes temporisés. Par la suite, elle effectue le re-pliage temporel du modèle de commande synthétisé, aboutissant à un automate temporisé.

2.2 Introduction au réseau de Petri

Le réseau de Petri est un modèle mathématique introduit par le mathématicien Carl Adam Petri au début des années 1960 pour décrire le comportement des systèmes (Brauer & Reisig, 2009). Avec le temps, l'utilisation du réseau de Petri s'est étendue grâce à sa simplicité et sa flexibilité dans la modélisation. De cette base, plusieurs extensions ont émergé, notamment les réseaux de Petri continus, temporisés, hybrides, colorés et stochastiques.

Les réseaux de Petri servent à représenter, analyser et simuler des systèmes à événements discrets. Ils trouvent leur utilité dans des domaines comme les protocoles de communication, les processus de fabrication et les systèmes de gestion des flux de travail, etc.

Parmi les caractéristiques principales du réseau de Petri, telles que mentionnées dans (Seatzu et al., 2013), on observe qu'il propose un modèle à la fois mathématique et graphique. Cela présente l'avantage d'une meilleure clarté lors de la modélisation et de l'analyse par rapport aux équations mathématiques. De plus, il permet une représentation compacte d'un espace d'états étendu et facilite la modélisation de systèmes complexes en en permettant la représentation modulaire.

2.2.1 Définition du réseau de Petri

Un réseau de Petri peut être décrit comme un graphe biparti composé de deux types de nœuds, les événements, représenté par des transitions, qui correspondent à des actions ou à des changements dans le système, et les conditions, représentées par des places, qui symbolisent une condition, un état ou une situation au sein du système.

Un réseau de Petri comprend les composants suivants (figure 2-5) :

Les Places : représentés par des cercles, illustrent les états ou les conditions d'un système. Dans le cadre de la modélisation, une place d'entrée évoque souvent une précondition, des données d'entrée, une ressource nécessaire pour réaliser une tâche ou encore un buffer d'entrée. Par ailleurs, une place de sortie met en avant une postcondition, des données de sortie, la libération d'une ressource ou encore un buffer de sortie.

Les transitions: Sont représentées par des rectangles ou des barres et elles marquent les événements, les actions ou les processus de transformation qui permettent au système de passer d'un état à un autre. En ce qui concerne leur relation avec les places, chaque transition a généralement une ou plusieurs transitions d'entrée, notées ${}^{\circ}T_j$ et des transitions de sortie, notées T_j°. Cependant, il y a quelques exceptions, par exemple, la transition T_1 dans la figure 2-5 n'a pas de place d'entrée et est donc considérée comme une transition source. De même, certaines transitions n'ont pas de place de sortie et sont appelées transitions puits. Ces dernières sont souvent utilisées pour représenter des flux continus dans le modèle.

Les arcs : symbolisés par des flèches orientées, servent à connecter les places aux transitions et vice versa. Ils représentent le mouvement des jetons entre ces éléments.

Les Jetons : sont représentées par des points ou des petites marques. Elles indiquent la présence ou la disponibilité de ressources, ainsi que l'occurrence ou la validation d'une condition spécifique, etc. La dynamique du réseau de Petri est basée sur le franchissement de transitions. Lorsqu'une transition est franchissable, des jetons sont consommés à partir des places d'entrée et produits dans les places de sortie, ce qui modifie l'état du système. Le marquage, où l'état actuel du système donne une vision de la répartition des jetons dans les différentes places du réseau de Petri après l'occurrence d'un évènement.

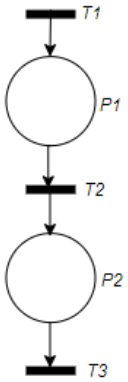


Figure 2-5 Réseau de Petri : Places et Transitions

Définition 2.1 (Un réseau de Petri)

Un réseau de Petri peut être formellement défini comme un quadruplet $(P, T, Pre, Post, M_0)$ où

$P = \{P_1, P_2 \dots, P_m\}$: est un ensemble de m places.

$T = \{T_1, T_2 \dots, T_n\}$: est un ensemble de n transitions.

$Pré : P \times T \rightarrow N$: est l'application d'incidence avant. Elle spécifie le nombre d'arcs sortant des places vers les transitions, et elle est représentée par une matrice de dimension m, n , chaque élément de cette matrice est défini par :

$(P_i, T_j) \rightarrow Pré(P_i, T_j)$: Qui donne le poids de l'arc reliant la place P_i à la transition T_j .

$Post : P \times T \rightarrow N$ est l'application d'incidence arrière, qui spécifie le nombre d'arcs sortant des transitions vers les places. Elle aussi est représentée par une matrice de dimension m, n , chaque élément de cette matrice est défini comme suit :

$(P_i, T_j) \rightarrow Post(P_i, T_j)$: Qui donne le poids de l'arc reliant la transition T_j à la place P_i .

Afin de comprendre l'évolution des états au sein d'un RDP autonome, nous aborderons l'exemple du modèle producteur-consommateur, illustré à la Figure 2-6

Le système fonctionne selon un processus cyclique entre le producteur et le consommateur. Le producteur crée des éléments et les place dans un buffer ayant une capacité maximale de n éléments. De son côté, le consommateur récupère ces éléments un par un.

Au départ, si le buffer contient k éléments, représentés par le marquage de la place P_2, alors le marquage de la place P_1, qui indique le nombre de places disponibles dans ce buffer, affichera une marque de $n - k$ éléments.

Le producteur : Chaque fois que la transition T_1 est franchie, cela entraîne l'ajout d'un jeton dans le buffer. Cependant, cette production ne se produit que s'il y a de la place disponible dans le buffer, c'est-à-dire si la place P_1 est marquée.

Le consommateur : retire un élément chaque fois qu'il en trouve un disponible dans le buffer, c'est-à-dire lorsque la place P_2 est marquée.

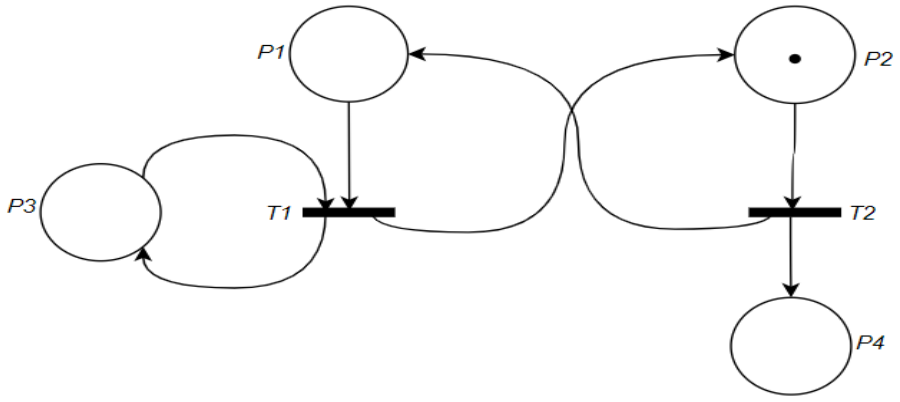


Figure 2-6 Réseau de Petri : Exemple de producteur et consommateur

Définition 2.2 (la matrice d'incidence)

La matrice d'incidence est déterminée par la différence entre la matrice d'incidence arrière $(Pr\acute{e})$ et la matrice d'incidence avant $(Post)$, exprimée mathématiquement par : $C = Post - Pr\acute{e}$.

Des notations spécifiques sont employées pour définir la position d'une transition et d'une place dans un modèle réseau de Petri.

Pour une transition

Les places entrantes sont notées : ${}^{\circ}T_j \{p \epsilon P / Pr\acute{e}(p,t) > 0\}$

Les places sortantes sont notées : $T_j^{\circ} \{p \epsilon P / Post(p,t) > 0\}$

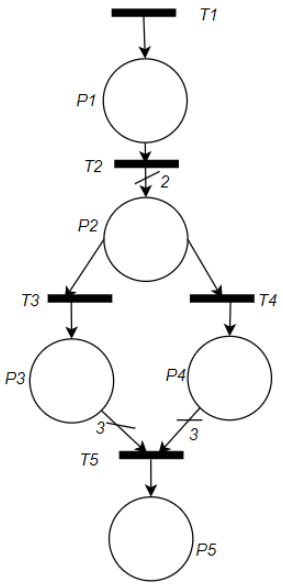
Pour une place

Les transitions entrantes sont notées : ${}^{\circ}P_i \{t \epsilon T / Post(p,t) > 0\}$

Les transitions sortantes sont notées : $P_i^{\circ} \{t \epsilon T / Pr\acute{e}(p,t) > 0\}$

Exemple :

Un exemple de modèle de réseau de Petri, ainsi que sa matrice d'incidence correspondante, sont présentés à la Figure 2-7



$$Pré = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad Post = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2-7 Réseau de Petri avec la matrice d'incidence associée.

2.2.1.1 L'évolution du marquage dans un modèle réseau de Petri

L'évolution d'un réseau de Petri d'un état à un autre est fonction de la répartition des jetons entre les places. Cette dispersion survient après le franchissement de transitions. On nomme cette configuration de jetons le "marquage". Par conséquent, chaque mouvement de jetons conduit à un changement d'état du système, représenté par un nouveau marquage.

Définition 2.3 (le marquage)

Le marquage, noté M, est une fonction qui associe à chaque place d'un réseau de Petri un nombre entier non négatif de jetons : $m: P \rightarrow N$. Dans les représentations graphiques de réseaux de Petri, les jetons sont symbolisés par des petits cercles noirs à l'intérieur des places.

Définition 2.4 (une transition validée)

Une transition T_j est validée dans un marquage M d'un réseau de Petri ,si et seulement si pour chaque place P_i entrant dans T_j :

$$\forall\, P_i \in {}^{\circ}T_j\,, \frac{m_i}{Pré\left(P_i\,, T_j\right)} > 0$$

Donc pour que la transition T_j soit validée, il faut que chaque place P_i contienne un nombre de jetons supérieur ou égale à $Pré\ (P_i, T_j)$ pour le marquage M . (figure 2-8)

Exemple (figure 2-8)

L'exemple présente différentes configurations de la transition T_1 en fonction du marquage des places entrant dans cette transition. Dans la figure (a), T_1 n'est pas validée car le nombre de jetons dans P_1 est inférieur à au $Pré\ (P_1, T_1)$, qui est égal à 2. En revanche, dans la figure (b), cette condition est satisfaite et donc T_1 est validée. De la même manière, dans la figure (c), T_1 n'est pas validée car P_1 ne contient pas suffisamment de jetons et le nombre de jetons dans P_3 est inférieur à $Pré\ (P_3, T_1)$. Cependant, dans la figure (d), la transition T_1 est validée.

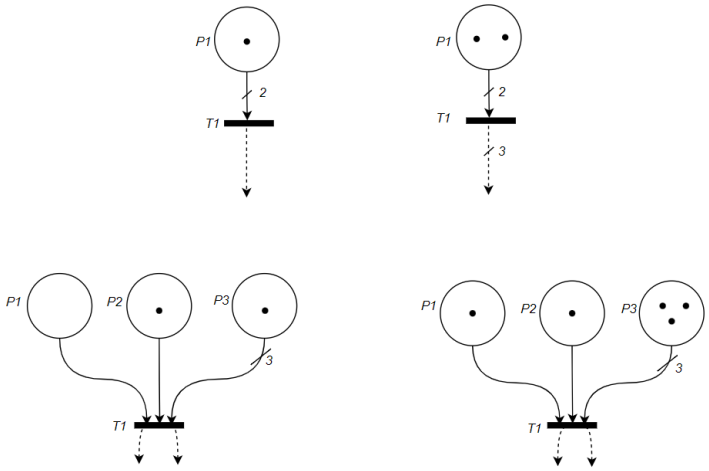


Figure 2-8 Réseau de Petri : validation d'une transition

Définition 2.5 (Le franchissement d'une transition)

Pour qu'une transition T_j soit franchissable par un marquage M, elle doit tout d'abord être validée. Franchir cette transition consiste à retirer des jetons des places entrantes de T_j . Concrètement, cela signifie supprimer un nombre de jetons de la place P_i égal à $Pré\ (P_i, T_j)$, ensuite ajouter un nombre de jetons aux places sortantes de T_j, équivalent à $Post\ (P_i, T_j)$pour chaque placeP_i .

Le franchissement d'une transition conduit à un nouveau marquage, noté M'.

Définition 2.6 (Une séquence de franchissement)

Une séquence de franchissement est une suite successive de transitions franchies, conduisant à une succession de marquages. Nous notons cette suite par S.

Définition 2.7 (un marquage atteignable)

Le marquage M est dit atteignable s'il existe une séquence de franchissement S qui conduit à ce marquage depuis le Marquage initial M_0 . Autrement dit, s'il excite une séquence S telle que $M_0[S \rightarrow M$.

L'ensemble atteignable d'un réseau de Petri marqué, noté $< N, M0 >$, représente tous les marquages qui peuvent être atteints à partir du marquage initiale M_0. Cet ensemble est généralement représenté sous la forme d'un graphe nommé un graphe d'atteignabilité ou le graphe de marquage accessible.

2.2.1.2 Configurations classiques dans un modèle de réseau de Petri (RdP)

Dans cette section, nous allons présenter les configurations classiques observées dans un RdP en proposant des exemples simples pour chaque configuration.

Exclusion mutuelle

Cette configuration modélise un système où deux processus tentent d'accéder à une ressource partagée de manière exclusive. L'exclusion dans notre exemple (figure 2-9) est assurée par le fait que la ressource, représentée par la place P_1, ne peut être détenue que par un seul processus à la fois. Par exemple, lorsque le processus A obtient la ressource, P_2, est marqué et P_1, ne l'est pas. Inversement, si le processus B obtient la ressource, P_3, est marqué et P_3, ne l'est pas.

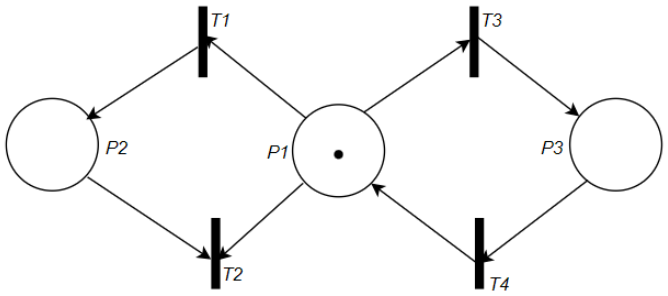


Figure 2-9 Exclusion mutuelle dans un réseau de Petri

Un conflit

Considérons l'exemple illustré par la figure 2-10. Nous observons que les transitions T_1 et T_2 sont validées. Toutefois, elles ne peuvent pas être franchies simultanément car il n'y a qu'un seul jeton dans la place P_1. Par conséquent, elles entrent en compétition pour consommer ce jeton. Si T_1 est franchie, l'événement associé à la transition T_2 devient irréalisable. Cette situation est désignée comme un "conflit". La présence de tels conflits dans un réseau de Petri le caractérise comme non déterministe.

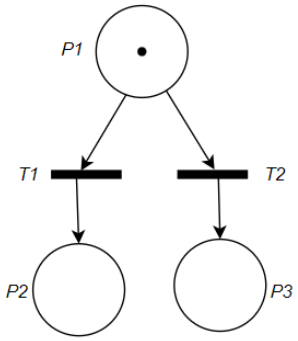


Figure 2-10 Un conflit dans un réseau de Petri

Deux types de conflits peuvent être présents dans un modèle de réseau de Petri , les conflits structurels et les conflits effectifs. Deux transitions sont en conflit structurel lorsqu'elles partagent une place entrante commune. Un conflit effectif entre deux transitions est observé lorsqu'il existe un marquage où les deux transitions sont validées et que l'activation de l'une empêche l'activation de l'autre. Par exemple, le RdP présenté dans la figure 2-11(a) montre un conflit effectif, tandis que le RdP la figure 2-11(b) illustre seulement un conflit structurel.

Définition2.8 un conflit effectif pour un marquage M sur une place Pi est un conflit structurel tel que le nombre de jetons dans Pi est inferieur strictement au nombre de transition franchissables en aval de Pi.

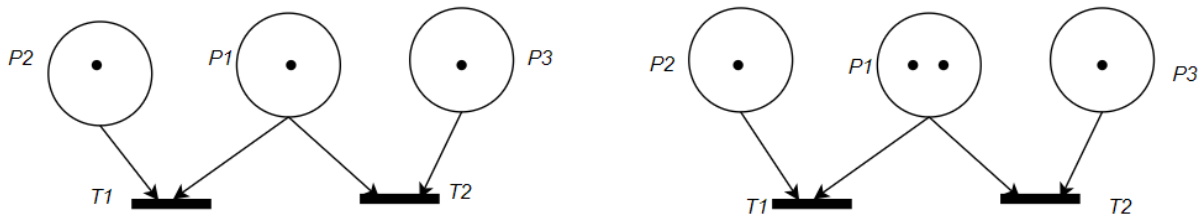


Figure 2-11 Conflit structurel et Conflit effectif

Remarque 2.1 :

Plusieurs politiques de résolution de conflits peuvent être appliquées pour régler ces situations. Parmi les options, on peut établir des priorités prédéfinies pour les transitions, opter pour une sélection aléatoire, ou encore utiliser une sélection basée sur des critères temporels. Le choix de la politique dépendra en grande partie des besoins spécifiques du système étudié et des objectifs visés par la modélisation.

La Synchronisation

Dans le contexte des réseaux de Petri, la synchronisation désigne une configuration où plusieurs taches doivent être achevées avant qu'une nouvelle tâche puisse être activée. Elle

garantit l'ordre ou le timing précis selon lequel certaines actions ou processus se déroulent, assurant ainsi une coordination entre eux.

Dans un réseau de Petri, cette synchronisation est généralement symbolisée par une place qui reçoit des jetons issus de diverses transitions. Cette place doit accumuler un nombre déterminé de jetons avant que la transition suivante puisse se franchir. Cela assure que les processus relatifs aux transitions antérieures sont bien finalisés avant de démarrer le processus ultérieur.

Par exemple dans la figure 2-12, P_1 et P_2 représentent deux processus ou tâches distinctes qui doivent être achevées avant que l'opération, symbolisée par le franchissement de T_3, puisse démarrer, comme dans le cas d'un processus d'assemblage.

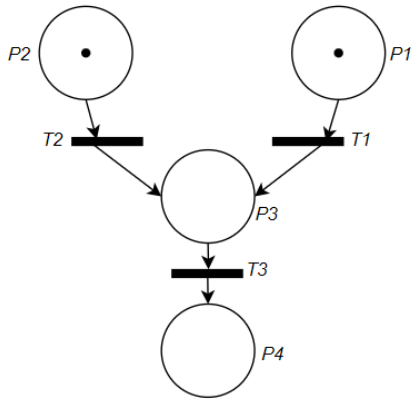


Figure 2-12 La synchronisation dans un réseau de Petri

Le parallélisme

Le parallélisme se produit lorsque plusieurs processus ou activités sont exécutés simultanément et indépendamment les uns des autres.

On considère que les deux transitions, telles que T_2 et T_3 illustrées dans la figure 2-13, sont structurellement parallèles si elles n'ont aucune place d'entrée en commun.

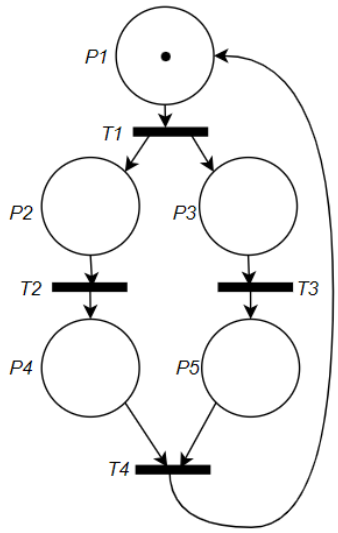


Figure 2-13 Le parallélisme dans un réseau de Petri

2.2.2 L'Analyse d'un réseau de Petri

Dans cette section, nous allons introduire les techniques d'analyse basées sur l'énumération des états atteignables. Si l'ensemble des états atteignables est fini, le graphe d'atteignabilité est jugé adéquat pour analyser le comportement d'un réseau de Petri. En revanche, si cet ensemble est infini, nous établissons un arbre de couverture. Nous allons détailler ci-après la construction du graphe d'atteignabilité et de l'arbre de couverture.

2.2.2.1 Les outils d'analyse d'un réseau de Petri

Le graphe d'atteignabilité:

Le graphe d'atteignabilité représente les marquages accessibles d'un réseau de Petri, où le franchissement d'une transition permettant de passer d'un marquage a un autre. Dans ce graphe :

- Chaque sommet est correspond à un marquage.
- Chaque arc représente une transition modélisant un évènement.

Les principales étapes de construction du graphe d'atteignabilité sont les suivants :

Etape 1 : Construire le sommet initial du graphe qui correspond au marquage initial

Etape 2 :

- Pour chaque transition T_j validée par le marquage M, calculer le marquage M' résultant du franchissement de transition T_j , en se basant sur l'équation suivante :
- $\forall P_i \in P : M'(P_i) = M(P_i) - Pré\left(P_i, T_j\right) + Post\left(P_i, T_j\right)$
  Si le sommet M' est déjà dans le graphe,

- Ajouter un arc étiqueté par Tj partant de M vers M'

Si non

-Ajouter le sommet M' au graphe,

-Ajouter un arc étiqueté par Tj partant de M vers M'

Pour chaque sommet non encore analysé, répéter l'étape 2.

L'arbre de couverture

L'arbre de couverture est utilisé lorsque la construction du graphe d'atteignabilité n'est pas faisable, notamment lorsque l'ensemble atteignable est non fini. En effet, même si l'ensemble atteignable est infini, Karp (Karp & Miller, 1969) a démontré que l'arbre de couverture est toujours fini. Pour élaborer cet arbre, la notion de ω-marquage est introduite. La procédure de construction de l'arbre de couverture est détaillée ci-dessous :

Étape 1 : construire la racine de l'arbre de couverture qui représente le marquage initial $M0$

Étape 2 :

Pour chaque sommet M non étiqueté dans l'arbre de couverture faire:

Pour chaque transition T_j validée par le marquage M fait le suivant :

a- Calculer le marquage atteignable M' résultant du franchissement de T_j depuis le marquage M.

b- Pour tous les marquages $M*$ strictement inférieurs a M' sur le chemin allant du sommet $M0$ vers le sommet , et pour tous les places $p \epsilon P$

Si $*[p] < m'[p]$, alors $m'[p] = w$.

c- Ajouter un nouveau sommet M' à l'arbre de couverture.

d- Ajouter un nouvel arc étiqueté par Tj du sommet M vers le nouveau sommet M'

e- S'il le sommet M' existe déjà dans l'arbre, étiqueté M' par <dupliqué>

et le sommet M par <ancien>.

S'il existe des sommets non encore étiquetés, aller à l'étape 2

2.2.2.2 Les propriétés d'un réseau de Petri

L'étude des propriétés principales des réseaux de Petri s'avère essentielle pour l'analyse de ces modèles. De nombreux articles en littérature traitent en détail des différentes propriétés d'un réseau de Petri. Dans la suite, nous mettrons l'accent sur les concepts d'atteignabilité et de décidabilité associées à ces réseaux.

Atteingnabilité et décidabilité dans un RdP

La décidabilité peut être définie comme suit : un problème est considéré comme décidable s'il existe un algorithme capable de traiter toutes les formulations possibles de ce problème en un nombre d'itérations fini.

En revanche, il est "semi-décidable" lorsque l'algorithme peut traiter toutes les formulations possibles, mais le nombre d'itérations pour trouver une solution est fini dans certains cas seulement. Par exemple, si l'on est confronté à deux alternatives : en choisissant la première, on parvient à une solution, tandis que la deuxième n'offre pas de solution.

Problème d'atteignabilité dans un réseau de Petri : L'atteignabilité est une préoccupation majeure dans les réseaux de Petri. Lorsque l'ensemble des états est fini, on utilise le graphe d'accessibilité. Cependant, si cet ensemble est infini, on fait appel à l'arborescence de couverture. Cependant, cette dernière ne fournit pas toujours toutes les informations nécessaires pour déterminer si un marquage spécifique peut être atteint. En effet, lorsqu'on cherche à savoir si un marquage M peut être atteint dans un réseau de Petri, nous générons toutes les séquences possibles en commençant par les plus courtes afin de déterminer le marquage correspondant, si le marquage M peut être atteint avec une séquence de longueur k, l'algorithme se termine à l'itération numéro k. Si le marquage n'est pas accessible, alors l'algorithme diverge. Ce problème d'accessibilité ou d'atteignabilité est donc considéré comme semi-décidable.

2.2.3 Réseau de Petri T–temporel

Le réseau de Petri T-temporel est une extension du réseau de Petri autonome en y ajoutant la dimension temporelle. Dans ce type de réseau, une ou plusieurs transitions, notées T_j peuvent être associées à un intervalle de temps noté [a_j , b_j].

Dans un réseau de Petri autonome, le marquage du réseau détermine si une transition est validée ou non. Une fois validée, elle est franchissable automatiquement, sauf en cas de conflit. Toutefois, avec la dimension temporelle introduite dans un RdP T-temporel, même si une transition est validée par le marquage autonome sous-jacent, elle ne peut être franchissable que si le temps écoulé depuis sa validation est supérieur ou égal à une durée minimale de temps (t) notée a_j , où ai et b_j sont respectivement le temps au plus tôt et au plus tard pour le franchissement de la transition T_j .

Cet intervalle temporel permet de préciser la durée pendant laquelle une transition peut ou ne peut pas être franchissable.

Les réseaux de Petri t-temporels sont d'une grande utilité dans la modélisation et l'analyse des systèmes où le temps et les délais jouent un rôle crucial, tels que les systèmes de communication (Berthomieu & Diaz, 1991; Murata, 1989), les systèmes temps réel et les systèmes de contrôle (Silva & Del Foyo, 2012). Ils permettent de vérifier des propriétés de sûreté et de vivacité, ainsi que d'étudier des problématique liées à la performance et à la planification.

Exemple : Atelier d'assemblage

Nous examinons un atelier d'assemblage simple de deux pièces différentes, illustré dans la figure 2-14. L'atelier se compose de deux machines ayant une capacité limitée et chaque machine fabrique une pièce à la fois. La première machine est dédiée à la fabrication d'une pièce de type A tandis que la deuxième s'occupe de la pièce de type B. Un poste d'assemblage est également présent dans cet atelier, il assemble les deux pièces pour former un produit fini. Dans cet exemple, nous faisons l'hypothèse qu'il n'y a pas de contrainte de disponibilité de matière première.

En ce qui concerne le temps de fabrication, la machine 1 dédiée à la pièce A nécessite entre 2,5 et 3 unités de temps (u.t) pour la fabrication, tandis que la machine 2 produisant la pièce B a un temps de traitement qui varie entre 1,8 et 2 u.t.

Si nous traduisons ce processus en termes de réseau de Petri la disponibilité de la machine 1 est symbolisée par la place P_1 et celle de la machine 2 par la place P_2 . La transition T_1 qui est associée à un intervalle [2.5, 3] représente le traitement de la pièce A. De même, la transition T_2, liée à l'intervalle [1.8, 2], symbolise le traitement de la pièce B.

Après la phase de fabrication, les pièces A et B sont stockées dans des buffers spécifiques pour chaque machine, et ces buffers sont représentés par les places P_3 et P_4 respectivement. Quand les deux pièces sont prêtes, elles sont assemblées, cette condition est représentée par la place P_5 .

Dans le marquage initial, les places P_1 et P_2 sont marquée, validant ainsi les transitions T_1 et T_2. Cependant, ces transitions ne peuvent être franchies qu'après un certain délai, la transition T_1 sera franchissable après 2,5 u.t de sa validation et devra l'être avant 3 u.t. Quant à la transition T_2, elle pourra être franchie après 1,8 u.t et devra l'être avant 2 u.t.

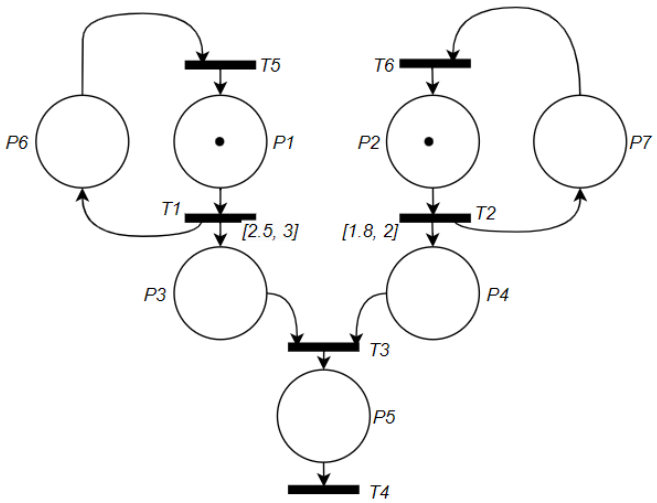


Figure 2-14 Modèle RdP T-Temporel : Atelier d'assemblage

Définition 2.9 (réseau de Petri T-temporel)

Le réseau de Petri T-temporel est un 6-uplet $(P, T, Pré, Post, M0, Is)$:

P : $\{P_1, P_2, \ldots . P_n\}$ c'est un ensemble fini de places.

T : $\{T_1, T_2, \ldots . T_m\}$ Un ensemble fini de transitions, représentant les événements du système.

Pré : Une matrice de pré-incidence, $Pré$: $P \times T \rightarrow N$, qui indique le nombre de jetons requis pour le franchissement de chaque transition depuis chaque place d'entrée.

Post : C'est une matrice de post-incidence, $Post$: $P \times T \rightarrow N$, qui indique le nombre de jetons produits pour chaque transition vers chaque place de sortie.

M_0: Un vecteur de marquage initial, M_0: $P \rightarrow N$, qui attribue un nombre entier non négatif de jetons à chaque place, représentant ainsi l'état initial du système.

Is : Une fonction d'intervalle de temporisation, $Is : T \rightarrow (Q \times (Q \cup \{\infty\})$, qui associe à chaque transition T_i un intervalle de temps $[a_{is}, b_{is}], Is(Ti) = [a_{is}, b_{is}]$ tel que $a_{is} <= b_{is}$, sachant que a_{is}, bis appartient a $Q +$.

a_{is} :représente l'instant de franchissement au plus tôt de la transition T_i .

b_{is} : est l'instant de franchissement au plus tard e la transition T_i.

L'intervalle $[a_{is}, b_{is}]$ illustre la fenêtre de temps pendant laquelle la transition peut être franchie une fois qu'elle est validée, et cela modélise la contrainte temporelle imposée pour le franchissement de la transition T_i.

- L'état d'un réseau de Petri t-temporel est défini par un couple $S\ (M, I)$:

M : Il représente le marquage du réseau de Petri, c'est-à-dire la répartition des jetons dans chaque place dans ce marquage.

I : C'est un vecteur d'intervalles de franchissement. Il mémorise les dates possibles pour le franchissement d'une transition validée.

Note

Pour qu'une transition temporelle puisse être franchie, elle doit d'abord être validée par le marquage autonome sous-jacent. De plus, après sa validation, un temps minimal (représenté par ais) doit s'écouler, et elle doit être franchie avant un délai maximal (représenté par bis).

2.2.4 Le réseau de Petri continu RdPC

Définition 2.10 (réseau de Petri continu RdPC) (Alla & David, 1998)**.**

Un réseau de Petri continu est quinq uplet $(PC, TC, Pré, Post, M_0)$

P: Un ensemble fini de places continues `C-places`.

T: Un ensemble fini de transitions continues `C-transitions`.

$Pr\acute{e}$: Une matrice de pré-incidence, $Pr\acute{e}$: $P \times T \rightarrow Q+$. Qui indique le nombre réel de marques nécessaires pour chaque transition depuis chaque place d'entrée.

$Post$: Une matrice de post-incidence, $Post$: $P \times T \rightarrow Q+$, qui indique le nombre réel de marques produits pour chaque transition vers chaque place de sortie.

M_0: Un vecteur de marquage initial, M_0: $P \rightarrow R \geq 0$, qui attribue une valeur réelle non négative à chaque place.

Remarque 2.2 . Afin de différencier les nœuds discrets des nœuds continus dans un modèle de réseau de Petri, les transitions continues sont représentées par un double cercle, et les transitions continues sont symbolisées par une double barre. (Figure 2-15)

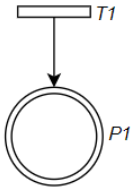


Figure 2-15 Transition et Place continues

Définition 2.11. Le degré de validation d'une transition Tj pour un marquage M noté par q ou $q(T_j, m)$, tel que $q = min\,(i : P_i \in {}^{\circ}T_j)\ \frac{m(P_i)}{Pr\acute{e}\,(P_i, T_j)}$)

Si $q > 0$, la transition Tj est validé, noté $q - valid\acute{e}$. (David & Alla, 2010)

Définition 2.12. Une transition continue T_j peut être franchie β fois simultanément, où $\beta \leq q$. B est nommé la quantité de franchissement de la transition continue T_j. La notation $[T_j]^{\beta}$ dénote le franchissement simultané de la transition T_j β fois.

Il n'est pas possible de construire un graphe de marquage pour un RdP continu car il est non dénombrable. Par conséquent, on utilise la notion de macro-marquage pour construire un graphe d'atteignabilité.

Dans la littérature, on distingue quatre différents types de réseau de Petri continu. La différence réside dans la façon de calculer ou de définir la vitesse de franchissement d'une transition continue. Dans notre travail, nous nous intéressons aux réseaux de Petri continus à vitesse constante.

2.2.4.1 Le réseau de Petri continu a Vitesse constante

Définition 2.13 . Un RdPCC est un couple $RdCC = (RC, V)$ tel que :

RC : est RdP continu autonome marqué.

$V : T \rightarrow R+$, où pour chaque transition continue T_j on associe sa vitesse de franchissement maximale Vj, $T_j \rightarrow Vj$

Exemple d'un RdPCC

Pour expliquer le principe de fonctionnement ou l'évolution d'un RDPCC, prenons un exemple classique d'un simple flow shop composé de trois machines. Devant les machines 1 et 2, il y a un buffer associé pour stocker les encours qui attendent le traitement par la machine. (figure 2.16)

Considérons $v1, v2$ et $v3$, des vitesses représentent le flux des pièces par unité de temps.

À l'instant initial, toutes les transitions T_1, T_2 et T_3 sont franchissables à leurs vitesses maximale, car $T1$ est une transition source, elle est toujours franchissable à sa vitesse maximale. Pour T_2 et T_3, les places en amont P_1 et P_2 sont marquées. Lorsqu'une transition validée est franchissable avec sa vitesse maximale, on dit qu'elle est fortement validée.

À l'état initial, on suit l'évolution des marquages des C-places en se basant sur les équations suivantes :

$$m_1(t+dt) = m_1(t) + (V1 - V2)dt$$

$$m_2(t+dt) = m_2(t) + (V2 - V3)dt$$

On a le vecteur de marquage à l'état initial M_0=$(20\ 45)^T$, donc :

$$m_1(t+dt) = 20 + (3.5 - 4)dt$$

$$m_1(t+dt) = 45 + (4 - 3.8)dt$$

Ces équations sont vraies à condition que le marquage des places P_1et P_2 Notés m_1 et m_2 soit strictement supérieur à 0. Mais à $t = 40$, $m_1 = 0$. Donc, la transition T_2 n'est plus franchissable à sa vitesse maximale, mais elle continue d'être alimentée par T_1, donc sa vitesse de franchissement maximale devient égale à $v1$. Dans ce cas, T_2 devient faiblement validée. Et à $t = 265$, m_2 s'annule et elle va être alimentée à son tour par T_1, et la transition T_3 devient aussi faiblement validée.

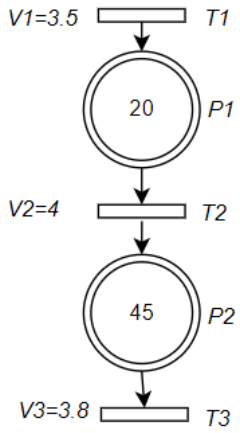


Figure 2-16 RdPCC d'un système manufacturier composé de deux machines

Le vecteur des vitesses instantanées du RdPCC est donné par :

$$\begin{pmatrix} \mathrm{V1(t)} \\ \mathrm{V2(t)} \\ \mathrm{V3(t)} \end{pmatrix} = \begin{pmatrix} [3.5 \quad 4 \quad 3.8]^T & Pour\ 0 <= t <= 40 \\ [3.5 \quad 0 \quad 3.8]^T & Pour\ 40 <= t <= 265 \\ [3.5 \quad 0 \quad 0]^T & Pour\ t <= 265 \end{pmatrix}$$

L'état d'un RDPCC est défini par le vecteur de franchissement instantané des transitions, noté V, ou par le vecteur des dérivées des marquages des c-places noté $\dot{m}$. Ce qui modifie l'état d'un RDPCC est l'annulation du marquage de l'une de ces c-places.

Le comportement d'un modèle RDPCC est illustré à travers un graphe : chaque sommet symbolise un IB-état, tandis qu'un arc est étiqueté par le marquage de la place qui s'annule et l'instant t de cette annulation.

Définition 2.14 (un IB-état)

Un IB-état d'un RDPCC correspond à une phase d'évolution durant laquelle le vecteur des vitesses instantanées demeure constant. En d'autres termes, lorsque nous sommes dans le même IB-état, l'évolution du marquage est linéaire.

La figure 2-17 illustre le graphe d'évolution correspondant à celui du RDPCC présenté dans la figure2-16.

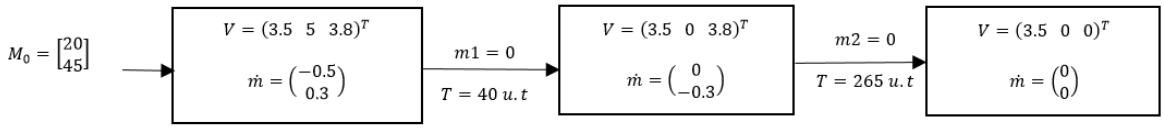


Figure 2-17 Graphe d'évolution du modèle RDPCC présenté dans figure2.12

2.2.5 Les réseaux de Petri hybrides RdPH

Un réseau de Petri hybride est une extension des réseaux de Petri (RdP). Il permet de modéliser explicitement les deux dynamiques d'un système hybride sans avoir à faire abstraction de l'une des dynamiques. Ceci offre une vision claire et visuelle du système modélisé. Un tel réseau intègre des nœuds discrets, comme les places et transitions discrètes, ainsi que des nœuds continus, comme les places et transitions continues. Cette configuration établit un lien entre un fonctionnement continu et un comportement logique (Alla & Ghomri, 2012; David & Alla, 2001).

Nous avons déjà décrit le fonctionnement d'un RDP T-temporel ainsi que celui du réseau de Petri continu à vitesse constante (RDPCC). Nous aborderons ensuite l'interaction et l'influence de la composante continue sur la partie discrète, et vice versa. Nous illustrerons ces interactions à travers des exemples simples.

Dans la figure 2-18 'a', la partie continue est influencée par la partie discrète. Même si la place continue P_1 est marquée, la transition T_1 ne peut être validée que si la transition discrète P_3 est également marquée.

La figure 2-18 'b' et figure 2-18 'c' illustrent l'influence de la partie continue sur la partie discrète. Dans la figure 'b', la transition discrète T_1 est validée si la place discrète P_1 est marquée et que le marquage de la place continue P_3 est inférieur à 7.8 marques. . Tandis que dans la figure 'c', la transition T_1 est validée si la place discrète P_1 est marquée et que le marquage de la place continue P_3 est supérieur à 2.2 marques.

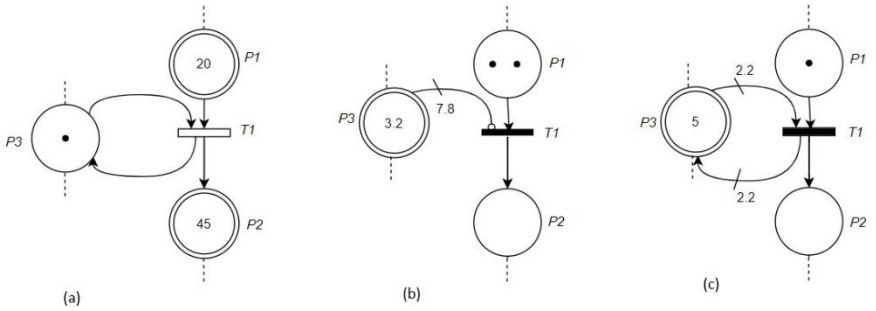


Figure 2-18 L'interaction entre les nœuds continus et discrets dans un RdPH

Remarque2.3 Dans la figure 2-18, nous n'observons pas de transformation de marquage entre le discret et le continu, ni vice versa. Ceci est garanti par les boucles qui relient les places discrètes aux transitions continues et les places continues aux transitions discrètes. Par exemple, dans la figure 'a', nous voyons que le jeton de la place discrète P_3 n'est pas consommé ou fluidité par la transition continue T_1. De même, dans la figure 'c', la quantité entrante de la place continue P_3 vers la transition discrète T_1 est égale à la quantité sortante, donc il n'y a pas de consommation de marquage dans la place continue P_3 .

Transformation de marquage dans un RdPH

Dans figure 2-19 ' a', nous remarquons une transformation du marquage du discret vers le continu. Lorsque la transition T_1 est validée, deux jetons sont retirés de la place discrète P_1 et un marquage de 1,1 est déposé dans la place continue P_2. Dans la figure 2-19 b, il y a une transformation du marquage du continu vers le discret. Après le franchissement de la transition discrète T_1, 0,2 marques sont retirée de la place continue P_1 et un jeton est ajouté à la place discrète P_2.

Figure 2-19 Transformation du marquage de continu a discret et vice versa dans un RdPH

Définition 2.15

Un IB-état d'un RDPH représente des phases durant lesquelles l'évolution des marquages est constante et continue. Un IB-état d'un RDPH correspond à un intervalle de temps pendant lequel :

- Le marquage des places discrètes demeure constant.
- Le vecteur de vitesses instantanées des transitions continues reste constant, c'est-à-dire que les dérivées des marquages des c-places sont constantes.

Définition 2.16. Les réseaux de Petri hybrides RdPHs

Les réseaux de Petri hybrides sont une combinaison de nœuds (places et transitions discrètes et continues). Dans les places continues, le marquage est représenté par un nombre réel, tandis que dans les places discrètes, il est représenté par des jetons (David & Alla, 2001, 2010).

Un réseau de Petri est n-uplet $(P, T, h, Pre, Post, Tempo, I, M0)$, où:

$P = \{P1, P2, \dots, Pn\}$ est un ensemble fini de places, $P = PC \cup PD$, où:
PC : ensemble de places continues.
PD : ensemble de places discrètes.
$T = \{T1, T2, \dots, Tm\}$ est un ensemble fini de transitions, $T = TC \cup TD$, où:
TC : ensemble de transitions continues.
TD : ensemble de transitions discrètes.
$h : P \cup T \rightarrow \{D, C\}$, appelée 'fonction hybride' indique pour chaque nœud s'il est un nœud discret (PD et TD) ou un nœud continu (PC et TC).
Pre: Une matrice de pré-incidence.

$Post$: Une matrice de post-incidence

Les fonctions Pre et $Post$ doivent satisfaire au critere suivant : $\forall (Pi, Tj) \in PD \; x \; TC$:
$Pre\ (Pi, Tj) = Post\ (Pi, Tj)$

$M0$: Le marquage initial.

Définition 2.17 . Réseaux de Petri hybrides temporisés

Un réseau de Petri hybride temporisé est un couple (R, Tempo), où:

R: est un RdP hybride autonome marqué.

$Tempo$: est une fonction de l'ensemble des transitions T Si $T_j \in T_D$, $d_j = tempo(T_j)$ = temporisation associée a T_j . Si $T_j \in TC, Vj = \frac{1}{Tempo(Tj)}$ = vitesse de franchissement maximale associée a T_j.

Définition 2.18 . Réseaux de Petri hybrides temporels

Un réseau de Petri hybride temporel (RdPHT) est un triplet (R, I, U), où:

R: est un RdP hybride autonome marqué.
$I: T \rightarrow Q+$: La fonction de contraintes temporelles pour la transition T_j , $I\ (T_j)$= $[a_j, b_j]$, $a_j \leq b_j$, $0 \leq a_j \leq \infty$, $0 \leq b_j \leq \infty$.

$U \in \mathbb{Q}+$: La vitesse associée aux transitions continues.

2.2.5.1 Les RdPH D-élémentaires

Lorsqu'il n'y a pas de transformation de marquage du discret vers le continu et vice versa, les deux parties peuvent néanmoins s'influencer mutuellement. Dans notre travail, c'est uniquement la partie discrète qui commande la partie continue. Ce modèle est appelé réseau de Petri hybride D-élémentaire.

Les RdPH D-élémentaires ont été introduits dans (Ghomri & Alla, 2007). Ces modèles se construisent autour de l'intégration d'un RdP continu et d'un RdP T-temporel.

L'utilisation d'un RdP T-temporel pour représenter la partie discrète confère au modèle hybride un comportement non déterministe. Il convient de rappeler que, dans un RdP T-temporel, les dates de franchissement des transitions discrètes ne sont pas déterministes mais sont choisies dans des intervalles de franchissement, notant que ces transitions sont contrôlables.

Définition 2.19

Un RdP hybride D-élémentaire est défini par la structure $PNHD = (P, T, h, E, \Sigma, Pre, Post, U, V, M0)$. Les paramètres $P, T, h, E, \Sigma, V, M0$, sont définis comme dans le RdP hybride. De plus:

• $Pré\ et\ Post$ sont les applications d'incidence avant et arrière. Ces applications sont définis telles que :

$\forall\ (P_i, T_j) \in PC\ x\ TD\ , Pre(P_i, T_j) = Post\ (P_i, T_j) = 0$;

$\forall\ (P_i, T_j) \in PD\ x\ TC\ , Pré(P_i, T_j) = Post\ (P_i, T_j)$;

$U: TD \rightarrow R + x\ (R + \cup \{\infty\})$ associe à chaque D–transition T_j son intervalle de franchissement $[\alpha j, \beta j]$.

2.3 Introduction aux automates

Dans ce travail, notre objectif est de combiner la capacité de modélisation des Réseaux de Petri Hybrides (RDPH) avec la puissante analyse offerte par les automates. Nous détaillerons ensuite les caractéristiques fondamentales des automates, en mettant particulièrement l'accent sur les extensions pertinentes pour notre étude.

Nous explorerons d'abord les propriétés essentielles des automates temporisés avant de nous passer aux automates hybrides. En effet, ces derniers sont une extension des automates temporisés où la dynamique continue n'est plus limitée à être représentée par des horloges du type $\dot{x} = 1$, mais plutôt par des équations différentielles.

2.3.1 Les automates temporisés

Un automate temporisé est un modèle formel conçu pour modéliser et analyser le comportement des systèmes. Il s'agit d'une extension des automates à états finis qui intègre la notion de temps en utilisant des horloges, qui sont des variables continues par morceaux (Yovine, 1993).La dynamique des horloges dans un sommet de l'automate est décrite par une simple équation différentielle, telle que $\dot{x} = 1$, ce qui signifie que l'horloge mesure ou incrémente le temps passé dans ce sommet.

Dans chaque sommet de l'automate, on attribue un prédicat sur les valeurs des horloges, appelé invariant du sommet. Pendant le séjour dans un sommet, les valeurs des horloges doivent vérifier cet invariant.

Dans un automate temporisé, pour initialiser ou mettre à zéro une horloge x_i, on lui associe une affectation exprimée par $x_i := 0$. Selon les spécificités du modèle, il sera précisé si, lors du franchissement d'une transition T_j , l'horloge doit être initialisée ou non. De plus, chaque transition est associe à un prédicat sur les valeurs des horloges, appelé "garde". Parmi les conditions de franchissement d'une transition dans un automate temporisé, les valeurs des horloges doivent satisfaire ce prédicat. En effet, ce dernier assure le respect des contraintes temporelles du système modélisé.

Exemple :

L'automate temporisé présenté dans la figure 2-20 modélise une simple minuterie de cuisine qui démarre un compte à rebours de 15 secondes dès son activation. Une fois ce compte à rebours terminé, la minuterie se met à sonner. Cette sonnerie a une durée variant entre 2 et 5 secondes avant que la minuterie ne retourne à son état inactif.

L'automate temporisé en question est constitué de trois sommets L_0, L_1 et L_2, liés par des transitions modélisant l'occurrence des événements appropriés. Dans ce modèle, une horloge x est associée pour mesurer les contraintes temporelles imposées par le système. Elle est représentée par l'équation $\dot{x} = 1$ quand elle est activée dans un sommet, et $\dot{x} = 0$ dans le cas contraire.

Les gardes associées aux transitions et les invariants liés aux sommets garantissent le respect des contraintes temporelles de la minuterie en fonction des valeurs de l'horloge.

Dans le modèle présenté, la condition initiale de l'horloge est $x = 0$. Dans le sommet initial L_0, l'horloge est désactivée $\dot{x} = 0$ car il n'y a pas de contraintes temporelles dans cet état, correspondant à la désactivation de la minuterie, le système demeure dans cet état jusqu'à

l'activation de la minuterie ce qui est interprété par l'absence de l'invariant dans ce sommet. Lorsque la minuterie est activée, le système transite vers le sommet L_1. Dans cet état, l'occurrence de l'événement "activer" déclenche l'activation de l'horloge $\dot{x} = 1$, commençant ainsi l'incrément de temps. Le système est contraint de séjourner exactement dans L_1 pendant 15 secondes, ceci étant garanti par l'invariant $x <= 15$ associe au sommet L_1 et la garde $x >= 15$ associée à la transition T_2, Cela signifie que le système ne peut pas rester dans L_1 plus de 15 secondes mais doit également y rester au moins 15 secondes.

Après ces 15 secondes, l'événement "temps écoulé" est déclenché, réinitialisant l'horloge en raison de l'indépendance des deux contraintes temporelles, $x := 0$. Le système migre alors vers l'état sonore. Dans cet état correspondant au sommet L_2 de l'automate, l'invariant $x \leq 5$ garantit que le système y reste au plus 5 secondes. Toutefois, la garde $x \geq 2$ permet au système de quitter cet état après au moins 2 secondes.

Enfin, après cette phase sonore, le système revient à son état initial, L0, avec une initialisation de l'horloge lors de la transition de L_2 vers L0.

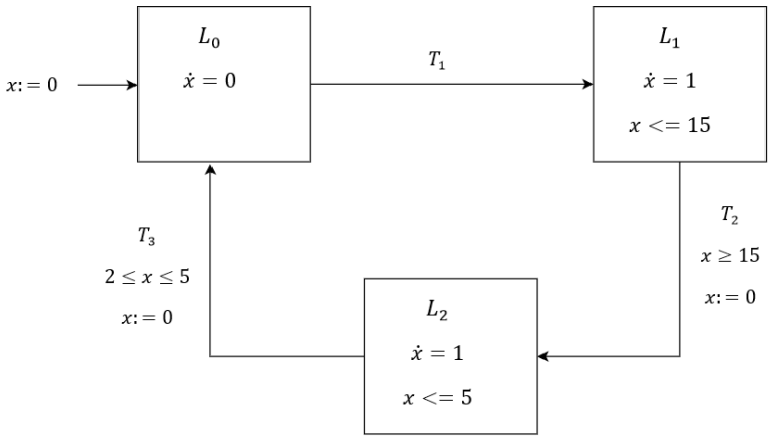


Figure 2-20 Automate temporisé : Système de minuterie

Dans ce qui suit on va présenter, une définition d'un automate temporisé, et des propriétés des horloges (Alur, 1999).

Définition 2.20 un automate temporisé

Considérons des horloges $x1, x2 \epsilon X$, et une constante rationnelle $c \epsilon Q$ et une relation d'ordre $< \epsilon\{<, <=, >=\}$. $g1, g2$ sont des contraintes sur la valeur des horloge .Une contrainte g sur la valeur des horloges est définie par une des expressions suivantes :

$g := x1 < c$,

$g := x1 - x2 < c$,

$g := g1\ g2$,

$g := g1$,

On note Gx l'ensemble de toutes des contraintes sur la valeur des horloges.

Définition 2.21 Une valuation des horloges est une fonction $v : X \rightarrow R+$, qui associe un nombre réel positif à chaque horloge x.

Cela signifie qu'une valuation des horloges associe une valeur v de chaque horloge à un instant précis.

L'ensemble des valuations des horloges $x \epsilon X$ est noté Vx. Si une valuation v vérifie une contrainte g, on écrit v I=g.

Définition 2.22 une affectation, notée Am, n, représente l'ensemble des horloges qui seront initialisées par le franchissement d'une transition $T_{m,n}$. Elle est exprimée par un ensemble d'équations de type $xi := 0$.

Définition 2.23 Un automate temporisé est défini comme un 6-uplet $(L, L0, X, \sum, I, T)$, où :

L : est un ensemble fini des sommets

$L0\ \epsilon\ L$:est le sommet initial

X :représente un ensemble fini des horloges

$\sum$:est un ensemble de symboles.

I :est une application qui associe à chaque sommet Lm, un invariant du sommet notée $I(Lm)$

T est l'ensemble des transitions. Une transition est défini par un 5 uplet $(Lm, a, g(m, m+1), A\ m, m+1, Lm+1)$, où :

Lm: est le sommet source,

a: est un symbole associe à un évènement,

$g(m, m+1)$ est la condition de franchissement (La garde), $A(m, m+1)$ est la fonction d'affectation.

$Lm+1$: est le sommet destination.

Définition 2.24: L'état d'un automate temporisé est représenté par le couple (Lm, v), où :

Lm : désigne le sommet et v est une valuation d'horloges qui vérifie l'invariant du sommet $I(Lm)$.

On peut séjourner dans sommet Lm tant que la valuations d'horloges vérifiée l'invariant $I(LM)$.

Cependant, un sommet peut ne pas représenter seulement un état unique, mais plusieurs. Cela est dû aux différentes valuations que peuvent prendre les horloges lors d'un séjour dans un

sommet Lm. On appelle cet ensemble d'états dans un sommet Lm par une région, notée (Lm, Qm), où Qm est un espace d'horloges dans le sommet Lm.

2.3.1.1 Analyse d'atteignabilité d'un automate temporisé

Pour vérifier si une région Q est atteignable depuis une région initiale $Qinit$, deux methodes peuvent être utilisées (Alur & Dill, 1994; Yovine, 1993).

La première est la méthode d'analyse en avant, où l'on calcul l'espace d'états qui peut être atteint depuis les états appartenant à la région $Qinit$, cet espace est défini par les successeurs de la région $Qinit$.

La deuxième est la méthode d'analyse en arrière. Dans cette méthode, on calcul l'ensemble de tous les états à partir desquels on peut atteindre les états de la région Q (les prédécesseurs de la région Q), si l'on trouve des espaces d'états qui appartiennent également à la région $Qinit$ alors la région Q est atteignable depuis $Qinit$.

Ces méthodes d'analyse ont été implémentées dans des logiciels dédiés à la vérification des systèmes temporisés et hybrides, comme kronos. (Yovine, 1997).

2.3.2 Les automates hybrides AHs

Les automates hybrides sont définis comme des automates à états finis intégrant des variables continues.(Raskin, 2005). Tandis que dans (Alur et al., 1995; Henzinger, 1996), ils sont illustrés sous la forme d'un graphe composé d'arêtes et de sommets, illustrant à la fois la dynamique continue et discrète au sein d'un même modèle. La partie discrète est symbolisée par le changement de sommet tandis que la dynamique continue est représentée par des équations différentielles associées à chaque sommet.

2.3.2.1 La syntaxe d'un automate hybride AH

L'automate hybride est un outil de modélisation et d'analyse, il est considéré comme une extension des automates temporisés. Dans ce contexte, la dynamique continue n'est plus uniquement représentée par des horloges, comme dans les équations d'état de type $\dot{x} = 1$, mais également par des équations différentielles, visant à modéliser et analyser divers types de systèmes, en particulier les systèmes hybrides.

On peut le définir aussi comme une fusion entre un automate à états finis et un ensemble d'équations différentielles décrivant le comportement du système à un instant précis.

L'adoption de l'automate hybride pour la modélisation et l'analyse offre une compréhension approfondie du comportement des systèmes ayant à la fois des composantes continues et discrètes. Grâce à sa capacité à combiner des commutations discrètes et processus continus, il est fréquemment employé dans la modélisation de systèmes tels que les systèmes biologiques, physiques, de gestion du trafic urbain et d'autres systèmes complexes. De plus, il est reconnu comme un outil puissant pour la vérification des systèmes hybrides ainsi que pour la synthèse de contrôleurs.

2.3.2.2 Sémantique d'un automate hybride AH

L'état d'un automate hybride est caractérisé par la paire (q, x), où q symbolise l'état discret du système et x désigne la valeur du vecteur d'état.

La sémantique d'un AH est décrite en fonction de l'ensemble des comportements possibles générés par l'évolution du système modélisé. Cette évolution au sein du modèle AH est caractérisée par le franchissement des transitions discrètes, traduisant ainsi un changement d'état discret, et par les évolutions des équations différentielles qui représentent la fonction des flux.

Une trajectoire dans un automate hybride est représentée par la séquence suivante :

$(q0, \delta0, x0) \quad \rightarrow \quad (q1, \delta1, x1) \qquad \rightarrow \qquad \ldots\ldots\ldots. \quad \rightarrow \quad (qi, \delta i, xi) \qquad \ldots.$

$\delta : (\delta \epsilon\ R+)$ définit la durée pendant laquelle l'automate reste dans l'état qi, cette valeur est initialisée à chaque commutation. Par ailleurs, xi est la fonction vectorielle qui décrit l'évolution de l'état continu durant cette période.

2.3.3 Les automates hybrides linéaires AHL

Dans le cadre de ce travail, notre attention est portée principalement sur les automates hybrides linéaires, car notre cas d'étude concerne un système linéaire. Les automates hybrides linéaires présentent des particularités spécifiques on va les détailler dans le suivant.

Les automates hybrides linéaires sont une sous-classe des automates hybrides. Dans ces automates, la dynamique des variables continues est définie par des inégalités différentielles linéaires.

Un automate hybride est linéaire s'il vérifie les conditions suivantes (Henzinger et al., 1995):

Linéarité : dans un automate hybride linéaire, la condition de flux, la condition d'invariance ainsi que la condition initiale, sont des prédicats linéaires convexes. Donc ils sont définis par des inégalités linéaires sur l'ensemble des variables (Alur et al., 1993).

Indépendance de flux : pour chaque mode de commande, la condition de flux est un prédicat se rapportant uniquement aux variables de X et ne contient aucune variable de X. Par exemple, des conditions de flux de la forme $x = x$, ne sont pas admises.

D'autres définitions proposées par des auteurs définissent l'automate hybride comme suit :

Un automate hybride doit satisfaire la caractéristique principale suivante : la fonction décrivant les évolutions des variables dans un sommet L de l'automate est formulée par des équations différentielles du premier ordre sous la forme $x^{\cdot} = kl$, où kl est un vecteur constant (David & Alla, 2001). Le problème d'accessibilité d'un automate hybride linéaire est indécidable (Henzinger et al., 1995).

Définition 2.25

Un automate hydride linéaire a sept-uplet $< X , L, \ F , Inv, T, Guard, As >$, où:
X : est un ensemble fini de variables à valeurs réelles.
L: est un ensemble fini de sommets.

F : est la fonction qui décrit l'évolution des variables dans chaque sommet. Dans l'automate hybride linéaire, elle est de type $\dot{x} = k$ où $k \in$ Q.
Inv : est la fonction qui associe à chaque sommet $l \in L$ un prédicat sur les variables, appelé invariant de sommet.

T : est un ensemble fini de transitions. Chaque transition $t = (l, l') \in T$ est définie par un sommet source l, et un sommet destination l'.

$Guard$: est une fonction qui assigne à chaque transition $t = (l, l') \in T$ un prédicat appelé garde. La transition ne peut pas être franchie si la valeur des horloges ne satisfait pas la garde.

As: est une fonction qui attribue à chaque transition $t\ (l, l') \in T$ une relation appelée affectation, elle est utilisée pour mettre à jour les variables d'états après le franchissement de la transition t.

Dans ce qui suit, nous allons détailler les fonctions associées à un sommet dans un automate hybride : la fonction de flux et la fonction d'invariant.

$\boldsymbol{Flux}\ (L_i)$: Désigne la fonction attribuant à chaque sommet une description de l'évolution de la dynamique continue. Lors du séjour dans un sommet L_i de l'automate hybride, l'évolution des variables continues est exprimée sous la forme d'une équation d'état de type $flux\ (Li)$: $\dot{x} = \phi(Li)(t, x, u)$, ou $x \in X \subset R^n, u \in U \subset R^p$ et $\phi : X \times U \to X$. x est le vecteur d'état continu du système et u est le vecteur d'entrée (commande)

L'invariant $(\boldsymbol{inv(Li)})$: est une fonction qui associe à chaque sommet $L_i \in L$ une contrainte sur les variables d'´états continues. Le système peut séjourner dans un sommet tant que l'invariant attribuée à ce sommet est satisfait.

Contrairement aux automates temporisés, les invariants dans les automates hybrides peuvent inclure des contraintes non seulement sur le temps, mais aussi sur d'autres variables continues. Ces contraintes peuvent être représentées par des équations impliquant des variables discrètes et continues.

En résumé, les invariants dans les automates temporisés sont principalement axés sur les contraintes temporelles, tandis que les invariants dans les automates hybrides englobent des contraintes à la fois sur les variables temporelles et autres variables continues.

2.3.4 Analyse et vérification d'un automate hybride

Pour comprendre l'évolution et le comportement d'un système modélisé par un automate hybride, il est essentiel d'effectuer une analyse d'atteignabilité, et de vérifier les propriétés de ce modèle. Cette analyse permet de vérifier tous les états que l'automate peut atteindre à partir d'un état initial.

L'automate hybride, tel que nous l'avons décrit, comporte deux dynamiques : une dynamique discrète, caractérisée par le franchissement des transitions discrètes et une dynamique continue, où les variables d'état évoluent de manière continue au fil du temps.

L'analyse d'un système à l'aide des automates hybrides nécessite plusieurs étapes. La première étape consiste en la modélisation du système en question. Cette modélisation vise à représenter de manière fidèle le fonctionnement du système, tout en intégrant les différentes interactions possibles. Une fois le système modélisé, l'étape suivante est la vérification de ses propriétés. Il s'agit notamment d'interpréter physiquement ces propriétés. Par exemple, la stabilité d'un système est une propriété essentielle qui doit être vérifiée pour s'assurer de son bon fonctionnement dans divers scénarios.

Enfin, un des défis majeurs de l'analyse d'un système à l'aide des automates hybrides est l'analyse d'atteignabilité. Celle-ci permet de déterminer quels sont les états que le système peut atteindre, à partir d'un état donné, en respectant les contraintes de l'automate.

En résumé, les automates hybrides offrent un cadre de travail structuré pour la modélisation, l'analyse et la vérification des propriétés des systèmes complexes. Cependant, malgré leur efficacité, certaines difficultés, comme l'analyse d'atteignabilité, demeurent des problématiques centrales à résoudre.

2.3.4.1 L'analyse d'atteignabilité

L'analyse d'atteignabilité à partir d'un espace d'état initial (q, X), où q est un sommet et X un espace d'état, est défini comme étant l'ensemble des états visités par toutes les exécutions commençant à partir de (q, X).

Ces exécutions résultent de deux types de dynamiques : un évènement discret provoqué par le franchissement d'une transition discrète et à la suite de cet évènement, on quitte le sommet q pour migrer vers un nouvel état ou sommet discret, que l'on nomme "successeur discret". D'autre part, il y a l'évolution continue qui s'opère avec l'écoulement du temps sans transition discrète. Si on demeure dans le même sommet tout en laissant le temps s'écouler, les états atteints au cours de cette évolution sont appelés "successeurs continus".

Dans le suivant, nous présenterons la définition des concepts de successeur continu et de successeur discret.

Définition 2.26 .successeur continu

Soit un ensemble d'état (q, X) où $q \in Q$ et X un espace d'état continu. On définit l'ensemble des successeurs continus de (q, X), qu'on note$R_{cont}(q, x)$, comme suit :

$$R_{cont}(q, x) = \{(q, x')/\ \exists\ x \epsilon X, \exists t > 0, (q, x) \xrightarrow{t,\ f_q} (q, x')\}$$

Définition 2.27 successeur discret

Soit une transition T, dont la garde est g et la fonction de réinitialisation γ, reliant le sommet q à q' et X un espace d'état continu, où $q, q' \in Q$. On définit l'ensemble des successeurs discret de (q, X), par rapport à la transition T, qu'on note $R_{dis}(q, x)$, comme suit :

$$R_{dis}(q, x) = \{(q', x')/\ \exists\ x \epsilon X \cap g\ \wedge x' \epsilon \gamma \cap\ inv(q')\}$$

2.3.4.2 L'automate atteignable

Comme mentionné précédemment, l'analyse d'atteignabilité vise à déterminer l'ensemble des états qu'un automate peut atteindre depuis un état initial. Elle englobe le calcul des états accessibles, que ce soit par des évolutions discrètes ou continues, en se basant sur des conditions initiales définies. L'issue de cette analyse est la construction d'un automate atteignable représentant toutes les trajectoires possibles du système depuis un état initial.

La première étape de cette analyse est de sélectionner un formalisme pour représenter les régions atteintes. Bien que de nombreux formalismes soient discutés dans la littérature, la représentation par polyèdre est largement utilisée pour sa simplicité de description et d'utilisation.

Déterminer l'espace des états atteignables est une tâche complexe et souvent indécidable, comme affirmé par les références (Henzinger et al., 1995; Puri & Varaiya, 1994). C'est pour cette raison qu'on tend souvent à établir une surestimation de cet espace.

L'approche couramment utilisée pour déterminer l'ensemble des états accessibles repose sur une méthode de calcul de point fixe. À partir d'un ensemble des états initiaux, on ajoute

progressivement les successeurs discrets, jusqu'à ce que l'ensemble des états atteints converge vers un point stable.

2.4 Conclusion

Dans ce chapitre, nous avons examiné les différents types de systèmes physiques, tel que les systèmes discrets, continus et hybrides, afin de mieux comprendre la nature du système pipeline multi-produit. Cette étude a révélé que le comportement du pipeline, caractérisé par des événements discrets et des dynamiques continues, s'inscrit dans la catégorie des systèmes dynamiques hybrides. Nous avons exploré diverses méthodologies de modélisation, d'analyse et de commande adaptées à ces systèmes, éclairant ainsi notre démarche méthodologique. Dans la suite, nous avons présenté les outils de modélisation et d'analyse utilisés dans notre travail, notamment les réseaux de Petri et les automates, ainsi que leurs diverses extensions. Les aspects divers des réseaux de Petri, y compris leurs caractéristiques temporelles et hybrides, ont été expliqués à l'aide d'exemples simples. Les réseaux de Petri hybrides D-élémentaire se sont avérés particulièrement adaptés à la représentation précise du comportement du pipeline ASR. Nous avons aussi abordé les modèles d'automates temporisés et hybrides, en examinant leur comportement et évolution, ainsi que les méthodes d'analyse d'atteignabilité des automates hybrides.

Le prochain chapitre se focalisera sur la modélisation du système de pipeline ASR à l'aide des réseaux de Petri hybrides, et sur la traduction de ce modèle en un automate hybride linéaire pour une analyse plus approfondie et formelle.

Chapitre 3

Modélisation, Analyse et Vérification du Pipeline Multi-Produit ASR à l'Aide des Réseaux de Petri Hybrides et des Automates Hybrides

L'objectif de ce chapitre est de développer un modèle qui décrit explicitement le comportement du système de pipeline ASR, prenant en compte à la fois les événements discrets et la dynamique continue. Nous utilisons l'outil Réseau de Petri, et présentons donc le modèle de réseau de Petri D-élémentaire qui décrit le comportement du pipeline multi-produit ASR. Pour une analyse plus formelle, nous convertissons notre modèle de réseau de Petri hybride D-élémentaire en automate hybride, menant à un automate hybride linéaire.
Grâce à l'outil PHAVer, nous réalisons une analyse d'atteignabilité, explorant tous les états potentiels du système. Cette analyse aboutit à l'élaboration du modèle d'automate atteignable pour le pipeline multi-produit ASR, une étape fondamentale pour le développement futur de notre contrôleur.

3.1 Présentation du modèle de réseau de Petri hybride D-élémentaire du pipeline multi-produit ASR

La figure 3-1 représente le modèle de pipeline multi-produit ASR avec une seule source (la raffinerie d'Arzew) et assure la livraison de deux dérivés pétroliers différents au dépôt de Remchi.

Nous allons examiner le système pipeline ASR décrit dans la section 1.3 du chapitre 1. Comme mentionné précédemment, notre étude se concentrera sur un cas réduit. Bien que notre démarche nécessite plusieurs étapes et implique un effort de calcul considérable, nous avons choisi de ne pas aborder un problème de trop grande envergure. Néanmoins, l'algorithme de synthèse développé dans cette étude peut également être appliqué à des problèmes de plus grande échelle et à d'autres systèmes hybrides D-élémentaires.

Concernant le cas étudié, nous prenons en compte une raffinerie approvisionnant deux produits pétroliers distincts, $pr1$ et $pr2$, vers un dépôt éloigné. Ce dépôt, doté d'une grande capacité de stockage, et couvre de vastes régions en produits pétroliers.

Le volume total du pipeline est de $10100\ m_3$, et nous supposons que les produits sont toujours disponibles à la raffinerie.

Les produits sont transportés via le même pipeline où un seul produit est pompé dans la canalisation à la fois. Nous considérons que la raffinerie dispose de deux électrovannes, la vanne 1 est dédiée à l'injection du $pr1$ et la vanne 2 au $pr2$. La vitesse d'injection ou le débit d'écoulement de la raffinerie vers le dépôt est identique, soit 200 m_3 par heure pour chaque produit, tout comme la vitesse de stockage du pipeline vers les réservoirs du dépôt car le pipeline est toujours sous pression.

Le débit de sortie des produits du dépôt vers les clients finaux est calculé en fonction des demandes quotidiennes, soit un volume $3000\ m_3$, de produit $pr1$ en 24 heures et $816\ m_3$, volume pour Pr_2 en 24 heures également, ce qui donne un débit de sortie pour $pr1$ égal à $125\ m_3\ /h$ et un débit de sortie pour $pr2$ égal à $34\ m_3/h$.

Nous avons pris en compte toutes les contraintes opérationnelles du système. Seules les spécifications non opérationnelles restent à définir. Pour simplifier les calculs ultérieurs, nous avons limité le nombre de lots dans le pipeline à un maximum de trois.

Lors du fonctionnement du pipeline, la décision d'ouvrir ou de fermer une vanne constitue un événement discret. Cet événement influence directement des paramètres tels que la date et la quantité de transport. Il est donc manifeste que la partie discrète commande la partie continue du système.

Cette décision est sous le contrôle de l'opérateur qui détermine à quel moment ouvrir ou fermer la vanne. Toutefois, l'instant précis de cette action demeure incertain. Par conséquent, on associe à ces événements un intervalle de temps [0, ∞[. Deux transitions temporelles sont utilisées pour mémoriser l'instant précis d'ouverture de chaque vanne.

Au niveau du centre de stockage et de distribution, le stock du produit 1 est représenté par la place continue P_{14} et celui du produit 2 par P_{16}. La procédure d'ouverture des vannes pour le stockage d'un produit est incontrôlable. En effet, dès que le produit atteint l'extrémité de la canalisation, l'opérateur doit ouvrir la vanne. Cela est dû à l'hypothèse selon laquelle le pipeline fonctionne en continu et doit toujours être sous pression.

Pour relever ce défi tout en respectant l'ordre d'injection et de réception, nous avons associé un délai, équivalent au rapport entre le volume total du pipeline et la vitesse d'injection, aux deux transitions incontrôlables responsables de l'approvisionnement au dépôt. Ainsi, dès l'ouverture d'une vanne à la raffinerie, cette transition est validée. Elle attend ensuite le délai prédéfini avant de devenir franchissable.

La partie continue du modèle de réseaux de Petri hybrides D-élémentaire

Les produits à la raffinerie sont considérés comme étant toujours disponibles, donc les transitions sources (T_9 et T_{12}) représentent respectivement l'injection du produit 1 ou du produit 2 dans le pipeline.

Les places continues (P_{13}, P_{15}) représentent l'état du pipeline (le volume du produit 1 et le volume du produit 2 à un moment donné dans le pipeline), par exemple : sachant que le volume total du pipeline est de $10100 m_3$, au temps $t = 0$, le pipeline est entièrement rempli de produit 1, donc le marquage m (P_{13}) = 10100 m_3 et m(P_{15}) = 0 m_3.

Les places (P_{14} et P_{16}) représentent respectivement le niveau de stock des produits Pr_1 et Pr_2 dans le dépôt.

Les transitions puits (T_{11} et T_{14}) alimentées par (P_{14} et P_{16}), c'est-à-dire par le stock de produits dans le dépôt, représentent les sorties des produits 1 et 2 vers les clients.

Note.

Les vitesses de franchissement des transitions T_{11} et T_{14} sont calculées en fonction des demandes quotidiennes de produits, $vpk(m_3/\text{h})$ = demande quotidienne du produit k (m_3) $/24h$, $k = 1..2$.

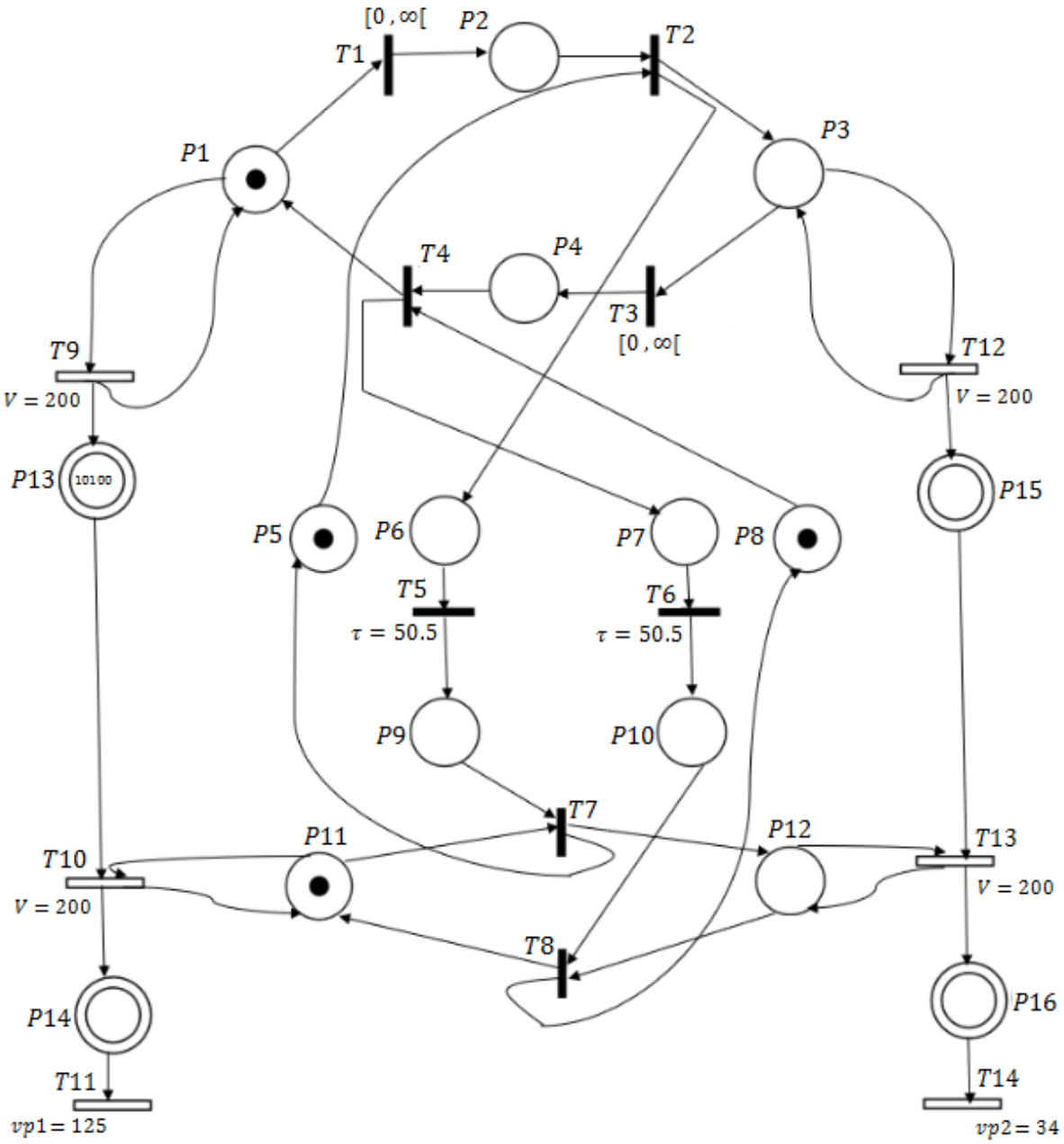


Figure 3-1 Modèle réseau de Petri D-élémentaire du pipeline multi-produit ASR

Partie discrète du modèle de réseaux de Petri hybrides (la partie commande)

- **L'injection des produits**

Par hypothèse, nous ne pouvons injecter qu'un produit à la fois, soit le produit $Pr1$ ou $Pr2$. Cette opération est représentée sur la figure 3-1 par la boucle ($P_1 - T_1 - P_2 - T_2 - P_3 - T_3 - P_4 - T_4$), qui permet de contrôler les électrovannes à la raffinerie en injectant les produits dans le pipeline : produit 1 si la place 1 est marquée ou produit 2 si P_3 est marquée.

Les transitions temporelles T_1 et T_2 sont respectivement responsables du passage de l'injection du produit 1 au produit 2, et inversement, Ces transitions permettent non seulement de mesurer le temps d'injection, mais aussi de déterminer le début et la fin de chaque opération. De plus, elles fournissent des informations sur le volume de produit injecté ainsi que sur sa position dans le pipeline.

P_2 et P_4 sont des places intermédiaires pour que les horloges associées aux transitions T_1 et T_3 soient indépendantes des places P_5 et P_8 . Par ailleurs Les places (P_6 , P_7) sont respectivement associe aux places (P_5 et P_8) qui ont un marquage binaire. Cette configuration limite le nombre de lots dans le pipeline à trois au maximum à un instant donné

- **La réception des produits au dépôt**

Les chaînes (T_2, P_6 , $T_5 - T_4$, P_7 , T_6) assurent la mise en ordre des produits dans le pipeline. Nous avons associé les transitions temporisées (T_5 et T_6) à un délai de franchissement noté (τ). Ce délai est calculé comme suit : τ = volume total du pipeline (m_3) / vitesse d'injection du produit (m_3/h) = 10100/200, soit $\tau = 50{,}5h$, à l'aide de ces transitions nous pouvons savoir quand le produit est arrivé à l'extrémité du pipeline, pour être ensuite stocké au dépôt.

P_9 et P_{10} sont des places intermédiaires de synchronisation, pour que les horloges associées aux transitions T_5 et T_6 soient indépendantes du marquage des places P_{11} et P_{12} .

3.2 Traduction du modèle de réseaux de Petri hybride en automate hybride

3.2.1 Du réseau de Petri à l'automate : état de l'art

Le couplage des réseaux de Petri pour la modélisation et des automates pour l'analyse est couramment utilisé dans la littérature. Ceci est dû aux avantages que présentent les réseaux de Petri en matière de modélisation, notamment leur facilité d'adaptation, leur capacité à décrire de manière exhaustive différentes dynamiques et leur clarté visuelle. Cependant, en termes d'analyse, les automates sont un outil très puissant comparé aux réseaux de Petri.

De nombreux auteurs ont suggéré des méthodes structurelles et des algorithmes pour convertir les réseaux de Petri hybrides en leurs automates équivalents (David & Alla, 2001; Ghomri et al., 2005; Sava & Alla, 2001). De plus, la translation des réseaux de Petri temporels en automates temporisés a également été explorée (Cassez & Roux, 2006). Je présente ici certains travaux qui ont influencé ma recherche.

Sava et Alla (Sava, 2001)dans leur travail consacré à la synthèse de commande des systèmes à événement discret, ont élaboré un algorithme pour traduire un réseau de Petri T-temporel en automate temporisé, pour ce faire, ils ont associé à chaque marquage du modèle RdP T-

temporel un sommet de l'automate temporisé, et chaque transition à une horloge; l'intervalle de franchissement de la transition spécifie la garde de l'automate.

Dans une approche similaire, (Cassez & Roux, 2006) ont mis en avant une stratégie de traduction structurelle pour traduire un réseau de Petri T-temporel en automate temporisé. Premièrement, ils ont caractérisé la sémantique du RdP-t temporel comme étant un système de transition temporisé, par la suite, pour chaque transition, ils ont introduit un automate temporisé munie d'une unique horloge. L'état de cet automate est déterminé par l'état de la transition, si elle est validée ou non et par sa capacité à être franchie, et ses conditions dans le marquage initial. Finalement, les automates résultants sont fusionnés avec un superviseur via un produit d'automates temporisés.

Concernant la translation du RdPh en automate hybride, je tiens à souligner certains travaux qui ont été instructifs à ma compréhension et à la mise en œuvre de cette traduction dans le cadre de mon travail. (Alla & David, 1998) ont développé une méthode pour traduire le RDPH en automate hybride. L'automate résultant possède le même nombre de sommets que le graphe IB-états du RDPH. Cependant, cette méthode est uniquement efficace si le RdPh est borné.

Ghomri (Ghomri, 2012), dans ses recherches, a introduit un algorithme permettant de convertir le RdPh D-élémentaire en automate hybride linéaire. Sa méthode repose principalement sur la séparation de la partie discrète de la partie continue du modèle. Dans un premier temps, elle a traduit la partie discrète du modèle RdP-temporel en automate temporisé. Ensuite, elle a intégré la partie continue RDPCC au RdP-temporel pour élaborer l'automate hybride.

3.2.2 Le principe de l'approche de traduction du RdPH-D-élémentaire vers l'AH

Le fonctionnement d'un réseau de Petri hybride est principalement défini par le vecteur d'évolution, qui demeure constant entre l'occurrence de deux événements discrets successifs, et par le marquage des places continues, qui peut varier continuellement au fil du temps même lorsque le vecteur d'évolution est constant. Ce comportement peut être représenté par une évolution des variables d'état au sein d'un sommet d'un automate hybride, où le vecteur d'évolution modélise un sommet dans cet automate.

Lorsqu'un événement se produit, tel que le franchissement d'une transition discrète, le vecteur d'évolution peut être modifié. Cette modification est comparable au franchissement d'une transition dans un automate hybride. Ainsi, le fonctionnement d'un RDPH peut être assimilé à celui d'un automate hybride avec ses composants distincts.

Notre étude se focalise sur un modèle de réseau de Petri hybride D-élémentaire, où la partie discrète commande la partie continue. Il est envisageable d'isoler cette partie discrète et de la traiter comme un réseau de Petri temporel. Une fois ce réseau traduit en automate temporisé, nous y intégrons les dynamiques continues décrivant l'évolution du système.

Notre approche se divise en deux étapes principales. La première consiste à construire le graphe d'évolution basé sur le RDP-T-temporel. Dans la seconde étape, pour chaque marquage du RDP-T-temporel, nous calculons la dynamique de marquage des places continues et l'associons au sommet discret correspondant.

Avant d'entamer la translation, il est essentiel de souligner les composants fondamentaux d'un automate hybride linéaire qui traduit le comportement d'un RDP hybride D-élémentaire :

Les horloges

A chaque transition discrète temporelle ou temporisée T_j on attribue une horloge C_j. Cette horloge joue le rôle essentiel de mémoriser le temps écoulé depuis la validation de cette transition. En d'autres termes, dès que la transition est validée, l'horloge est activée.

Il est crucial de rappeler que la transition ne peut être franchie que si certaines contraintes temporelles sont respectées. Dans notre modèle RDP, nous distinguons deux types de transitions. La première, nommée t-temporelle, est celle où T T_j appartient à un intervalle $[\, a_j, b_j]$. Dans ce cas, la transition doit être franchie au plus tôt à a_j et au plus tard à b_j. Le second type est la transition temporisée. Ici, le franchissement de la transition T_j se fait à un moment précis, c'est-à-dire lorsque la valuation du prédicat de la garde T_j est exactement égale à une valeur donnée d_j.

La dynamique de l'horloge est définie par $\dot{C}_J = 1$. Elle est activée lorsque la transition temporelle associée du RDP t-temporel est validée par le marquage Mi, et $\dot{C}_J = 0$.dans le cas contraire

Les sommets

Lors de la traduction vers un automate temporisé à partir d'un RDP T-temporel, chaque marquage Mi est représenté par un sommet Li. Dans ce sommet de l'automate, divers éléments sont intégrés, tels que la fonction flux, la fonction d'invariant et la dynamique des horloges.

Les transitions

À l'instar des sommets, chaque transition du RDP T-temporel est modélisée par une transition dans l'automate hybride. Le franchissement d'une transition dans un automate hybride dépend de la valuation de l'horloge associée au sommet source, à savoir la dynamique de l'horloge et l'espace des horloges à l'entrée de ce sommet. Ces valuations doivent vérifier un prédicat lié à la transition, appelé garde.

La garde

La garde d'une transition T_j se rapportant à la valeur de l'horloge C_j est exprimée sous la forme de l'expression $a_j \leq C_j \leq b_j$, si la transition est temporel ou $C_j = d_j$ si elle est temporisée. Chaque transition est également associée une fonction d'affectation permettant d'initialiser des horloges liées à la transition nouvellement validée.

La dynamique des places continues (Bilans)

La partie continue est définie par le bilan des dynamiques des places continues P_i, notées $\dot{m}_i$ .Cette dynamique est déterminée par la différence des vitesses de franchissement des transitions continues en amont et en aval, formulée comme $\dot{m}_i = V(Tpi\ amont)) - V(Tpi\ aval)$. Ainsi, pour chaque marquage du RDP T-temporel, nous calculons les bilans de toutes les places continues et les associons au sommet correspondant de l'automate.

Invariant du sommet

L'invariant associé à chaque sommet doit être respecté par la valuation des horloges pendant le séjour dans ce sommet. Cet invariant garantit le respect des contraintes temporelles associées à la transition discrète d'un RDP T-temporel.

Étant donné que la partie continue n'a aucune influence sur la partie discrète, il n'y a pas d'invariant pour les flux continus. L'invariant dans notre cas dépend de l'horloge active dans le sommet ainsi que de la transition de sortie. Il est important de changer de sommet lorsque le marquage d'une transition continue s'annule. Dans ce contexte, cette condition peut aussi être perçue comme un invariant. Cependant, en cas d'annulation du marquage de l'un de C-places il faudrait se diriger vers un autre sommet, la transition reliant ces deux sommets est étiquetée par m_i qui s'est annulé.

Les conditions initiales :

Les conditions initiales définissent les valeurs d'entrée du sommet initial. Elles se composent des valeurs prises par les horloges, désignées par l'espace d'horloges, et des dynamiques continues, représentées par l'espace des variables d'état, au moment où l'horloge globale $C0$ est égale à zéro. Il est important de noter que l'horloge globale n'est influencée par aucune transition. Toujours active, elle mesure et incrémente le temps écoulé depuis l'instant initial.

3.3 Translation du modèle RdPH D-élémentaire décrivant le fonctionnement du réseau de pipeline 'ASR' en automate hybride

Le modèle RdPH D-élémentaire qui décrit le fonctionnement du pipeline multi-produit ASR est présenté dans la figure 3-1.

Nous mettrons en œuvre les étapes de la méthode de translation en traduisant notre modèle RdPH D-élémentaire en automate hybride.

Notations utilisées

C_j : Horloge associe à la transition T_j , mesurant le temps écoulé depuis la validation de cette transition .

m_i : Marquage de C-Place (P_i).

$[a_j, b_j]$: Intervalle de franchissement pour la transition T_j , Étant donné que C_j est l'horloge responsable de mémoriser le temps écoulé depuis la validation de cette transition, la garde de la transition T_j est décrit sous la forme $a_j <= C_j <= b_j$.

d_j : Durée de temporisation de la transition T_j, la garde associe à la transition T_j est exprimée sous la forme $C_j = d_j$.

Les horloges

Nous commençons par associer les horloges aux transitions temporelles et temporisées du modèle RdPH D-élémentaire :

C_0 : Il s'agit de l'horloge globale. Dès l'initiation de notre simulation ou analyse, cette horloge est activée. Par conséquent, dans chaque sommet de l'automate, l'horloge $C0' = 1$.

C_1 : associe à la transition T_1 . La garde liée à cette transition dans l'automate est exprimée par $C_1 \epsilon [0, \infty[$.

C_2 : associe à la transition T_3. La garde pour cette transition est donnée par $C_2 \epsilon [0, \infty[$.

C_3 : associe à la transition T_5. La garde pour cette transition est $C_3 = 50.5$.

C_4 : associe à la transition T_6. La garde pour cette transition est $C_4 = 50.5$.

Construction du model AHL

1. Isolation des Parties Discrète et Continue :

La première étape est d'isoler la partie discrète de la partie continue. Nous atteignons cela en supprimant les boucles reliant les places discrètes aux C-transitions dans le RdPH D-élémentaire. Ainsi, nous distinguons la partie discrète, qui est un RDP T-temporel, du RdPCC qui décrit l'évolution de la partie continue.

2- Création du Graphe de Marquage Équivalent :

Comme notre modèle RDP T-temporel est borné, nous générons son graphe de marquage équivalent.

3- Construction du sommet initial (L_1) de l'automate hybride

Le Marquage initial du RdP T-temporel est le suivant : $(1,0,0,0,1,0,0,1,0,0,1,0)^T$. À partir de ce marquage, nous observons qu'une seule transition, T_1 , est validée. Nous savos que l'horloge est activée si la transition est validée et désactivée dans le cas contraire. La dynamique des horloges dans le sommet initial L_1 est :

$$\begin{pmatrix} \dot{C}_0 = 1 \\ \dot{C}_1 = 1 \\ \dot{C}_2 = 0 \\ \dot{C}_3 = 0 \\ \dot{C}_4 = 0 \end{pmatrix}$$

Les conditions initiales à l'entrée du sommet L_1 :

L'espace d'horloge à l'instant initial et les marquages des c-places (m_i) a l'entrée du sommet initial L_1 ,sont présentés comme suit :

$$\begin{pmatrix} C_0 = 0 \\ C_1 = 0 \\ C_2 = 0 \\ C_3 = 0 \\ C_4 = 0 \end{pmatrix} \quad , \begin{pmatrix} m_{13} = 10100 \\ m_{14} = 0 \\ m_{15} = 5572 \\ m_{16} = 4350 \end{pmatrix}$$

Pour plus de détails concernant les conditions initiales des horloges et l'espace d'état continu à $t = 0$, veuillez consulter le tableau 1.1 du chapitre 1.

Intégration du comportement du RDPCC dans le sommet initial

Nous intégrons le comportement du RDPCC dans le sommet initial, ce qui nous conduit à calculer les bilans de marquage de toutes les places continues (m_i)

Exemple pour la place continu P_{13} le bilan est calculé comme suit :

$$\dot{m}_{13} = V(T_9) - V(T_{10}) = 200 - 200 = 0$$

$$\begin{pmatrix} \dot{m}_{13} = 0 \\ \dot{m}_{14} = 75 \\ \dot{m}_{15} = 0 \\ \dot{m}_{16} = -34 \end{pmatrix}$$

Calcul de la garde du sommet initial

Dans le marquage initial du RdP T-temporel, une seule transition, T_1, peut être franchie dans l'intervalle $[0, \infty[$. Ainsi, la garde de la transition discrète T_1, reliant le sommet initial L_1 au sommet successeur désigné par L_2 dans l'automate hybride, est décrite par : $0 \leq C_1 \leq \infty$.

De plus, considérant qu'un événement lié à l'annulation du marquage d'une place continue peut modifier l'état discret d'un sommet, et que le marquage m_{16} du sommet initial est susceptible de s'annuler, un deuxième successeur pour le sommet L_1, nommé L_1', est identifié. La garde associée à ce successeur est donnée par : $m_{16} = 0$.

Calcul de l'invariant du sommet initial

L'invariant d'un sommet dans un automate hybride D-élémentaire est déterminé de la manière suivante : pour un sommet L_k, l'invariant noté $I(L_k)$ est donné $par\ C_j \leq b_j \wedge \ldots \wedge C_n \leq b_n \wedge m_i \geq 0$. Donc, pour toutes les transitions sortantes de ce sommet $T_j \ldots T_n$, la valeur de l'horloge associée à chaque transition doit être inférieure ou égale à la date au plus tard à laquelle cette transition doit être franchie. De plus, si le marquage d'une place continue Pi peut atteindre zéro dans ce sommet, on ajoute un invariant $m_i \geq 0$.

Dans notre cas, l'invariant est donné par $C_1 < \infty$ (cette condition est toujours vraie, donc elle peut être éliminée) et m_{16} peut s'annuler, donc $I(L_1)$ est $m_{16} \geq 0$ . .

4- Construction des sommets successeurs du sommet initial

Le nombre de sommets successeurs dépend du nombre de transitions qui peuvent être franchies dans le sommet prédécesseur et du nombre de mi qui peuvent atteindre zéro. Bien que les conditions initiales du sommet initial soient bien définies, en évoluant dans le comportement du RDPH et par conséquent de l'automate hybride, ce comportement devient indéterministe en raison de l'intervalle de franchissement associé aux transitions discrètes. De ce fait, nous ne pouvons pas prévoir quelle transition sera franchie avant l'autre ou quelle dynamique sera annulée avant l'autre.

Dans notre exemple, nous avons deux sommets successeurs pour le sommet initial, à savoir les sommets L_2 et L_1'. L_2 résulte du franchissement de la transition T_1, tandis que L_1' découle de l'annulation du marquage de la place P_{16} , m_{16} =0.

Le sommet L_2 correspond au marquage du RdP T-temporel résultant du marquage initial après le franchissement de la transition T_1. Ce sommet est généré de la même façon que le sommet L_1, en se basant sur le marquage du RdP T-temporel associé (le marquage discret successeur du marquage initial, obtenu après le franchissement de la transition T_1).

Si de nouvelles transitions sont validées pour ce marquage, on initialise les horloges associées à ces transitions. On appelle cela la fonction d'affectation. Dans ce cas, les transitions T_2 et T_3 sont nouvellement validées, donc on écrit : $C_2 := 0 \; et \; C_3 := 0$.

Le second sommet successeur, $L_1{}'$, est généré par l'annulation du marquage de la place P_{16} ,

m_{16} =0. Ce sommet est considéré comme un duplicata du sommet L_1, à l'exception que le bilan du marquage de la place continue qui s'annule sera égal à 0. Donc, on conserve les mêmes valeurs pour L_1, mais avec $\dot{m}_{16}$=0 au lieu $\dot{m}_{16}$=-34. (figure 3-2)

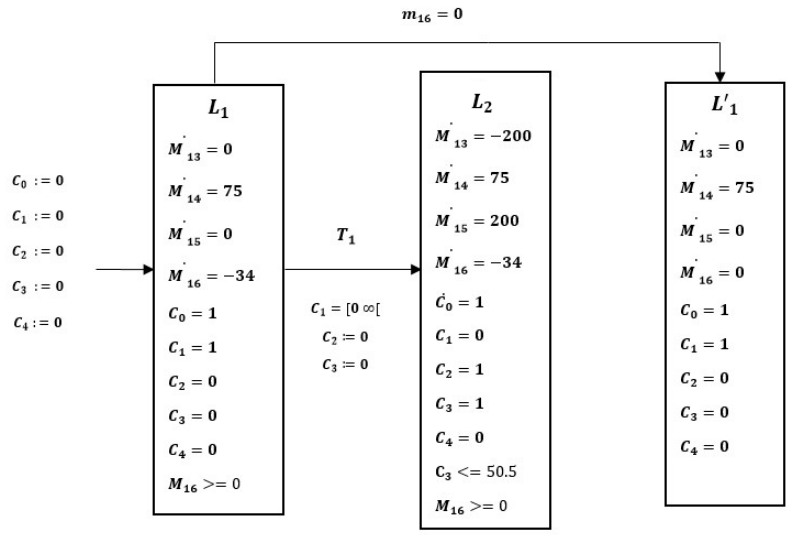


Figure 3-2 Sommet initial et ses successeurs discrets.

Remarque 3.1.

Le sommet successeur, étiqueté par un marquage mi qui s'annule, présente le même comportement que le sommet prédécesseur. Ce sommet successeur correspond également au même marquage du RDP T-temporel que le sommet prédécesseur. La principale distinction réside dans le fait que, pour le sommet successeur, le bilan de mi doit être égal à 0. Lors de la création de nouveaux successeurs pour ce sommet spécifique, si le marquage associé indique

un bilan mi négatif, mi doit rester égal à 0. Toutefois, si le bilan pour un marquage successeur devient positif, on retient alors la valeur du bilan déduite de ce marquage.

Remarque 3.2.

L'annulation du marquage des places m_{13} et m_{15} n'est pas modélisée dans l'automate hybride. En effet, elles s'annulent au même instant lors du franchissement des transitions incontrôlables T_5 ou T_6. Il n'est donc pas nécessaire de les modéliser.

En pratique, ils correspondent au même événement, donc il n'est pas nécessaire de les représenter séparément. Par exemple, la fin du stockage du produit Pr_k dans le dépôt coïncide avec le moment où le volume de ce produit dans le pipeline atteint zéro.

3.3.1 Présentation du modèle AHL issu de la traduction du modèle RdPH D-élémentaire du pipeline multi-produit ASR

L'automate hybride, issu de la traduction du RdPH D-élémentaire et modélisant le comportement du pipeline multi-produit ASR, est présenté dans la Figure 3-3.

La figure 3-4 représente l'automate final après élimination des sommets intermédiaires où le temps de séjour est égal à 0 ainsi que des sommets interdits.

Les sommets intermédiaires sont : L_2, L_4 , L_5, L_8 , L_{10} , L_{11}, L_{13}, L_{14}, L_{17} , L_{18} .

Les sommets L_9 et L_{16} sont interdits et correspondent aux marquages suivants :

L_9 (P_2 , P_6 , P_7 , P_{11}) : cet état décrit l'arrêt de l'injection du produit Pr_1 tout en continuant la réception du produit Pr_1 (le pipeline n'est pas sous pression selon notre hypothèse).

L_{16} (P_4 , P_6 , P_7 , P_{12}) : cet état décrit l'arrêt de l'injection du produit Pr_2 tandis que la réception de Pr_2 continue (le pipeline n'est pas sous pression selon notre hypothèse).

Remarque 3.3 : Supposant que les sommets l_n et l_n sont atteints depuis le sommet l_m en franchir respectivement la transition contrôlable$0 \leq T_{m,n} < \infty$, et la transition non contrôlable $T_{m,q} = d_j$, avec d_j étant un nombre fini. Dans ce cas, si l_n est un sommet interdit, nous pouvons l'éliminer sans affecter ni modifier l'espace d'état de notre système.

Note. Bien que nous ayons la capacité d'éliminer tous les états interdits en utilisant l'algorithme de synthèse de contrôleur développé dans le cadre de notre recherche, notre objectif est de simplifier l'automate pour minimiser la complexité des calculs, tout en offrant une explication claire et détaillée des étapes de l'approche de synthèse de contrôleur que nous proposons à travers cet exemple. De ce fait, j'ai retiré les sommets pour lesquels l'intégralité de l'espace d'état est considérée comme interdite.

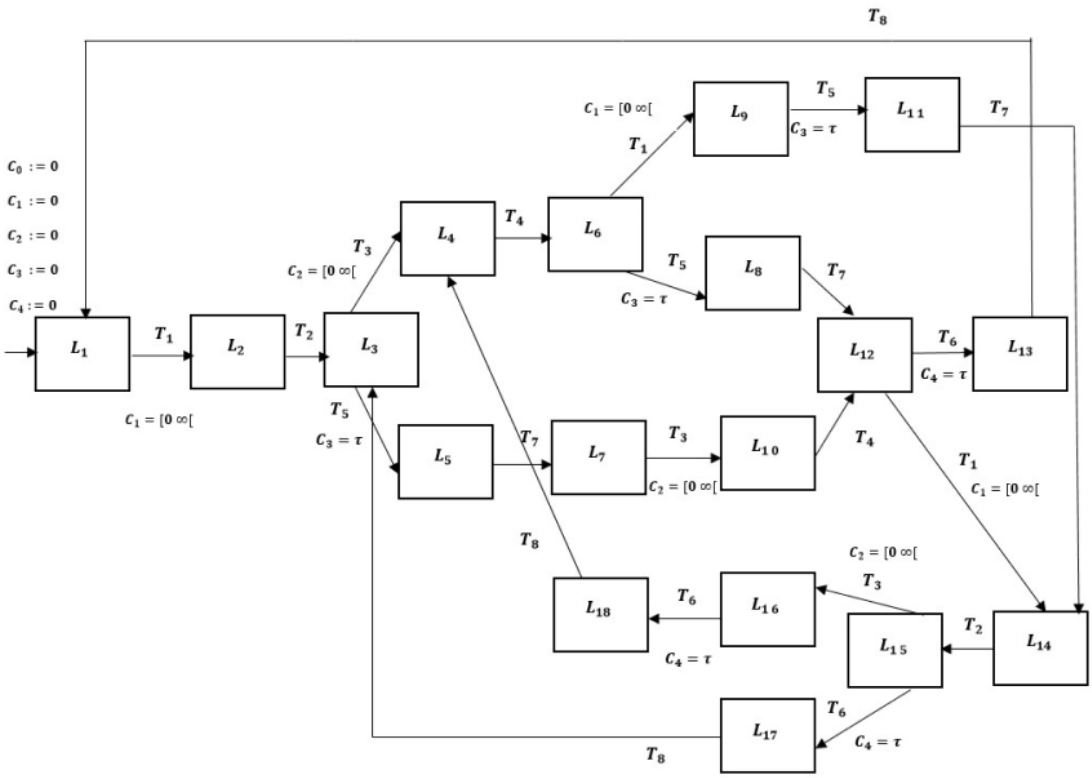


Figure 3-3 L'automate hybride résultant de la traduction du RdPH D-élémentaire de la Figure 3-1

3.3.2 Particularités de l'automate hybride issu de la traduction du RdPH D-élémentaire

La méthode de traduction adoptée ici conduit à un automate hybride linéaire avec les caractéristiques suivantes:

Nous disposons de deux types de variables d'état continues: m_i, C_j. Où les horloges C_j correspondent à la durée de validation des transitions discrètes dans le modèle RdPH D-élémentaire. Quant aux variables mi, elles représentent les marquages des places continues dans le modèle RdPH D-élémentaire.

Les horloges C_j sont susceptibles d'être remises à zéro, en revanche, les variables mi ne peuvent pas être initialisées et ne subissent donc aucun saut. m_i

La garde de chaque transition dépend uniquement d'une variable réelle et peut prendre l'une des formes suivantes:

$m_i = 0$: Si la transition modélise l'annulation du marquage d'une c-place dans le RdPH D-élémentaire.

$a_j \leq C_j \leq b_j$:Si la transition modélise le franchissement d'une d-transition temporelle.

$C_j = d_j$: Si la transition modélise le franchissement d'une transition temporisée.

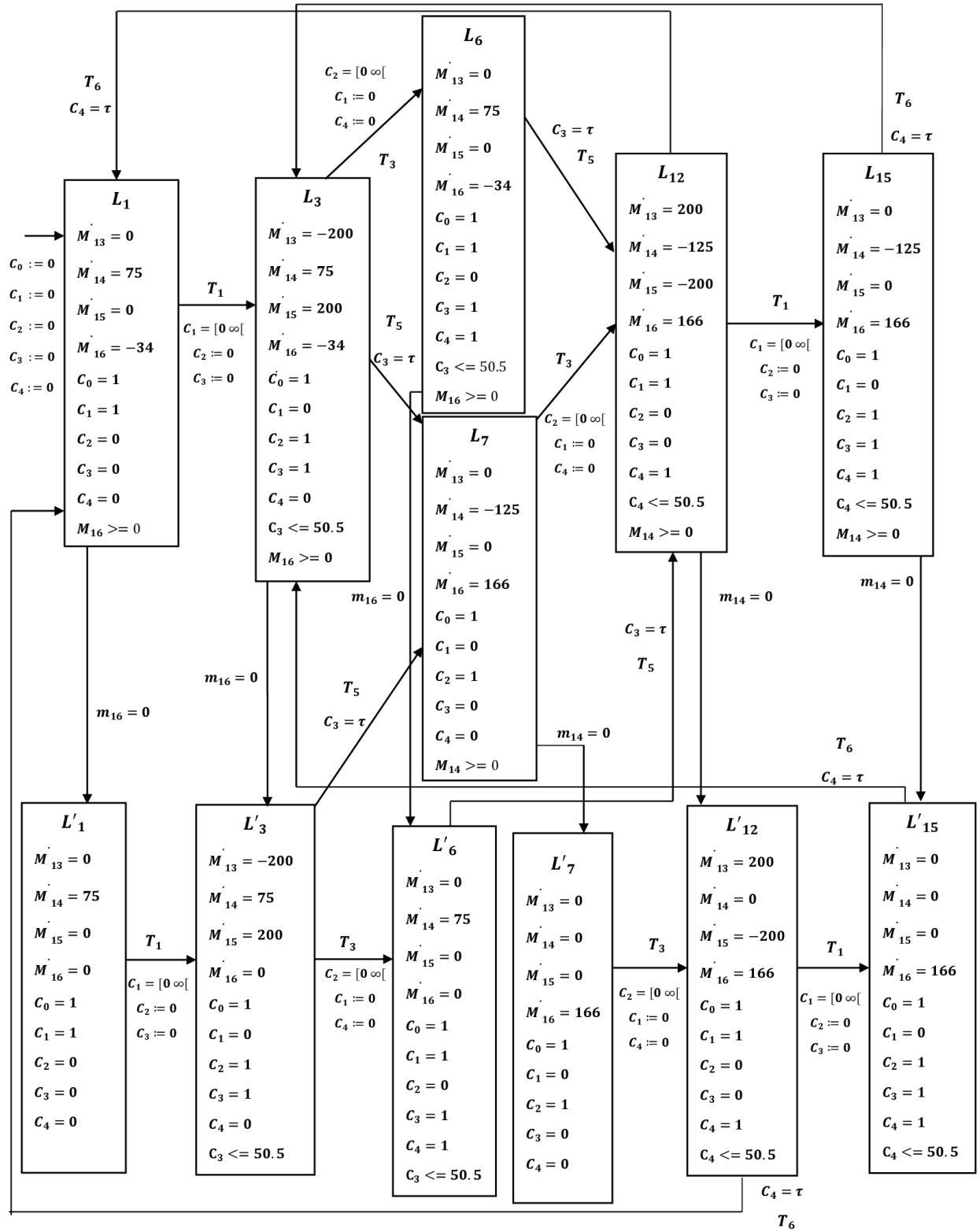


Figure 3-4 Automate hybride final correspondant au d RdPH D-élémentaire (Figure 3-1)

3.4 L’automate atteignable du système pipeline muli-produit ASR

3.4.1 L'analyse d'atteignabilité des automates hybrides avec PHAVer

PHAVer (Polyhedral Hybrid Automaton Verifier) est un outil conçu pour la vérification des propriétés de sûreté des systèmes dynamiques hybrides linéaires par morceaux. Il a été développé par Goran Frehse (Frehse, 2005). La vérification de ces propriétés est étroitement liée à l'analyse d'atteignabilité des automates hybrides, qui est souvent un problème indécidable. Dans cette perspective, PHAVer fournit une sur-approximation des dynamiques affines à l'aide d'automates hybrides linéaires. Pour effectuer des calculs précis sur les polyèdres, il s'appuie sur la bibliothèque Prama Polyhedra. Bien qu'il partage certaines similitudes avec HYTECH, PHAVer est capable d'analyser des systèmes plus complexes.

L'un des avantages de PHAVer est qu'il fournit l'espace des états atteignables pour chaque sommet sous une forme analytique, en présentant des inégalités entre les différentes variables continues. Cette représentation analytique sera exploitée ultérieurement pour déduire l'automate atteignable.

Programme PHAVer pour notre automate hybride linéaire

Nous présentons ici un aperçu du programme PHAVer pour notre modèle d'automate hybride. Le programme intégral est illustré en Annexe. Nous commençons par définir les variables d'états continus, puis les transitions discrètes. L'analyse du modèle s'étend sur 100 heures, car la convergence n'est pas atteinte en un temps déterminé. Par conséquent, nous intégrons à notre programme un sommet, appelé "sommet final", où toutes les variables prennent la valeur de 0.

De plus, pour chaque sommet de l'automate, nous ajoutons un invariant de la forme $C_0 \leq 100$. Nous introduisons également une transition depuis chaque sommet vers le sommet final, avec une garde définie par $C_0 = 100$. Cette démarche vise à stopper la computation de l'espace d'état une fois que l'horloge globale atteint 100 heures. Ci-après, nous offrons un aperçu de la manière de décrire le programme PHAVer.

```
automaton pipeline

state_var: C0,C1, C2, C3, C4, m13, m14, m15, m16;

synclabs: T1, T3, T5, T6,T7,T8,T9 ,T10 ,T11 ,T12 ,T13;

loc L1: while m16>=0  & C0<=100  wait {m13'== 0 & m14'== 75 & m15'==0 & m16'== -34 & C0'== 1 &
C1'== 1 & C2' == 0 & C3' == 0 & C4'==0}

        when C1 >= 0  sync T1 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0
& C1'== C1 & C2'== 0 & C3'== 0 & C4'==C4} goto L3;

         when m16 == 0  sync T7 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'==
C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L1x;
```

```
when C0>=100  sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 &
C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;


loc L3: while C3 <= 50.5 & m16>=0  & C0<=100  wait {m13'== -200 & m14'== 75 & m15'== 200 & m16'==
-34 & C0'==1 & C1'== 0 & C2' == 1 & C3' == 1 & C4'==0}

when C2 >= 0   sync T3 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'==
C0 & C1'== 0 & C2'== C2 & C3'== C3 & C4'==0} goto L6;

when C3 == 50.5   sync T5 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 &
C0'== C0 &  C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L7;

when m16 == 0  sync T8 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 &
C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L3x;

when C0>=100  sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 &
C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

……

end

echo "analyse en avant";

analyse_avant = pipeline.reachable;

analyse_avant.print;
```

3.4.2 Présentation de l'automate atteignable pour le pipeline multi-produit ASR Résultats obtenus après la compilation du fichier PHAVer

La décidabilité de l'algorithme n'est pas assurée pour les automates hybrides linéaires, donc la terminaison n'est pas garantie, ce qui est le cas dans notre travail. Par conséquent, nous avons limité l'horloge globale à une durée finie afin d'étudier les différents comportements et scénarios de notre système.

Résultats obtenus après la compilation du fichier PHAVer

En utilisant PHAVer, nous obtenons l'espace des états atteignables pour chaque sommet, exprimé sous forme d'inégalités entre les différentes variables continues C_j et m_i. À chaque visite de ce sommet, un espace d'état différent est obtenu. L'espace des états atteignables du sommet initial est illustré dans le tableau 3.1, tandis que l'espace atteignable pour tous les autres sommets est présenté en Annexe.

L'espace d'état atteignable est visualisé sous la forme d'un automate que nous appelons l'automate atteignable ou l'automate déplié. Dans la Figure 3-5, nous présentons cet automate pour une période limitée à 100 heures (où l'horloge globale C0 <= 100 heures).

Le terme "nombre de visites d'un sommet $L(k)$" dans cet automate désigne le nombre de fois qu'un sommet spécifique est atteint. Cependant, chaque fois que ce sommet est atteint, il est caractérisé par un espace d'état d'entrée différent des précédentes visites.

Dans notre étude couvrant une période de 100 heures, le sommet L_1 est visité deux fois ($L_{1.1}$ et $L_{1.2}$), le sommet L_3 est visité trois fois ($L_{3.1}$, $L_{3.2}$et $L_{3.3}$), le sommet L_6 est également visité trois fois. Le sommet L_7 est visité une seule fois, tandis que les sommets L_{12} et L_{15} sont chacun visités deux fois.

Sommet L_1		
Conditions initiales	Visite 1 ($L_{1.1}$)	Visite 2 ($L_{1.2}$)
m16 == 4350 & m15 == 0 & m14 == 5572 & m13 == 10100 & C4 == 0 & C3 == 0 & C2 == 0 & C1 == 0 & C0 == 0	34*C0 - 200*C2 + m16 == 4350 & m15 == 0 & 75*C0 - 200*C2 - m14 == -5572 & m13 == 10100 & 2*C4 == 101 & 2*C3 == 101 & -C0 >= -100 & C2 >= 0 & 2*C1 >= 101 & C0 - C1 - C2 >= 0	34*C0 + m16 == 4350 & m15 == 0 & 75*C0 - m14 == -5572 & m13 == 10100 & C4 == 0 & C3 == 0 & C2 == 0 & C0 - C1 == 0 & C0 >= 0 & -C0 >= -100

Table 3-1 L'espace atteignable pour le sommet initiale L1

Dans le cadre de notre approche de synthèse de contrôleur, la construction de l'automate atteignable représente la première étape de notre démarche.

En analysant notre automate atteignable, nous pouvons observer que le système peut atteindre des états où les spécifications sont violées. Par exemple, dans le sommet $L_{1.1}$, $\dot{m}_{16}$=−34. Si le système reste dans ce sommet plus de 72,058 heures, la spécification m_{16} ≥1900 sera violée. Il est à noter que l'invariant de ce sommet indique qu'il est possible d'y séjourner jusqu'à ce que m_{16} atteigne zéro.

Cette observation montre que l'espace d'état continu contient des régions interdites qui doivent être éliminés. Pour y parvenir, il est impératif de calculer un contrôleur. Ce dernier aura pour rôle de limiter le comportement du système aux fonctionnements souhaités en éliminant les espaces d'état qui ne satisfont pas les spécifications imposées.

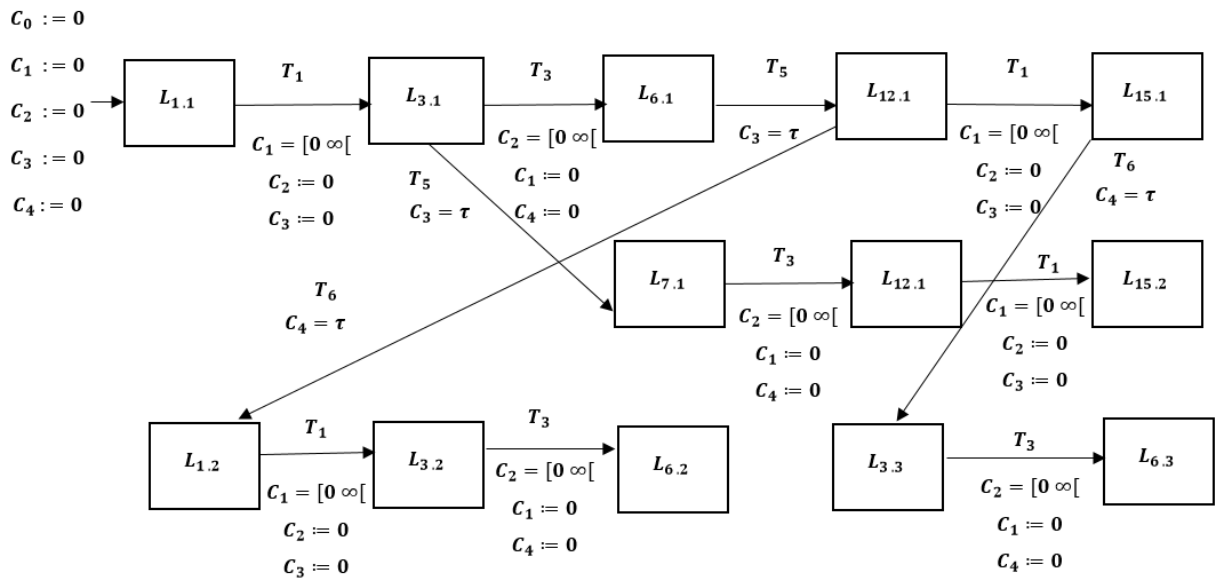


Figure 3-5 L'automate atteignable (période de 100 heures)

3.5 Conclusion

Dans ce chapitre, nous avons présenté le modèle réseau de Petri D-élémentaire. Ce modèle offre une représentation précise et complète du système de pipeline multi-produit ASR, répondant à toutes les exigences opérationnelles et illustrant clairement les interactions entre la dynamique continue et les événements discrets du système. La translation du modèle réseau de Petri hybride vers l'automate hybride a permis d'intégrer la capacité de modélisation des réseaux de Petri avec la puissance d'analyse des automates. Nous avons détaillé la méthode de conversion des réseaux de Petri hybrides D-élémentaires en automates hybrides, en appliquant cette technique à notre modèle réseau de Petri D-élémentaire qui décrit le fonctionnement du pipeline multi-produit ASR, ce qui a conduit à un automate hybride linéaire. L'utilisation de l'outil PHAVer a facilité l'analyse d'atteignabilité, permettant une exploration exhaustive des états atteignables et menant à la déduction de l'automate atteignable. Le chapitre suivant se concentrera sur le développement d'une commande basée sur ce modèle d'automate atteignable et explorera en détail la méthodologie pour concevoir un contrôleur optimal pour notre système de pipeline multi-produit.

Chapitre 4

Synthèse du Contrôleur du Système Pipeline Multi-produit ASR

Après avoir effectué la traduction du réseau de Petri hybride, nous obtenons un automate atteignable qui constitue la base pour la conception de notre contrôleur. Ce chapitre est axé sur une approche de commande inspirée des travaux de Ghomri (Ghomri, 2012). Nous explorons deux méthodes distinctes pour la synthèse du contrôleur. La première méthode se focalise sur le calcul des limites temporelles des gardes associées aux transitions contrôlables, en d'autres termes, la recherche des durées maximale et minimale de temps de séjour, au sein d'un sommet de l'automate atteignable, dans le but d'éviter les régions indésirables sans limiter le fonctionnement autorisé du système. Nous détaillons cette méthode en mettant en avant les défis et les complexités rencontrés lors de son application à notre modèle d'automate atteignable. La deuxième approche considère le temps de séjour comme une variable de décision et vise à résoudre un problème de programmation linéaire en nombres entiers mixtes. Un algorithme général est développé pour les systèmes dynamiques hybrides modélisés par un automate hybride linéaire. L'application de cette méthode à notre étude de cas a pour objectif d'assurer un contrôleur permissif tout en optant pour des trajectoires optimales. Il est important de souligner que le calcul du contrôleur se fait en temps réel (en ligne).

4.1 Principes et défis de notre approche pour la synthèse du contrôleur du pipeline multi-produit

La synthèse de contrôleur joue un rôle crucial dans l'étude des systèmes, que ce soit lors de la conception ou pour l'assurance de sûreté et le respect des spécifications imposées par le cahier des charges. Le rôle du contrôleur est d'intervenir sur la dynamique du système, le restreignant afin d'éviter d'atteindre des zones indésirables où les spécifications seraient violées. Dans notre étude, le mécanisme de contrainte n'agit que sur les variables d'état continues, représentées

par les m_i . Une particularité de notre modèle est que la partie discrète commande la partie continue. Par conséquent, ce sont les variables discrètes qui vont définir et restreindre le comportement du système. Nous nous concentrerons alors sur la date d'occurrence de l'événement contrôlable associé aux transitions discrètes pour élaborer ce contrôleur.

Dans notre approche de synthèse de contrôleur, l'idée fondamentale de notre travail s'inspire des recherches de Ghomri. Elle a proposé une méthode formelle pour la synthèse de contrôleurs adaptés aux systèmes à flux continus. Plus précisément, elle a élaboré des formules algébriques pour un automate atteignable centré sur un sommet donné, en modifiant la garde de chaque transition de sortie de ce sommet de façon qu'on le quitte avant que l'espace d'état atteigne des régions indésirables. Une des perspectives de son travail visait la mise en œuvre globale de cette approche sur un cas d'étude réel. Une autre perspective se concentrait sur la prise en compte des événements incontrôlables dans la partie discrète. Nous avons exploré ces deux perspectives distinctes en appliquant cette méthode au système pipeline multi-produit ASR. Par ailleurs, tant le réseau de Petri que l'automate correspondant, décrivant le comportement du système, comportent des événements incontrôlables.

L'approche de synthèse de contrôleur abordée dans ce travail se décline en quatre étapes principales: en premier lieu, la modélisation du système par un RdP hybride D-élémentaire sans tenir compte des spécifications (modélisation du système en boucle ouverte), ensuite, la traduction du réseau de Petri hybride en automate hybride pour procéder à une analyse formelle du système. Enfin, la modélisation des spécifications suivi du calcul du contrôleur en intervenant sur les gardes des transitions contrôlables, en introduisant de nouvelles gardes pour assurer le respect des spécifications.

Dans cette vision, nous avons modélisé le système pipeline multi-produits à l'aide des réseaux de Petri hybrides D-élémentaires. Nous avons ensuite procédé à son analyse à l'aide de l'automate hybride, résultant de la traduction du modèle RdP D-élémentaire. Afin de garantir le bon fonctionnement de notre système, nous avons abordé l'étape de calcul de contrôle. Cette étape repose sur la méthode formelle que nous avons décrite précédemment. La méthode vise à établir de nouvelles gardes pour les transitions discrètes, afin de limiter l'espace atteignable dans chaque sommet et d'éliminer les zones indésirables. Cependant, lors de l'application de cette méthode à notre système de pipeline multi-produit, nous avons rencontré des obstacles. Ces difficultés ont limité notre capacité à construire un contrôleur maximal permissif pour le système pipeline multi-produits.

Au cours de l'application des formules algébriques pour déduire des nouvelles gardes, nous nous sommes confrontés à plusieurs problématiques dues à la nature complexe du système pipeline où les états discrets (sommets) sont interdépendants. Cela qui signifie qu'une horloge peut être active dans plusieurs sommets sans être réinitialisée, en outre, la présence des évènements incontrôlables. Cette méthode stipule que si une spécification est violée avant le temps de séjour minimal d'un sommet, alors ce sommet est interdit. Dans notre contexte, cette règle pose problème, interdire une séquence dès le départ est envisageable, mais bloquer un

état spécifique au milieu d'une séquence est problématique en raison de l'interdépendance des états.

Face à l'indication qu'un sommet est interdit, notre première intuition a été d'essayer de modifier l'espace d'état à l'entrée de ce sommet en remontant les branches. La modification de cet espace d'état nécessite des ajustements au niveau du temps de séjour des sommets prédécesseurs pertinents, ce qui entraîne une modification des gardes de ces sommets. Bien entendu, ce changement dépend des spécifications violées et de la dynamique des places continues des sommets prédécesseurs. L'objectif était de garantir que le temps nécessaire pour la violation des spécifications au sein du sommet soit supérieur ou égal au temps de séjour minimal, évitant ainsi son interdiction.

Cependant, toute modification entraîne une révision des états de tous les sommets successeurs des sommets modifiés. Pour certains de ces sommets, le contrôle a déjà été déterminé. Remonter les branches pour ajuster l’espace atteignable s'avère être une démarche computationnellement intense, où chaque ajustement nécessite une nouvelle vérification des spécifications et un recalcul des gardes pour tous les sommets successeurs concernés. De plus, il arrive fréquemment que la méthode restreigne l'espace d'état des sommets prédécesseurs de manière peu formelle, rendant ainsi l'obtention d'un contrôleur maximal permissif incertaine.

Pour illustrer ces complexités, nous appliquons cette méthode sur le modèle d'automate atteignable décrivant le comportement du pipeline multi-produit ASR, ce qui permet une meilleure compréhension de la situation.

Confrontés à ces contraintes, nous avons envisagé une approche pour le calcul du contrôleur en temps réel, en enregistrant l'instant où nous quittons un sommet, c'est-à-dire l'instant d'occurrence de l'évènement discret. À partir de cet instant précis, nous abordons la résolution d'un problème de programmation linéaire. La fonction objective de ce modèle alterne entre la minimisation et la maximisation du temps de séjour au sein du sommet successeur, et les contraintes sont développés de manière à ce que l’espace atteignable respecte les spécifications. Toutefois a conditions d’éliminer au préalable tous les chemins interdits a pour ne pas se bloquer dans un état ultérieurement, cette étape sera expliquée en détail dans les sections suivantes.

Cette méthode nous offre le plus large intervalle de temps de séjour dans un sommet, sans restreindre l'espace atteignable, nous amenant ainsi à évoquer un contrôleur maximal permissif. De plus, elle nous permet de sélectionner la séquence d'injection de produit optimale. Dans notre cas, cette séquence serait celle qui présente le minimum de commutations, afin de réduire le nombre d'interfaces. Cette optimisation s'effectue en identifiant le meilleur chemin, qui répond à nos besoins, parmi les chemins accessibles de l'automate atteignable.

Les détails de cette approche seront explicités dans les sections suivantes. Nous y présenterons l'algorithme de calcul du contrôleur, puis nous l'appliquerons à notre système, le pipeline multi-produit ASR.

4.2 Modélisation des spécifications

Les spécifications imposées sur notre système ont été expliquées dans le chapitre 1 et sont résumées dans le tableau 1.1. Dans cette section, nous les formulons de manière formelle.

Dans notre modèle, les spécifications visent principalement à maintenir le niveau de produit entre une valeur maximale, pour ne pas dépasser la capacité, et une valeur minimale servant d'autonomie de stock ou de stock de sécurité. Ces spécifications concernent essentiellement la partie continue de notre système hybride, plus précisément les variables d'état continues m_i. Une spécification est représentée sous forme d'inégalité linéaire, correspondant à l'équation d'un hyperplan affine qui divise l'espace atteignable en deux régions.

Pour chaque sommet de l'automate atteignable, il est impératif que les spécifications soient respectées par l'espace d'état atteignable continu. Si cet espace d'état continu ne respecte pas les spécifications pour un sommet donné, nous nous retrouvons avec des états indésirables. Pour éviter ces situations. Nous devons quitter cet espace avant d'atteindre des régions qui violent les spécifications.

Définition 4.1 (Spécification)
Une spécification, notée $Spec$, relative au comportement continu de l'automate donné, est un prédicat linéaire défini par : $S^T \cdot M_C \leq b$, Où $S^T = (s_1, s_1, \dots, s_n)^T$ est un vecteur réel constant de dimension n, et b est une constante réelle.

Une spécification pour un sommet k est notée $Spec(k)$. Elle correspond à la conjonction de l'ensemble des spécifications imposées à l'espace d'état de ce sommet :

$Spec(k) = Spec\ 1\ \wedge Spec\ 2 \wedge \dots \wedge Spec\ i$

Note : On désigne par le vecteur S le vecteur des facteurs de spécifications. Afin de déterminer les intervalles optimaux et d'assurer ainsi un contrôleur pleinement permissif, nous avons décomposé le vecteur S pour distinguer les facteurs positifs $S+$ des facteurs négatifs $S-$, de telle manière que $S = S- + S+$.

4.2.1 Spécifications pour le pipeline multi-produit ASR

L'espace d'état atteignable de l'automate représentant le comportement du système de pipeline multi-produit inclut des régions indésirables qui ne respectent pas les spécifications imposées à ce système. À titre d'exemple, prenons le cas du sommet $L_{1.1}$ illustré dans (la figure 4-1).

L'invariant associé à ce sommet est $m_{16} \geq 0$. Compte tenu des conditions initiales mentionnées dans le tableau1.1, et sachant que toutes les horloges sont initialisées à 0, nous pouvons séjourner dans ce sommet pendant une durée maximale de $127.94\ heures$. Cependant, on observe que certaines spécifications peuvent être violées avant d'atteindre ce temps de séjour maximal. Par exemple, la spécification $m_{13} \leq 22000$ est violée à $219.4\ heures$. L'espace d'état atteignable dans ce sommet respecte donc cette spécification. En revanche, la spécification $m_{16} \geq 1900$, est violée à $72{,}058\ heures$, avant d'atteindre le temps de séjour maximal. Cela indique la présence d'un espace atteignable au sein du sommet $L_{1.1}$ qui ne respecte pas cette spécification. Il est donc nécessaire de restreindre cet espace pour qu'il soit conforme à toutes les spécifications imposées pour ce sommet.

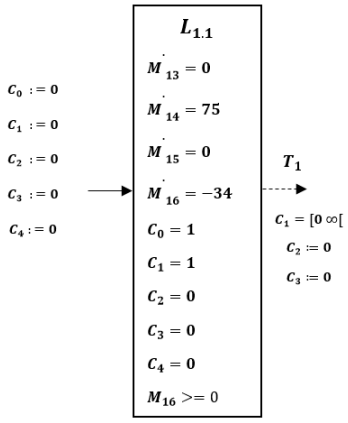


Figure 4-1 Sommet initial de l'automate atteignable

Affectation des spécifications au sommet de l'automate atteignable

Dans notre cas, la spécification est de la forme $m_i \leq b$, où b est une constante rationnelle, les spécifications relatives imposées à un sommet L_k d'un automate atteignable sont celles qui peuvent être violées pendant le temps de séjour dans ce sommet. Prenons l'exemple du sommet $L_{1.1}$, la spécification $m_{14} \geq 4400$ est vérifiée par l'espace d'état à l'entrée de ce sommet , $m_{14} = 5572 > 4400$ et le bilan de m_{14} est positif. Ainsi, l'espace de ce sommet ne peut pas atteindre une valeur inférieure à 5572, garantissant que la spécification est toujours respectée. De même, pour la spécification $m_{16} \leq 9500$, la valeur à l'entrée est m_{16} =4350 et comme le bilan de m_{16} est négatif, on ne peut pas atteindre un espace où la valeur dépasserait 9500. Par conséquent, cette spécification ne sera pas violée pendant le séjour au sommet $L_{1.1}$. Ces deux spécifications ne s'appliquent pas à ce sommet. Cependant, les autres spécifications mentionnées précédemment, $m_{16} \leq 22000$ et $m_{14} \geq 1900$, peuvent être violées et il faut vérifier si l'espace d'état atteignable en sommet $L_{1.1}$ vérifie ces spécifications.

En général, pour vérifier si une spécification de la forme $S^T \cdot M_C \leq b$ est applicable à un sommet k, on parcourt le vecteur des facteurs de spécification i, $S^T_{i,k}$. Chaque élément du vecteur est

comparé avec le vecteur des bilans des marquages noté C au sommet k. Si des éléments non nuls de $S_{i,k}^{T}$ ont le même signe que ceux du vecteur C, alors spécification k peut être une spécification pour le sommet i.

Note : Dans notre approche de synthèse, il est essentiel que l'espace d'entrée du sommet initial respecte toutes les spécifications imposées sur le système.

4.3 Application de la méthode formelle de Ghomri au pipeline Multi-produit ASR

En utilisant le théorème présenté ci-dessous, nous allons déterminer les nouvelles gardes qui garantiront le fonctionnement souhaité de notre système.

Théorème (Ghomri et al., 2005)

Le contrôle maximal permissif permettant de garantir le respect de la spécification $S^T \cdot M_C \leq b$, dans un sommet k avec une seule transition de sortie T_j , de garde : $\alpha_i \leq t_i \leq \beta_i$ est obtenu avec la nouvelle garde de T_j : $\alpha'_i \leq t_i \leq \beta'_i$

Tel que : $\alpha'_i = max(\alpha_i, t_{imin})$ et $\beta'_i = min(\beta_i, t_{imax})$)

Où : t_{imin} = t_{0min} et t_{imax} sont calculés comme suit:

$Si\ S^T \cdot C > 0\ t_u = \frac{b + S^T \cdot C.t_{0min} - S^{+T} D_{0max} - S^{-T} D_{0min}}{S^T \cdot C}$

Si $t_u < \alpha'_i$ Le sommet k est interdit ;

Si $tu \geq \alpha'_i$ $t_{imax} = t_u$

Si $S^T \cdot C < 0$ $t_u = \frac{b + S^T \cdot C.t_{0min} - S^{+T} D_{0min} - S^{-T} D_{0max}}{S^T \cdot C}$

Si $t_u > t_{0min}$ Le sommet k est interdit ;

Si $tu \leq t_{0min}$ $t_{imax} = d_{max}$

- Si $S^T \cdot C = 0$

Si $S^T D_0 \leq b$, La spécification est toujours vérifiée ;

Si $S^T D_0 > b$, Le sommet *q* est interdit ;

Remarque 4.1 : La notion de sommet interdit dans notre travail est utilisée uniquement si je ne peux pas modifier l'espace atteignable à l'entrée de ce sommet.

Application de la méthode au modèle de pipeline multi-produit ASR

$$A\ t = 0,\ D_{0min} = D_{0max} = \begin{pmatrix} 10100 \\ 5572 \\ 0 \\ 4350 \end{pmatrix},\ C0_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

Vecteur des dérivées des marquages au sommet k est donné comme suit : $C_k = \begin{pmatrix} \dot{m}_{13} \\ \dot{m}_{14} \\ \dot{m}_{15} \\ \dot{m}_{16} \end{pmatrix}$.

Le sommet $L_{1.1}$:

Les spécifications sont :

$m_{14} <= 22000$ Et $m_{16} >= 1900 \implies m_{14} <= 22000$ et $-m_{16} <= -1900$

$\alpha_i = 0\ , \beta_i = \infty$

$\alpha_i' = max(timin = 0, \alpha_i = 0) = 0$

La nouvelle garde Pour la transition T_1 est calculée comme suit :

Pour la spécification 1 $(m_{14} <= 22000)$

$S^T.C = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}^T . \begin{pmatrix} 0 \\ 75 \\ 0 \\ -34 \end{pmatrix} = 75 > 0\ \alpha_i \leq t_i \leq \beta_i$

$T_u = \frac{22000 + 75 * 0 - 1 * 5572}{75} = 219.04$

$T_u >= \alpha_i'\ donc\ T_{imax} = T_u$

$\beta_i' = min(\beta_i, timax) = \min(\infty, 219.04) = 219.04$

Pour la spécification 2 $(-m_{16} <= -1900)$

$S^T.C = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix}^T . \begin{pmatrix} 0 \\ 75 \\ 0 \\ -34 \end{pmatrix} = 34 > 0$

$T_u = \frac{-1900 + 34(0) - (-1) * 4350}{34} = 72.058$

$Specification1 \wedge Specification2$

$(0, 219.04) \wedge (0, 72.058)$, Donc la nouvelle garde de la transition T_1 est :

$(\alpha_i', \beta_i') = (0,\ 72.058)$

Donc : $0 <= C_1 <= 72{,}085$ (voir la figure 4-2)

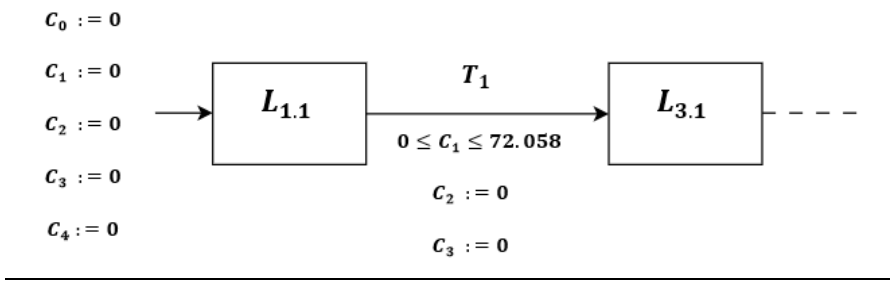


Figure 4-2 Synthèse de contrôleur : sommet L1.1

Le sommet $L_{3.1}$:

Dans ce sommet nous avons deux transitions de sortie : l'une est contrôlable 'T_3', et l'autre est incontrôlable ' T_5'

Les spécifications sont :

$m_{14} <= 22000$ Et $m_{16} >= 1900 \quad \Longrightarrow m_{14} <= 22000$ et $-m_{16} <= -1900$

En appliquant le théorème, nous obtenons les solutions suivantes :

Spécification 1 : $m_{14} <= 22000 \Longrightarrow T_u = 146.98$

Spécification 2 : $-m_{16} <= -1900 \Longrightarrow T_u = 0$

Pour la première transition T_3 :

$\alpha_i = 0\ , \beta_i = \infty$

En basant sur les valeurs T_u, la nouvelle garde de la transition T_3 est : $\alpha_i' = \beta_i' = 0$.

Pour la deuxième transition T_5 :

$T_u = 0 < d_j = 50{,}5.$

Donc la garde de la transition T_5 n'est pas vérifier (figure 4-3). Dans ce cas, selon le théorème le sommet $L_{7.1}$ est interdit. Dans la suite, nous vérifierons si nous pouvons lever cette interdiction pour autoriser ce sommet. Nous appelons cette étape : la remontée des branches.

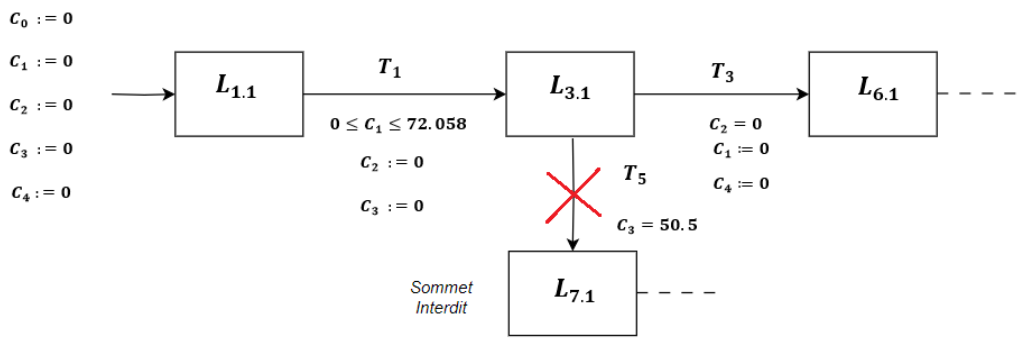


Figure 4-3 Synthèse de contrôleur : sommet interdit

La remontée des branches signifie que nous devons modifier de nouveau les gardes des transitions des sommets prédécesseurs du sommet interdit, dans le but de changer l'espace d'état continu à l'entrée de ce sommet. Ceci s'effectue en cherchant un marquage des m_i à l'entrée de ce sommet qui peut satisfaire nos spécifications durant le temps de séjour minimal dans ledit sommet.

Remarque 4.2: Les nouvelles valeurs des marquages, nécessaires pour satisfaire les spécifications, sont choisies en fonction de l'espace dynamique continu et de l'espace des horloges à l'entrée du sommet

Exemple

$T_u = 0$, et la garde de la transition incontrôlable T_5 est $d_j = 50\,,5$

Je remonte les branches, ce qui signifie que je dois modifier à nouveau la garde de la transition T_1 reliant ($L_{1.1}$ à $L_{3.1}$).

Le temps de séjour minimal requis au sein du sommet $L_{3.1}$ pour satisfaire la garde de la transition T_5 est donné comme suit :

$d_j - T_u$, si la transition est contrôlable,

$\alpha_i - T_u$ si la transition est incontrôlable

Dans ce cas, la spécification qui a été violé durant Le temps de séjour dans le sommet $L_{3.1}$ est $m_{16} >= 1900$. Nous cherchons à augmenter la valeur D_{0min} (m_{16}) à l'entrée de ce sommet de plus d'une valeur égale à $(d_j - T_u + C0_{x*}).\dot{m}_{16}(L_{3.1})$. Ou $\dot{m}_{16}$ est le bilan de

m_{16} dans le sommet $L_{3.1}$, et $C0_{x*}$ est la valeur de l'horloge active dans la transition T_5 a l'entrée du sommet $L_{3.1}$.

$$(d_j - T_u + C0_3).\dot{m}_{16}(L_{3.1}). = (50.5 - 0 + 0) \;\; 34 \;\; = 1717$$

La garde de la transition T_1 sera modifiée de la manière suivante :

$$\beta_i{}'\,(Nouvelle\ valeur) = \beta_i{}'\,(l'ancien\ valeur) - \frac{(d_j - T_u + C0_3).\dot{m}_{16}(L_{3.1}}{\dot{m}_{16}(L_{1.1})}$$

$$\beta_i{}'\,(Nouvelle\ valeur) = 72.058 - 50.5 = 21{,}558$$

$$\alpha_i{}'\,(nouvelle\ valeur) = \alpha_i{}'\,(l'ancien\ valeur) = 0$$

Donc la nouvelle garde de la transition T_1 est :

$$(\alpha_i{}', \beta_i{}') = (0, \;\; 21.558)$$

Étant donné que l'espace d'état continu à l'entrée du sommet $L_{3.1}$ a été modifié suite à la mise à jour de la garde précédente, nous devons recalculer la valeur T_u des spécifications et déduire la nouvelle garde pour la transition T_3.

Donc la nouvelle garde de la transition T_3 est : $(\alpha_i{}', \beta_i{}') = (0, \;\; 50{,}5)$ (figure 4-4).

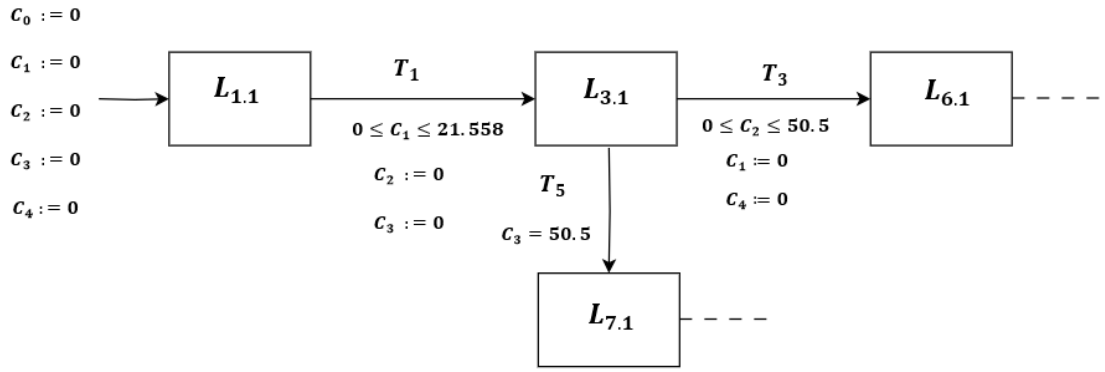


Figure 4-4 Synthèse de contrôleur : Remontée des branches

4.4 Synthèse de contrôleur en temps réel pour les systèmes hybrides linéaires

Dans cette section, nous présenterons une approche de synthèse de contrôleur en temps réel pour un système hybride dynamique linéaire.

L'approche de synthèse de contrôleur proposée est appliquée à un automate atteignable en utilisant une méthode basée sur la modification des gardes des transitions contrôlables pour respecter un ensemble de spécifications imposés par le système.
Le contrôle est divisé en deux principales parties. Tout d'abord, nous développons un algorithme pour formuler l'espace d'état désiré dans chaque sommet de l'automate atteignable sous forme de contraintes où nous considérons le temps de séjour comme une variable de décision à optimiser. Nous calculons le contrôleur en temps réel, nous résolvons un problème d'optimisation (programmation linéaire).
La fonction objective alterne entre la minimisation et la maximisation du temps de séjour à chaque résolution. D'abord, nous résolvons le programme linéaire avec un objectif de minimisation du temps de séjour, puis nous le résolvons à nouveau avec un objectif de maximisation du temps de séjour. Ceci nous permet d'obtenir l'intervalle le plus large possible pour le temps de séjour et, par conséquent, d'élargir l'espace atteignable dans ce sommet.
Enfin, cette approche est appliquée à un cas réel sur le pipeline multi-produit ASR.

Nomenclature

k	Indice du sommet.
i	Indice de spécification.
M	Nombre des sommets dans le chemin P_z.
N_k	Nombre de spécifications pour le sommet k.
$S_{i,k}^T$	Transposée du vecteur de spécification i dans le sommet k.
$q_{i,k}$	Un nombre rationnel relatif à la spécification i dans le sommet k.
$D0_{k,j}$	Élément j du vecteur $D0_k$.
$C_{k,x}$	Valeur de l'horloge x dans le sommet k.
B	Liste des contraintes pour le chemin P_z.
$a_{k,k+1}, b_{k,k+1}$	Date de franchissement au plus tôt et au plus tard pour la transition contrôlable $T_{k,k+1}$
$d_{k,k+1}$	Date de franchissement pour la transition incontrôlable $T_{k,k+1}$.
h_k	le temps de séjour exacte au sommet k (variable de décision).
D_k	Vecteur des dérivées des marquages au sommet k.
$D_{k,j}$	Élément j du vecteur D_k.
$D0_k$	Vecteur des marquages à l'entrée du sommet k.
$D0_{k,j}$	Élément j du vecteur $D0_k$.
$C_{k,x}$	Valeur de la dérivée de l'horloge x au sommet k.
$C0_{k,x}$	Valeur de l'horloge x à l'entrée du sommet k.
$C0_{k,x*}$	Valeur de l'horloge active pour la transition $T_{k,k+1}$ à l'entrée du sommet k.
$A_{k,x}$	Affectation de l'horloge x à l'entrée du sommet k, $A_{k,x} := 0$, si l'horloge est initialisée à l'entrée du sommet k, sinon $A_{k,x} := 1$.
$A_{k,x*}$	Affectation de l'horloge active pour la transition $T_{k,k+1}$ à l'entrée du sommet k.
$TS_{i,k}$	le temps de séjour maximal au sommet k avant de violer la spécification i.

$inv_{k,x}$ Invariant sur l'horloge x au sommet k.

4.4.1 Présentation de l'approche de synthèse de contrôleur en temps réel pour un système hybride linéaire

Cette approche peut être résumée en cinq étapes :
Étape 1. Déduire l'automate atteignable (nous limitons l'horloge globale à une période prédéfinie).
Étape 2. Formuler les spécifications dans chaque sommet de l'automate atteignable sous la forme $TS_{i,k}$. $M_k \leq q_{i,k}$, sachant que $q_{i,k}$, est un nombre rationnel.

Étape 3.
Pour chaque sommet L_k de l'automate atteignable, formuler les contraintes sur l'espace d'état atteignable basée sur l'équation suivante: $TS_{i,k} = \frac{q_{i,k} - S_{i,k}^T . D0_k}{S_{i,k}^T . D_k}$
Nous avons développé un algorithme pour construire des contraintes sur l'espace d'état en fonction du temps de séjour en supposant que, dans chaque sommet L_k, une horloge h_k mesure le temps de séjour exact et représente la variable de décision dans notre système. Ces contraintes définissent l'espace d'état souhaitable du système. Cet algorithme est détaillé dans la section suivante, et il est basé sur l'équation suivante qui calcule le temps de séjour maximum dans le sommet L_k avant de violer la spécification i.

Étape 4. Trouver les chemins interdits.
Pour trouver les chemins interdits dans l'automate atteignable, nous faisons ce qui suit :
Pour chaque chemin, nous résolvons un programme linéaire, dont la fonction objectif est de minimiser ou de maximiser le temps de séjour dans le sommet initial sous les contraintes formulées pour ce chemin. Si nous trouvons une solution, le chemin est accessible. Si aucune solution n'est trouvée, le chemin est interdit.
Étape 5. Contrôle en temps réel
Il y a deux méthodes pour synthétiser le contrôleur en temps réel :
Méthode 1 : Choisissez le chemin le plus optimal et résolvez uniquement les contraintes de ce chemin.
Méthode 2 : Ne choisissez pas un chemin précis. Par conséquent, résolvez l'ensemble des contraintes de tous les chemins accessibles (nous retirons les sommets des chemins interdits). Et chaque fois que nous arrivons à un sommet qui a plus d'un successeur, nous en choisissons un, et nous éliminons les contraintes des chemins non choisis et ainsi de suite.

4.4.2 Algorithme de formulation des contraintes (étape 3)

En supposant que nous disposions du chemin illustré dans la Figure 4-5 d'un automate d'atteignabilité, l'algorithme est présenté ci-après pour le sommet initial et pour un sommet successeur.

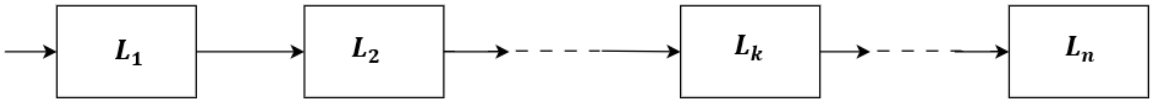


Figure 4-5 Chemin dans un automate atteignable

Sommet initial (L_1):

$k = 1$

$i = 1$

Pour chaque spécification i = 1 ... N_1 ($nombre\ de\ spécifications\ dan\ L_1$)

Calculer $S_{i,k}^T . D_k$

Si $S_{i,k}^T . D_k \neq 0$

Calculer $TS_{i,k}$

$$TS_{i,k} = \frac{q_{i,k} - S_{i,k}^T . D0_k}{S_{i,k}^T . D_k}$$

Si la transition est contrôlable, vérifier :

Si $TS_{i,k} + C0_{k,x*} . A_{k,x*} \geq a_k$

Ajouter la contrainte suivante à la liste B

h_k<=$TS_{i,k}$

$i = i \quad + 1$

Sinon, Si $TS_{i,k} + C0_{k,x*} . A_{k,x*} < a_k$

Pas de solution, aller à Fin

Fin si

Sinon si la transition est incontrôlable, vérifier :

Si $TS_{i,k} + C0_{k,x*} . A_{k,x*} \geq d_k$

Ajouter les contraintes suivantes à la liste B :

h_k<=$TS_{i,k}$

$i = i \quad + 1$

Sinon, Si $TS_{i,k} + C0_{k,x*} . A_{k,x*} < d_k$

Pas de solution, aller à Fin

Fin si

Fin si

I=i+1

Sinon si $S_{i,k}^{T}.D_k = 0$

Si $S_{i,k}^{T}.D0_k <= q_{i,k}$

La spécification i est toujours satisfaite

$i = i + 1$

Sinon, Si $S_{i,k}^{T}.D0_k >= q_{i,k}$

Pas de solution, aller à Fin

Fin si

Fin si

Fin pour

Si la transition est contrôlable, Ajouter les contraintes suivantes à la liste B :

$h_k + C0_{K,x*}.A_{k,x*} \geq a_k$

$h_k + C0_{K,x*}.A_{k,x*} \leq b_k$

Pour chaque invariant dans le sommet k

$h_k \leq inv(k,x) - C0_{k,x}$

Sinon si la transition est incontrôlable, Ajouter les contraintes suivantes à la liste B :

$h_k + C0_{k,x*}.A_{k,x*} = d_k$

Pour chaque invariant dans le sommet k

$h_k \leq inv(k,x) - C0_{k,x}$

Fin pour

Fin si

$k = k + 1$

Sommet L_k :

Calculer$D0_k$, $C0_k$

1 / $D0_{k,j}$ est calculé comme suit :

$D0_{k,j} = D0_{1,j} + h_1.D_{1,j}$+……+$h_{k-1}.D_{k-1,j}$

$$D0_{k,j} = D0_{1,j} + \sum_{i=1}^{k-1} h_i.D_{i,j}$$

2/ Chaque élément du vecteur $C0_k$ qui est $C0_{k,x}$ est calculé comme suit :

$C0_{k,x} = C0_{1,x}.\prod_{i=1}^{k} A_{i,x} + h_1\,.C_{1,x}.\prod_{i=2}^{k} A_{i,x} + \cdots h_{k-1}\,.C_{k-1,x}.A_{k,x}$

$$C0_{k,x} = C0_{1,x}.\prod_{i=1}^{k} A_{i,x} + \sum_{j=1}^{k-1} h_j.C_{j,x}.\prod_{i=j+1}^{k} A_{i,x}$$

Pour chaque sommet $k = 2$ …… n dans le chemin 1

$i = 1$

Pour chaque spécification $\mathrm{i} = 1..$ nombre de spécifications dans $\mathrm{L_k}$:

Calculer $S_{i,k}^T.D_k$

Si $S_{i,k}^T.D_k \neq 0$

Calculer $TS_{i,k}$

$$TS_{i,k} = \frac{q_{i,k} - S_{i,k}^T.D0_k}{S_{i,k}^T.D_k}$$

Ajouter les contraintes suivantes à la liste B :

h_k<=$TS_{i,k}$

$i = i + 1$

Sinon si, $S_{i,k}^T.D_k = 0$

Ajouter les contraintes suivantes à la liste B :

$S_{i,k}^T.D0_k$<= $q_{i,k}$

Fin si

$i = i + 1$

$Fin\ pour\ i$

Si la transition est contrôlable, ajouter les contraintes suivantes à la liste de contraintes

$h_k + C0_{k,x*} \geq a_k$

$h_k + C0_{k,x*} \leq b_k$

Pour chaque invariant dans le sommet k, faire :

$h_k \leq inv(k,x) - C0_{K,x}$

$Fin\ pour$

SINON, si la transition est incontrôlable, ajouter les contraintes suivantes à la liste de contraintes

$h_K + C0_{K,x*} = d_k$

Pour chaque invariant dans le sommet k, faire :

$h_k \leq inv(k,x) - C0_{K,x}$

$Fin\ pour$

Fin si

$k = k + 1$

Fin pour k

Fin

4.5 Application de l'approche de synthèse du contrôleur au pipeline ASR multi-produit

Étape 1 : Déduire l'automate atteignable sur une période prédéfinie (Figure 3-5).

Étape 2 : Formuler les spécifications sous la forme $S_{i,k}^T . M_k \leq q_{i,k}$:

Stock de sécurité du produit pr_k <=Stock du produit pr_k <= capacité du produit pr_k $\forall\ k = 1..2$

Exemple:

Les spécifications du sommet initial $L_{1,1}$ sont: $M_{14} \leq 22000, -M_{16} \leq -1900$

Les spécifications du sommet L_k $(L_{12,1})$ sont: $-M_{14} \leq -4400,\ M_{16} \leq 9500,$

Étape 3 : Construire les contraintes (Application de l'algorithme proposé)

Exemple

En appliquant l'algorithme au chemin 1, nous détaillons les phases de l'algorithme dans le sommet initial $L_{1.1}$ et le sommet successeur $L_{12.1}$, l'ensemble des contraintes du chemin sont illustrées dans le Tableau 4.1.

Sommet initial $L_{1.1}$: Les conditions initiales sont : (voir Tableau 1.1)

$M_{13} = 22000,\ M_{14} = 5572,\ M_{15} = 0,\ M_{16} = 4350$

Calculer $S_{i,k}^T . D_k$

$$D0_1 = \begin{pmatrix} 10100 \\ 5572 \\ 0 \\ 4350 \end{pmatrix},\ C0_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},\ A_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Pour chaque spécification $i = 1..2\ faire$

$Spécification\ ((1.1)\ , 1) : M_{14} \leq 22000$

$$S_{i,k}^T . D_k = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}^T . \begin{pmatrix} 0 \\ 75 \\ 0 \\ -34 \end{pmatrix} = 75 \neq 0$$

Calculer

$$TS_{i,k} = \frac{q_{i,k} - S_{i,k}^T . D0_k}{S_{i,k}^T . D_k} = \frac{22000 - \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}^T . \begin{pmatrix} 10100 \\ 5572 \\ 0 \\ 4350 \end{pmatrix}}{75} = 219.04$$

$TS_{i,k} + C0_{k,x} . A_{k,x*} = 219.04 \geq a_k = 0\ \ (x* \ \rightarrow \ C_1)$

Ajouter la contrainte suivante à la liste de contraintes:

h_k<=$TS_{i,k}$ So $h_{1.1}$<=219.04

$i = i + 1$

$Spécification\ 2 : -M_{16} \leq -1900$

h_k<=$TS_{i,k}$ Donc $h_{1.1} \leq 72.058$

La transition est contrôlable, ajoutez les contraintes suivantes à la liste des contraintes :

$h_k + C0_{k,x*} . A_{k,x*} \geq a_k\ Donc$ $h_{1.1}$>=0

$h_k + C0_{k,x*} . A_{k,x*} \leq b_k$ ($Toujours\ vérifié\ car\ \ b \ \rightarrow \ \infty$)

$K = k + 1$

Sommet k ($L_{12.1}$)

Calculer $D0_k$, $C0_k$

$D0_{k,x} = D0_{1,x} + h_1 . D_{1,x}$+……+$h_{k-1} . D_{k-1,x}$

$D0_{k,1} = D0_{1,1} + \sum_{i=1}^{k-1} h_i . D_{i,1} = 10100 + h_{1.1} . 0 + h_{3.1} . (-200) + h_{6.1} . 0 = 10100 - 200 . h_{3.1}$

$$D0_{12.1} = \begin{pmatrix} 10100 - 200.h_{3.1} \\ 5572 + 75(h_{1.1} + h_{3.1} + h_{6.1}) \\ 200.h_{3.1} \\ 4350 - 34(h_{1.1} + h_{3.1} + h_{6.1}) \end{pmatrix}$$

$$C0_{k,x} = C0_{k=1,x}.\prod_{i=1}^{k} A_{i,x} + h_{k=1}\,.C_{k=1,x}.\prod_{i=2}^{k} A_{i,x} + \cdots + h_{k-1}\,.C_{k-1,x}.\,A_{k,x}$$

$$C0_{k,x} = C0_{k=1,x}.\prod_{i=1}^{k} A_{i,x} + \sum_{j=1}^{k-1} h_j.C_{j,x}.\prod_{i=j+1}^{k} A_{i,x}$$

$$C0_{(k=12.1),1} = 0.(1.1.0.1) + h_{1.1}.(1)(1.0.1) + h_{1.1}.(0)(0.1) + h_{6.1}.(1)(1) = h_{6.1}$$

$$C0_{12.1} = \begin{pmatrix} h_{6,1} \\ h_{3,1} \\ h_{3,1} + h_{6,1} \\ h_{6,1} \end{pmatrix}$$

$Pour\ chaque\ spécifiaction\ i = 1..2\ faire$

$Spécification\ ((1.1)\ ,1) : -M_{14} \leq -4400$

Calculer: $S_{i,k}^{T}.D_k = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \end{pmatrix}^{T} . \begin{pmatrix} 200 \\ -125 \\ -200 \\ 166 \end{pmatrix} = 125 \neq 0$

Calculer:

$$TS_{i,k} = \frac{q_{i,k} - S_{i,k}^{T}.D0_k}{S_{i,k}^{T}.D_K} = \frac{-4400 - \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \end{pmatrix}^{T} . \begin{pmatrix} 10100 - 200.h_{3.1} \\ 5572 + 75(h_{1.1} + h_{3.1} + h_{6.1}) \\ 200.h_{3.1} \\ 4350 - 34(h_{1.1} + h_{3.1} + h_{6.1}) \end{pmatrix}}{125}$$

$$= 9.376 + \frac{75}{125}(h_{1.1} + h_{3.1} + h_{6.1})$$

Ajouter les contraintes suivantes à la liste de contraintes

h_k<=$TS_{i,k}$

$h_{12.1}$<=$9.376 + \frac{75}{125}(h_{1.1} + h_{3.1} + h_{6.1})$

$i = i + 1$

$Specification\ 2$: $M_{16} \leq 9500$

Ajouter les contraintes suivantes à la liste de contraintes

$h_{12.1}$<=$31.024 + \frac{34}{166}(h_{1.1} + h_{3.1} + h_{6.1})$

Transition incontrôlable, nous ajoutons les contraintes suivantes :

$h_k + C0_{K,x*} = d_k$

$h_{12,1} + h_{12,1} = 50.5$

Donc, pour le chemin 1, les contraintes sont illustrées dans le tableau suivant (tab 4.1) :

$\boldsymbol{Sommet\ L_{1.1}}$	$\boldsymbol{Sommet\ L_{3.1}}$	$\boldsymbol{Sommet\ L_{6.1}}$
$h_{1.1} <= 72.0588$ $h_{1.1} \geq 0$ $h_{1.1} <= 219.04$	$h_{3.1} <= 72.0588 - h_{1.1}$ $h_{3.1} <= 219.04 - h_{1.1}$ $h_{3.1} <= 50.5$ $h_{3.1} >= 5$	$h_{6.1} <= 72.05886 - h_{1.1} - h_{3.1}$ $h_{6.1} <= 219.04 - h_{1.1} - h_{3.1}$ $h_{3.1} + h_{6.1} = 50.5$ $h_{6.1} >= 0$
$\boldsymbol{Sommet\ L_{12.1}}$	$\boldsymbol{Sommet\ L_{1.2}}$	$\boldsymbol{Sommet\ L_{3.2}}$
$h_{12.1} <= 31.024 + (34/166)(h_{1.1} + h_{3.1} + h_{6.1})$ $h_{12.1} <= 9.376 + (75/125)(h_{1.1} + h_{3.1} + h_{6.1})$ $h_{6.1} + h_{12.1} = 50.5$ $h_{12.1} >= 0$	$h_{1.2} <= 219.04 - h_{1.1} - h_{3.1} - h_{6.1} + (125/75)\, h_{12.1}$ $h_{1.2} <= 72.0588 - h_{1.1} - h_{3.1} - h_{6.1} + (166/34)\, h_{12.1}$ $h_{1.2} >= 0$	$h_{3.2} <= 219.04 - h_{1.1} - h_{3.1} - h_{6.1} - h_{1.2} + (125/75)\, h_{12.1}$ $h_{3.2} <= 72.0588 - h_{1.1} - h_{3.1} - h_{6.1} - h_{1.2} + (166/34)\, h_{12.1}$ $h_{3.2} >= 5$
$\boldsymbol{Sommet\ L_{6.2}}$		
$h_{6.2} <= 219.04 - h_{1.1} - h_{3.1} - h_{6.1} - h_{1.2} - h_{3.2} + (125/75)\, h_{12.1}$ $h_{6.2} <= 72.0588 - h_{1.1} - h_{3.1} - h_{6.1} - h_{1.2} - h_{3.2} + (166/34)\, h_{12.1}$ $h_{3.2} + h_{6.2} = 50.5\,,\ h_{6.2} >= 0$		

Table 4-1 Liste des contraintes du chemin 1

Tableau 4.1 : Liste des contraintes du chemin 1

B/ Trouver les chemins interdits

Dans notre automate atteignable pour une période de 100 heures, nous avons trois chemins (voir figure 3-5) .

Pour chaque chemin j, résoudre le modèle de programmation linéaire suivant :

Fonction objective : max $h_{1.1}$ ou min $h_{1.1}$

Contraintes : ensemble des contraintes du chemin *j*

Chemin 1: $L_{1,1},\ L_{3,1},\ L_{6,1},\ L_{12,1},\ L_{1,2},\ L_{3,2},\ L_{6,2}$

Solution trouvée : chemin accessible

Chemin 2: $L_{1,1},\ L_{3,1},\ L_{6,1},\ L_{12,1},\ L_{15,1},\ L_{3,3},\ L_{6,3}$

Solution trouvée : chemin accessible

Chemin 3: $L_{1,1},\ L_{3,1},\ L_{7,1},\ L_{12,2},\ L_{15,2}$

Pas de solution trouvée : chemin interdit

Étape 4 : Déduire le contrôleur en temps réel :

Dans notre travail, nous cherchons à minimiser l'interface entre deux batches successifs en choisissant le planning optimale. Nous devons donc choisir le chemin optimal parmi les chemins accessibles (chemin 1 et chemin 2).

Dans les deux chemins, nous avons le même nombre d'interfaces (le même nombre de commutations). Je vais donc choisir l'un d'entre eux, par exemple le chemin 1. La fonction objective est de maximiser et minimiser le temps de séjour dans chaque sommet.

Sommet initial:

$$\begin{cases} fonction\ objectif\ :\ min\ (h_{1.1}) \\ ontraintes:\ ensemble\ des\ contraintes\ du\ chemin\ 1\ (table\ 4.1) \end{cases}$$

$$\begin{cases} fonction\ objectif\ :\ max\ (h_{1.1}) \\ contraintes:\ ensemble\ des\ contraintes\ du\ chemin\ 1\ (table\ 4.1) \end{cases}$$

Solution:

$0 \leq h_{1.1} \leq 21.558$

$C_{k,x*} = h_k + C0_{k,x*}$

$C_{1.1,1} = h_{1.1} + C0_{1.1,1}$

$0 \leq C_{1.1,1} \leq 21.558$

Sommet successeur :

Événement 1:

Si, par exemple, l'opérateur décide d'arrêter l'injection du produit 1 pour injecter le produit 2 à $h_{1.1} = 20$,$C_{1.1,1} = 20$.

Nous remplaçons $h_{1.1}$ par sa valeur dans notre modèle mathématique, soit $h_{1.1} = 20$ et nous résolvons de la même manière que pour sommet précédent un problème MILP. Ainsi nous obtenons $h_{3.1}$.

$min\ (h_{3.1}) = 10{,}024$, $max\ (h_{3.1}) = 45{,}46$

$10{,}024 \leq h_{3.1} \leq 45{,}46$

$10{,}024 \leq C_{3.1,2} \leq 45{,}46$

Événement 2:

Par exemple, l'opérateur décide d'arrêter l'injection du produit 2 pour injecter le produit 1 à

$h_{3.1} = 40$ et $C_{3.1,2} = 40$.

Nous remplaçons $h_{3.1}$ par sa valeur dans notre modèle mathématique, $h_{3.1} = 40$ et nous trouvons $h_{6.1}$.

$min\ (h_{6.1}) = max\ (h_{6.1}) = 10.5$ (Nous avons obtenu les mêmes valeurs pour max et min car la transition de sortie de ce sommet est incontrôlable).

$C_{6.1,3} = h_{6.1} + C0_{6.1,3}$, $C0_{6.1,3} = h_{3.1}$ (Nous utilisons l'équation proposée dans notre algorithme pour calculer la valeur de l'horloge à l'entrée d'un sommet).

$C_{6.1,3} = 40 + 10{,}5 = 50{,}5$ (C'est vérifié),

Le calcul du contrôleur en temps réel se termine de la même manière, jusqu'au sommet $L_{1.2}$

Remarque 4.3 : Nous ne pouvons pas poursuivre le calcul jusqu'au sommet $L_{6.2}$, nous devons construire de nouvelles contraintes à partir du sommet $L_{1.2}$. Il faut toujours faire le calcul à l'avance de 50,5 heures ou plus (le temps nécessaire entre l'injection et la réception d'un produit) car les sommets dépendent les uns des autres. Chaque batch injecté, son volume et le moment de son injection influencent les opérations suivantes.

4.6 Comparaison des approches MILP et de synthèse de contrôleur en temps réel pour la planification du pipeline multi-produit ASR

L'approche MILP a déjà été présentée dans le chapitre1. Nous introduisons maintenant les solutions basées sur la synthèse de contrôleur en temps réel pour ensuite les comparer à l'approche MILP. Le planning, pour les deux approches, sera présenté sur un horizon de temps similaire. La différence principale réside dans la définition des demandes journalières, ce qui peut conduire à des résultats variés selon le scénario.

L'objectif de notre étude est de minimiser le nombre d'interfaces tout en satisfaisant les demandes des clients et en respectant les exigences opérationnelles, comme la capacité de stockage et le niveau de stock de sécurité. Ces exigences sont gérées par notre contrôleur, qui établit un intervalle de séjour minimal et maximal pour chaque état du pipeline représenté

dans l'automate atteignable, permettant de respecter nos spécifications. L'optimisation de l'interface est réalisée en choisissant un chemin avec un nombre minimal de commutations. En outre, nous pouvons optimiser davantage nos résultats en jouant sur le temps de séjour, c'est-à-dire en maximisant la taille des batches pour obtenir le nombre minimum de batches sur la plus longue période possible.

4.6.1 Contrôle optimal en temps réel du pipeline multi-produit ASR

Comme démontré dans la section précédente, qui présentait les étapes de calcul du contrôleur en temps réel, nous poursuivons de manière similaire pour déterminer l'intervalle maximal permis de séjour pour chaque état du chemin choisi. Toutefois, au lieu de choisir un intervalle arbitraire, nous optons pour le temps de séjour maximal. Ainsi, si nous avons un intervalle de séjour $[a_j, b_j]$, avec a_j est le temps minimal et b_j comme temps maximal, nous sélectionnons b_j pour quitter l'état. Cette stratégie nous permet de maximiser le temps passé dans chaque état tout en respectant les spécifications requises. Cela se traduit par une durée d'injection considérable pour chaque batch, réduisant ainsi le nombre de batches nécessaires dans un horizon de planification déterminé.

Dans l'approche MILP, l'horizon de planification est fixé à 384 heures. Nous continuons donc le calcul du contrôleur en temps réel jusqu'à ce que cet horizon soit atteint ou dépassé.

Résultats du Calcul du Contrôleur en Temps Réel

A- Sélection du Chemin :

L'opération commence avec le choix du chemin qui représente le planning optimal, dans notre cas nous optons pour le chemin 1 de l'automate atteignable (figure 3-5) : $L_{1,1}$, $L_{3,1}$, $L_{6,1}$, $L_{12,1}$, $L_{1,2}$, $L_{3,2}$, $L_{6,2}$

B- Optimisation du Temps de Séjour

Pour chaque état, nous cherchons d'abord à identifier l'intervalle de temps de séjour le plus large possible, tout en respectant les spécifications requises. Une fois cet intervalle déterminé, dans le cadre de l'optimisation du planning, nous sélectionnons la durée maximale de séjour pour quitter l'état.

Pour $L_{1,1}$: L'intervalle est : $0 <= h_{1,1} <= 21.558$. Nous choisissons $h_{1,1} = 21.558\ h$ comme durée de séjour

En appliquant la même méthode aux états suivants, nous obtenons :

$h_{1,1} = 21.558$

$h_{3,1} = 45.78$

$h_{6,1} = 4.72 \quad et \quad h_{12,1} = 45{,}78.$

$h_{1,2} = 172.78$

Résultats du Calcul du Contrôleur en Temps Réel (suite)

Nous avons étendu notre automate atteignable pour permettre une exploration plus longue, dans le but d'atteindre de nouveaux états et de parvenir à l'horizon de planification fixé. Ensuite, en utilisant notre algorithme et les résultats précédents du calcul du contrôleur, nous sélectionnons un nouveau chemin optimal non interdit.

A- Sélection du Chemin

Le chemin choisit est le suivant : $L_{3,2},\ L_{6,2},\ L_{12,3},\ L_{1,3},\ L_{3,3}$

Les contraintes spécifiques développées pour ce chemin sont présentées dans le tableau suivant :

$\boldsymbol{Sommet\ L_{12.3}}$	$\boldsymbol{Sommet\ L_{1.3}}$	$\boldsymbol{Sommet\ L_{3.3}}$
$h_{12.3} <= 0.0031 + (34/166) * (h_{1.2} + h_{3.2} + h_{6.2})$ $h_{12.3} <= 6.68 + (75/125) * (h_{1.2} + h_{3.2} + h_{6.2})$ $h_{6.2} + h_{12.3} == 50.5$ $h_{12.3} >= 0$	$h_{1.3} <= 223.50$ $h_{1.3} >= 0$ $h_{1.3} <= 76.45$	$h_{3.3} <= 223.50 - h_{1.3}$ $h_{3.3} <= 76.45 - h_{1.3}$ $h_{3.3} <= 50.5$ $h_{3.3} >= 5$

Table 4-2 Liste des contraintes du chemin 1 (suite)

B- Optimisation du Temps de Séjour

$h_{3,2} = 45.73\ heures$

$h_{6,2}, = 4.77 \quad et \quad h_{12,3} = 45{,}73.$

$h_{1,3} = 25.95$

$h_{3,3} = 15.66$

Notez que notre planification s'arrête à $L_{1,3}$, car nous avons déjà dépassé l'horizon de planification prédéfini.

Établissement du Planning

Je vais brièvement expliquer comment le planning est construit en montrant les différents états du contrôleur et comment ceux-ci sont intégrés dans notre planning.

État $L_{1,3}$: Continuation de l'Injection du l'ancien Batch :

Contrairement à l'approche MILP, où l'injection d'un nouveau batch commence à l'instant initial, ici nous avons la possibilité de continuer l'injection du batch existant dans le pipeline. L'état $L_{1,1}$ permet la poursuite de l'injection du produit $P1$ jusqu'à $21.558\ heures$. Avec un débit constant de $200\ m^3/h$, cela se traduit par l'injection de $9156\ m^3$ du produit $P1$.

État $L_{3,1}$: Injection du du Nouveau Batch 1 (Produit 2)

Cet état représente l'injection du produit 2. Nous pouvons maintenir cette étape pour une durée de $45.78\ heures$, commençant à $21.558\ heures$ et se poursuivant jusqu'à $67.338\ heures$.

États $L_{6,1}$, $L_{12,1}$, $L_{1,2}$: Injection du Nouveau Batch 2 (Produit 1)

Ces états représentent l'injection du nouveau batch 2. La représentation détaillée de ce batch est illustrée dans la figure 4-6.

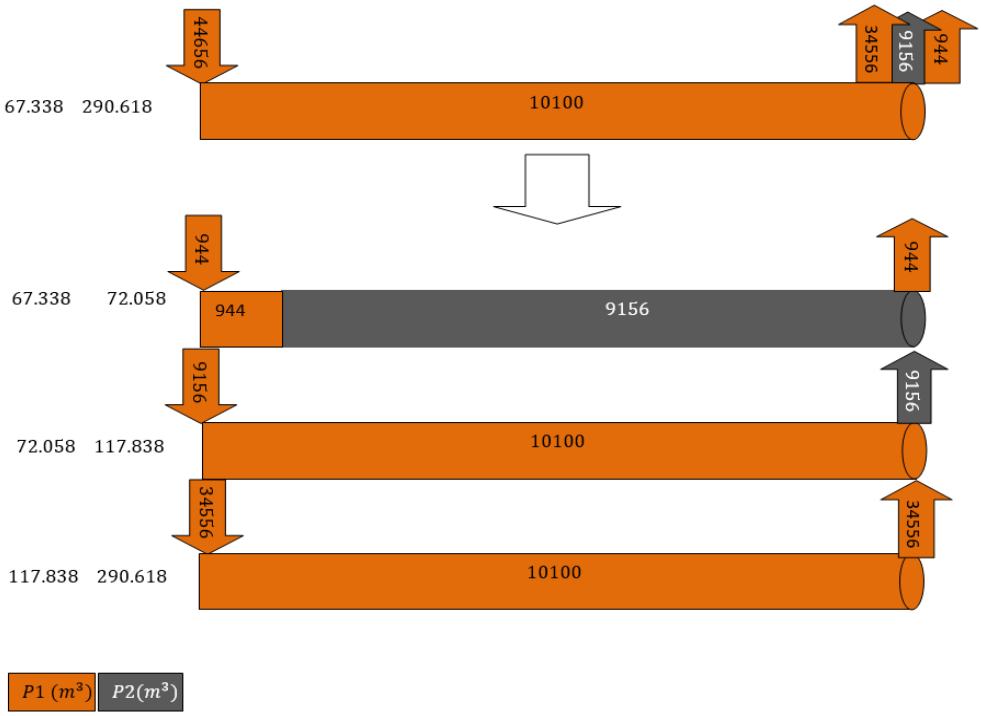


Figure 4-6 Injection du batch 02 (Approche de Contrôle Optimal)

Le planning optimal est représenté dans la figure 4-7. Comme nous pouvons le constater, il est assez similaire au planning MILP présenté dans la figure 4-7, où quatre batches sont injectés dans chaque approche.

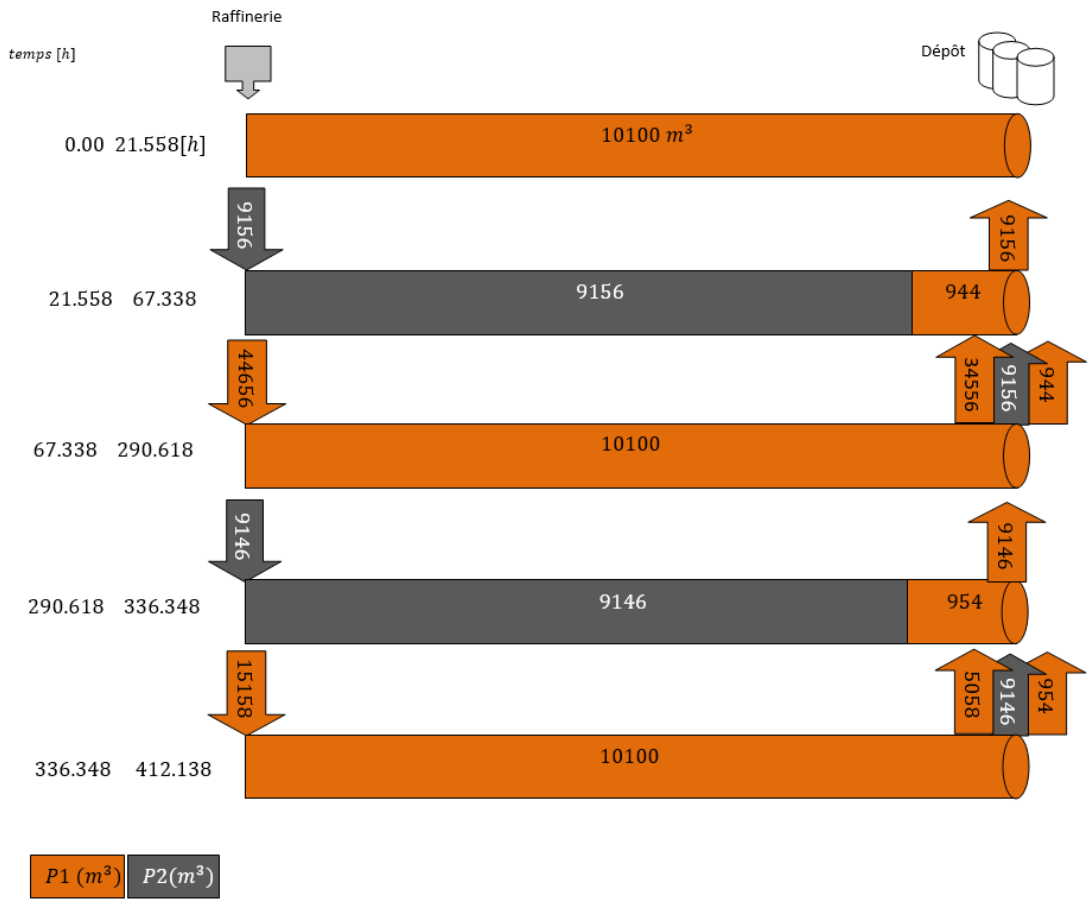


Figure 4-7 Planning optimal du transport de carburants via le pipeline multi-produit ASR (Approche de contrôle optimal)

4.6.2 Analyse Comparative des Approches MILP et Synthèse de Contrôleur

La comparaison est réalisée en se fondant sur trois facteurs clés : la modélisation et la facilité d'implémentation, la précision et la pertinence du planning, ainsi que l'adaptabilité et la flexibilité face aux changements, qu'ils soient prévus ou imprévus

- **la modélisation et la facilité d'implémentation**
  La méthode de synthèse du Contrôleur nécessite une connaissance approfondie du système et de l'outil de modélisation utilisé. La précision du modèle initial est cruciale car les étapes suivantes dépendent largement de la qualité de ce modèle. Une fois le modèle établi, le processus requiert principalement un effort de calcul.
  L'approche MILP (Programmation Linéaire en Nombres Entiers Mixtes) dépend également de la compréhension du système. Elle se distingue par l'utilisation de multiples variables de décision, à la fois continues et binaires, et l'intégration de diverses contraintes opérationnelles. Le défi réside dans la gestion de ces variables et contraintes pour parvenir à une solution optimale.

Dans les deux cas, la complexité de la modélisation et la nécessité d'une compréhension détaillée du système sont des facteurs clés. La MILP peut offrir plus de flexibilité dans la gestion des contraintes, mais cela peut également augmenter sa complexité.

- **la précision et la pertinence du planning**
  L'approche MILP génère des solutions directes et rapides. Dans notre modèle, chaque paramètre du système est identifié par une variable de décision dont la valeur est calculée lors de la résolution du modèle d'optimisation.
  Cependant avec cette approche, le suivi des sorties client est basé sur l'accumulation des demandes journalières, qui dépendent de l'injection des batches et des périodes journalières. Comme le débit n'est pas fixe, le suivi de l'état des stocks durant l'injection d'un batch, pouvant durer plusieurs jours, demeure opaque. Il est difficile de suivre les sorties clients de manière continue car le temps dans le programme est discrétisé. Une surveillance approchée nécessiterait un ensemble de temps de dimension élevée, rendant la résolution du programme infaisable dans un délai raisonnable. En contraste, la synthèse de contrôleur, bien que nécessitant une déduction simple des paramètres, permet un suivi plus précis et continu de l'état du pipeline et des stocks à chaque instant.

- **l'adaptabilité et la flexibilité face aux changements;**
  Lorsqu'il s'agit de modifier des données initiales du système, telles que les données journalières ou les changements de stock de sécurité, l'approche MILP se révèle plus pratique. La mise à jour des données, comme la modification des valeurs de demande dans les paramètres, est simple et directe.
  En revanche, avec l'approche de synthèse de contrôleur, une modification des données requiert une modélisation et un calcul entièrement nouveaux. Ceci implique de repartir de l'automate ou même de la modélisation initiale du Réseau de Petri, ce qui peut s'avérer plus complexe et plus long.

  Cependant, en ce qui concerne la gestion des imprévus journaliers, la synthèse de contrôleur en temps réel est nettement plus efficace. Grâce aux intervalles de commutation, elle permet, sans compromettre les exigences préétablies, d'arrêter l'injection en cas de panne ou de planifier facilement des actions de maintenance préventive ou corrective. Cette approche offre également une plus grande flexibilité dans la gestion des commutations dans le pipeline.

Les deux approches présentent des avantages et des inconvénients distincts. La sélection de l'une ou l'autre dépendra des besoins spécifiques de la planification du pipeline multi-produit et des capacités de gestion du système. L'approche MILP offre des solutions directes et rapides tandis que la synthèse de contrôleur en temps réel offre plus de flexibilité et une adaptabilité

supérieures aux imprévus. Le choix entre ces deux approches dépend des besoins spécifiques en matière de planification et de la capacité à gérer la complexité et les changements en temps réel.

4.7 Conclusion

Les recherches menées par Ghomri ont constitué notre point de départ pour notre approche de synthèse du contrôleur. Nous avons développé deux méthodes pour calculer un contrôleur maximal permissif pour un système dynamique hybride modélisé par un automate hybride. Dans la première méthode, nous avons rencontré certains défis lors de son application à notre modèle de pipeline multi-produit. Les obstacles que nous avons rencontrés en utilisant cette stratégie nous ont poussés à explorer une deuxième méthode. Celle-ci repose sur une approche de programmation linéaire en nombres entiers mixtes, où le temps de séjour est considéré comme une variable de décision.

Le processus de synthèse s'est articulé autour de quatre étapes essentielles. La première a été la modélisation initiale du système via le réseau de Petri hybride, offrant une description détaillée du système de pipeline et assurant sa conformité avec diverses contraintes opérationnelles. Par la suite, nous avons traduit le modèle en automates hybrides. Puis, nous avons modélisé les spécifications de manière formelle et élaboré des contraintes sur l'espace d'état atteignable de notre modèle pour garantir le respect des spécifications pendant le temps de séjour dans chaque sommet de l’automate. Un algorithme a été conçu pour modéliser ces contraintes. Enfin, nous avons résolu un problème de programmation linéaire en nombres entiers mixtes en temps réel pour déterminer le temps de séjour dans chaque sommet.

Le contrôleur conçu a répondu aux spécifications liées à la capacité et à l'autonomie de stock, Il permet ainsi de choisir le planning optimal qui réduit au maximum le nombre d'interfaces, en privilégiant des trajectoires optimales.

Conclusion générale et perspectives

Au cours de ce travail, nous avons examiné sur la synthèse de contrôleurs pour un système dynamique hybride : les pipelines multi-produits. Nous avons appliqué cette approche à un cas réel, le pipeline multi-produit ASR. Ce système intègre des dynamiques continues et des événements discrets qui interagissent entre eux. Sa particularité est que la partie discrète pilote la partie continue. En l'absence de commande, l'évolution de ce système pourrait atteindre des régions indésirables, enfreignant ainsi certaines spécifications nécessaires à son bon fonctionnement. Les principales contraintes imposées au système de pipeline concernent l'approvisionnement des produits. Les niveaux de produits doivent demeurer entre deux limites afin de ne pas excéder la capacité de stockage des dépôts tout en conservant un stock de sécurité. Le critère d'optimalité pour ce système est de minimiser le nombre d'interfaces qui résultent du transport successif de carburants.

Nous avons limité l'application de notre méthode à seulement une raffinerie et un dépôt dans le pipeline multi produit ASR, en raison de la taille du modèle. Cette approche peut néanmoins être généralisée à des réseaux de pipelines plus complexes.

Pour satisfaire ces spécifications, nous avons élaboré une modélisation du comportement dynamique du pipeline multi-produit en utilisant les réseaux de Petri hybrides. Ce modèle illustre le fonctionnement du système en boucle ouverte. Notre objectif est de parvenir à un comportement en boucle fermée qui respecte les spécifications imposées. Dans cette optique, nous avons traduit le réseau de Petri hybride, qui décrit le fonctionnement en boucle ouverte, en un automate hybride. Ce dernier a simplifié l'analyse et l'exploration de l'espace d'état atteignable de notre modèle à l’aide à de l'outil PHAVer, fournissant une base formelle solide pour la synthèse de notre contrôleur.

Nous avons étudié deux méthodes distinctes pour calculer notre contrôleur. Les deux se concentrent sur la modification des gardes des transitions contrôlables, afin de quitter le sommet de l’automate avant d'atteindre des régions où les spécifications sont violées. Ces méthodes sont basées sur les formules algébriques développées par Ghomri. La principale différence entre ces méthodes réside dans la manière dont les variables d'états continus sont traitées. Dans la première méthode, ces valeurs sont indéterminées à l'entrée du sommet et sont donc présentées sous forme d'intervalle. Tandis que dans la seconde méthode, cette indétermination est levée en travaillant en temps réel et en considérant le temps de séjour comme une variable de décision.

Dans la deuxième méthode nous avons intégré l'approche de synthèse du contrôleur à la programmation linéaire pour instaurer un contrôle en temps réel. Lorsque l'opérateur envisage d'injecter un nouveau batch, l'approche proposée aborde un problème de programmation

linéaire, maximisant et minimisant le temps de séjour dans le sommet de l'automate atteignable associé à cette opération. Cette stratégie d'optimisation nous a permis de définir le plus large intervalle de commutation, en respectant les spécifications, ce qui nous conduit à un contrôleur maximal permissif. En termes concrets, cet intervalle de temps de séjour représente la plage temporelle pendant laquelle nous pouvons interrompre l'injection du batch actuellement en cours et initier l'injection du nouveau batch. Il est essentiel de mentionner que l'ordre d'injection des produits est représenté par un chemin dans l'automate atteignable et est établi à l'avance. Cet ordre est déterminé sur une période spécifiée en identifiant le chemin optimal dans l'automate atteignable parmi les chemins accessibles, afin de minimiser le nombre d'interfaces.

L'approche de contrôle que nous avons élaborée nous accorde une grande flexibilité, permettant l'intégration aisée d'autres facteurs à optimiser. Elle est réactive face aux changements et imprévus. L'objectif de contrôle a été atteint avec succès, respectant les spécifications imposées par le cahier des charges, notamment en ce qui concerne la capacité et l'autonomie de stockage. Cela garantit la satisfaction des clients. De plus, nous avons réussi à minimiser le contaminât en optant pour des trajectoires optimales.

Cette étude constitue une base solide pour des recherches ultérieures, en particulier en ce qui concerne la généralisation de l'approche de synthèse du contrôleur pour les systèmes hybrides à flux positif. Ces défis représentent des perspectives intéressantes pour nos futures recherches.

Références

Alla, H., & David, R. (1998). Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers*, *8*(01), 159–188.

Alla, H., & Ghomri, L. (2012). *Modeling and simulation by hybrid Petri nets*. 1–8.

Alur, R. (1999). *Timed automata*. 8–22.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., & Yovine, S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, *138*(1), 3–34.

Alur, R., Courcoubetis, C., Henzinger, T. A., & Ho, P.-H. (1993). Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In *Hybrid Systems* (pp. 209–229).

Alur, R., Dang, T., & Ivančić, F. (2006). Predicate abstraction for reachability analysis of hybrid systems. *ACM Transactions on Embedded Computing Systems*, *5*(1), 152–199. https://doi.org/10.1145/1132357.1132363

Alur, R., & Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, *126*(2), 183–235.

Alur, R., Kanade, A., Ramesh, S., & Shashidhar, K. (2008). *Symbolic analysis for improving simulation coverage of Simulink/Stateflow models*. 89–98.

Antsaklis, P. J., Stiver, J. A., & Lemmon, M. D. (1993). Hybrid System Modeling and Autonomous Control Systems. In *Hybrid Systems* (pp. 366–392).

Bemporad, A., & Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, *35*(3), 407–427.

Bennacer, D., Saim, R., Abboudi, S., & Benameur, B. (2016). Interface calculation method improves multiproduct transport. *Oil Gas J*, *114*, 74.

Berrichi, A., Abdellaoui, W., Bennacer, D., Maliki, F., & Ghomri, L. (2021). Detailed scheduling distribution of real multi-product pipeline. *International Journal of Operational Research*, *42*(4), 505–523.

Berthomieu, B., & Diaz, M. (1991). Modeling and verification of time dependent systems using time Petri nets. *IEEE Transactions on Software Engineering*, *17*(3), 259.

Branicky, M. S. (1995). *Studies in hybrid systems: Modeling, analysis, and control*.

Branicky, M. S. (1998). Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, *43*(4), 475–482.

Branicky, M. S., Borkar, V. S., & Mitter, S. K. (1994). *A unified framework for hybrid control*. *4*, 4228–4234.

Brauer, W., & Reisig, W. (2009). Carl adam Petri and "Petri nets." *Fundamental Concepts in Computer Science*, *3*(5), 129–139.

Brice, W. R. (2003). *Frontmatter: Oil-Industry History, Volume 4, Number 1, 2003*.

Cafaro, V. G., Cafaro, D. C., Méndez, C. A., & Cerdá, J. (2012). Detailed scheduling of single-source pipelines with simultaneous deliveries to multiple offtake stations. *Industrial & Engineering Chemistry Research*, *51*(17), 6145–6165.

Cassandras, C. G., & Lafortune, S. (2008). *Introduction to discrete event systems*. Springer.

Cassez, F., & Roux, O. H. (2006). Structural translation from time Petri nets to timed automata. *Journal of Systems and Software*, *79*(10), 1456–1468.

Castro, P. M., & Mostafaei, H. (2019). Batch-centric scheduling formulation for treelike pipeline systems with forbidden product sequences. *Computers & Chemical Engineering*, *122*, 2–18.

Chen, C., Li, C., Reniers, G., & Yang, F. (2021). Safety and security of oil and gas pipeline transportation: A systematic analysis of research trends and future needs using WoS. *Journal of Cleaner Production*, *279*, 123583.

Chen, H., Wu, C., Zuo, L., Diao, F., Wang, L., Wang, D., & Song, B. (2017). Optimization of detailed schedule for a multiproduct pipeline using a simulated annealing algorithm and heuristic rules. *Industrial & Engineering Chemistry Research*, *56*(17), 5092–5106.

Clarke, E., Fehnker, A., Han, Z., Krogh, B., Stursberg, O., & Theobald, M. (2003). *Verification of hybrid systems based on counterexample-guided abstraction refinement*. 192–207.

Csontos, B., Halász, L., & Heckl, I. (2022). Event-driven simulation method for fuel transport in a mesh-like pipeline network. *Computers & Chemical Engineering*, *157*, 107611.

David, R., & Alla, H. (2001). On hybrid Petri nets. *Discrete Event Dynamic Systems*, *11*(1–2), 9–40.

David, R., & Alla, H. (2010). *Discrete, continuous, and hybrid Petri nets* (Vol. 1). Springer.

Dobbe, R., & Tomlin, C. J. (2015). Hybrid Systems Modeling for (Cancer) Systems Biology. *bioRxiv*, 035022.

Frehse, G. (2005). PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech. In M. Morari & L. Thiele (Eds.), *Hybrid Systems: Computation and Control* (pp. 258–273). Springer. https://doi.org/10.1007/978-3-540-31954-2_17

Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., & Maler, O. (2011). *SpaceEx: Scalable verification of hybrid systems*. 379–395.

Ghaffari-Hadigheh, A., & Mostafaei, H. (2015). On the scheduling of real world multiproduct pipelines with simultaneous delivery. *Optimization and Engineering*, *16*, 571–604.

Ghomri, L. (2012). Synthèse de contrôleur de systèmes hybrides à flux continu par réseaux de Petri hybrides. *Thèse de Doctorat*.

Ghomri, L., & Alla, H. (2007). Modeling and analysis using hybrid Petri nets. *Nonlinear Analysis: Hybrid Systems*, *1*(2), 141–153.

Ghomri, L., Alla, H., & Sari, Z. (2005). Structural and hierarchical translation of hybrid Petri nets in hybrid automata. *Proceedings of IMACS'05*.

Haoran, Z., Yongtu, L., Qi, L., Yun, S., & Xiaohan, Y. (2018). A self-learning approach for optimal detailed scheduling of multi-product pipeline. *Journal of Computational and Applied Mathematics*, *327*, 41–63.

Henzinger, T. A. (1996). *The theory of hybrid automata*. 278–292.

Henzinger, T. A., Ho, P.-H., & Wong-Toi, H. (1997). HYTECH: a model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer (STTT)*, *1*(1–2), 110–122.

Henzinger, T. A., Ho, P.-H., & Wong-Toi, H. (1998). Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, *43*(4), 540–554. https://doi.org/10.1109/9.664156

Henzinger, T. A., Kopke, P. W., Puri, A., & Varaiya, P. (1995). *What's decidable about hybrid automata?* 373–382.

Herrán, A., de la Cruz, J. M., & De Andrés, B. (2012). Global Search Metaheuristics for planning transportation of multiple petroleum products in a multi-pipeline system. *Computers & Chemical Engineering*, *37*, 248–261.

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, *32*(1), 60–65.

Janan Zaytoon. (2001). *Systèmes dynamiques hybrides*. HERMES SCIENCE.

Johansson, M., & Rantzer, A. (1997). *Computation of piecewise quadratic Lyapunov functions for hybrid systems*. 2005–2010.

Karp, R. M., & Miller, R. E. (1969). Parallel program schemata. *Journal of Computer and System Sciences*, *3*(2), 147–195.

Khalili Goudarzi, F., Maleki, H. R., & Niroomand, S. (2021). Mathematical formulation and hybrid meta-heuristic algorithms for multiproduct oil pipeline scheduling problem with tardiness penalties. *Concurrency and Computation: Practice and Experience*, *33*(17), e6299.

Kumar, R., Garg, V., & Marcus, S. I. (1991). On controllability and normality of discrete event dynamical systems. *Systems & Control Letters*, *17*(3), 157–168.

Kurovszky, M. (2002). *Etude des systèmes dynamiques hybrides par représentation d'état discrète et automate hybride*.

Li, Z., Liang, Y., Liao, Q., Zhang, B., & Zhang, H. (2021). A review of multiproduct pipeline scheduling: From bibliometric analysis to research framework and future research directions. *Journal of Pipeline Science and Engineering*, *1*(4), 395–406.

Liao, Q., Zhang, H., Wang, Y., Zhang, W., & Liang, Y. (2018). Heuristic method for detailed scheduling of branched multiproduct pipeline networks. *Chemical Engineering Research and Design*, *140*, 82–101.

Liao, Q., Zhang, H., Xu, N., Liang, Y., & Wang, J. (2018). A MILP model based on flowrate database for detailed scheduling of a multi-product pipeline with multiple pump stations. *Computers & Chemical Engineering*, *117*, 63–81.

Lunze, J., & Raisch, J. (2002). *Discrete models for hybrid systems*. 67–80.

Maire, A. (2011). *Le transport par pipeline: Aspects économiques et environnementaux*. Editions Technip.

Martínez-Palou, R., de Lourdes Mosqueira, M., Zapata-Rendón, B., Mar-Juárez, E., Bernal-Huicochea, C., de la Cruz Clavel-López, J., & Aburto, J. (2011). Transportation of heavy and extra-heavy crude oil by pipeline: A review. *Journal of Petroleum Science and Engineering*, *75*(3–4), 274–282.

Możaryn, J., Petryszyn, J., & Ozana, S. (2021). PLC based fractional-order PID temperature control in pipeline: Design procedure and experimental evaluation. *Meccanica*, *56*, 855–871.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541–580.

Nanez, P., Risso, N., & Sanfelice, R. G. (2014). *A symbolic simulator for hybrid equations.*

Pettersson, S., & Lennartson, B. (1995). *Hybrid modelling focused on hybrid Petri nets*. 303–309.

Pettersson, S., & Lennartson, B. (1996). *Stability and robustness for hybrid systems*. *2*, 1202–1207.

Pinto, J. M., Joly, M., & Moro, L. F. L. (2000). Planning and scheduling models for refinery operations. *Computers & Chemical Engineering*, *24*(9–10), 2259–2276.

Priyanka, E., Maheswari, C., & Thangavel, S. (2018). Online monitoring and control of flow rate in oil pipelines transportation system by using PLC based Fuzzy-PID Controller. *Flow Measurement and Instrumentation*, *62*, 144–151.

Priyanka, E., Maheswari, C., & Thangavel, S. (2019). Remote monitoring and control of LQR-PI controller parameters for an oil pipeline transport system. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *233*(6), 597–608.

Priyanka, E., & Thangavel, S. (2020). *Decision making based on machine learning algorithm for identifying failure rates in the oil transportation pipeline*. 914–919.

Puri, A., Borkar, V., & Varaiya, P. (1996). *ε-approximation of differential inclusions*. 362–376.

Puri, A., & Varaiya, P. (1994). *Decidability of hybrid systems with rectangular differential inclusions*. 95–104.

Ramadge, P. J., & Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, *25*(1), 206–230.

Raskin, J.-F. (2005). An introduction to hybrid automata. In *Handbook of networked and embedded control systems* (pp. 491–517). Springer.

Rejowski Jr, R., & Pinto, J. M. (2003). Scheduling of a multiproduct pipeline system. *Computers & Chemical Engineering*, *27*(8–9), 1229–1246.

Sava, A. T. (2001). *Sur la synthèse de la commande des systèmes à évènements discrets temporisés*.

Sava, A. T., & Alla, H. (2001). Combining hybrid Petri nets and hybrid automata. *IEEE Transactions on Robotics and Automation*, *17*(5), 670–678.

Seatzu, C., Silva, M., & Van Schuppen, J. H. (2013). *Control of discrete-event systems* (Vol. 433). Springer.

Shaikh, M. S., & Caines, P. E. (2007). On the hybrid optimal control problem: Theory and algorithms. *IEEE Transactions on Automatic Control*, *52*(9), 1587–1603.

Silva, J. R., & Del Foyo, P. M. (2012). Timed Petri nets. In *Petri Nets: Manufacturing and Computer Science* (pp. 359–378). InTech.

Souas, F., Safri, A., & Benmounah, A. (2021). A review on the rheology of heavy crude oil for pipeline transportation. *Petroleum Research*, *6*(2), 116–136.

Stéphanou, A., & Volpert, V. (2015). Hybrid modelling in cell biology. *Mathematical Modelling of Natural Phenomena*, *10*(1), 1–1.

Stiver, J. A., Antsaklis, P. J., & Lemmon, M. D. (1996). A logical DES approach to the design of hybrid control systems. *Mathematical and Computer Modelling*, *23*(11–12), 55–76.

Stiver, J. A., Koutsoukos, X. D., & Antsaklis, P. J. (2001). An invariant-based approach to the design of hybrid control systems. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, *11*(5), 453–478.

Wang, K., Wang, Z. M., & Song, G.-L. (2020). Batch transportation of oil and water for reducing pipeline corrosion. *Journal of Petroleum Science and Engineering*, *195*, 107583.

Witsenhausen, H. (1966). A class of hybrid-state continuous-time dynamic systems. *IEEE Transactions on Automatic Control*, *11*(2), 161–167.

Ye, H., Michel, A. N., & Hou, L. (1998). Stability theory for hybrid dynamical systems. *IEEE Transactions on Automatic Control*, *43*(4), 461–474.

Yongtu, L., Ming, L., & Ni, Z. (2012). A study on optimizing delivering scheduling for a multiproduct pipeline. *Computers & Chemical Engineering*, *44*, 127–140.

Yovine, S. (1993). *Méthodes et outils pour la vérification symbolique de systèmes temporisés*.

Yovine, S. (1997). Kronos: A verification tool for real-time systems. *Int. J. Softw. Tools Technol. Transf.*, *1*(1–2), 123–133.

Zhu, F., & Antsaklis, P. J. (2015). Optimal control of hybrid switched systems: A brief survey. *Discrete Event Dynamic Systems*, *25*, 345–364.

ANNEXE : Programme du fichier PHAVer et résultats après compilation sur Cygwin

Cette annexe est dédiée à la présentation du fichier pipeline.pha associé à l'automate hybride décrivant le fonctionnement du pipeline (figure 3-4). La compilation de ce fichier via Cygwin nous fournit l'espace d'état atteignable pour chaque sommet de cet automate, également présenté dans cette annexe.

1- Programme du fichier (pipeline.pha)

automaton pipeline

state_var: C0,C1, C2, C3, C4, m13, m14, m15, m16;

synclabs: T1, T3, T5, T6,T7,T8,T9 ,T10 ,T11 ,T12 ,T13;

loc L1: while m16>=0 & C0<=100 wait {m13'== 0 & m14'== 75 & m15'==0 & m16'== -34 & C0'== 1 & C1'== 1 & C2' == 0 & C3' == 0 & C4'==0}

when C1 >= 0 sync T1 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== 0 & C3'== 0 & C4'==C4} goto L3;

when m16 == 0 sync T7 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L1x;

when C0>=100 sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L3: while C3 <= 50.5 & m16>=0 & C0<=100 wait {m13'== -200 & m14'== 75 & m15'== 200 & m16'== -34 & C0'==1 & C1'== 0 & C2' == 1 & C3' == 1 & C4'==0}

when C2 >= 0 sync T3 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== 0 & C2'== C2 & C3'== C3 & C4'==0} goto L6;

when C3 == 50.5 sync T5 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L7;

when m16 == 0 sync T8 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L3x;

when C0>=100 sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L6: while C3 <= 50.5 & m16>=0 & C0<=100 wait {m13'== 0 & m14'== 75 & m15'== 0 & m16'== -34 & C0'== 1 & C1'== 1 & C2' == 0 & C3' == 1 & C4'==1}

when C3 == 50.5 sync T5 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L12;

when m16 == 0 sync T9 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L6x;

when C0>=100 sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L7: while m14>=0 & C0<=100 wait {m13'== 0 & m14'== -125 & m15'== 0 & m16'== 166 & C0'== 1 & C1'== 0 & C2' == 1 & C3' == 0 & C4'==0}

when C2 >= 0 sync T3 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== 0 & C2'== C2 & C3'== C3 & C4'==0 } goto L12;

when m14 == 0 sync T10 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L7x;

when C0>=100 sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L12: while C4 <= 50.5 & m14>=0 & C0<=100 wait {m13'== 200 & m14'== -125 & m15'== -200 & m16'== 166 & C0'== 1 & C1'== 1 & C2' == 0 & C3' == 0 & C4'==1 }

when C4 == 50.5 sync T6 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L1;

when C1 >= 0 sync T1 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== 0 & C3'== 0 & C4'==C4 } goto L15;

when m14 == 0 sync T11 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L12x;

when C0>=100 sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L15: while C4 <= 50.5 & m14>=0 & C0<=100 wait {m13'== 0 & m14'== -125 & m15'== 0 & m16'== 166 & C0'== 1 & C1'== 0 & C2' == 1 & C3' == 1 & C4'==1 }

when C4 == 50.5 sync T6 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4 } goto L3;

when m14 == 0 sync T12 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L15x;

when C0>=100 sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L1x: while C0<=100 wait {m13'== 0 & m14'== 75 & m15'==0 & m16'== 0 & C0'== 1 & C1'== 1 & C2' == 0 & C3' == 0 & C4'==0}

when C1 >= 0 sync T1 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== 0 & C3'== 0 & C4'==C4} goto L3x;

when C0>=100 sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L3x: while C3 <= 50.5 & C0<=100 wait {m13'== -200 & m14'== 75 & m15'== 200 & m16'== 0 & C0'==1 & C1'== 0 & C2' == 1 & C3' == 1 & C4'==0}

when C2 >= 0 sync T3 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== 0 & C2'== C2 & C3'== C3 & C4'==0} goto L6x;

when C3 == 50.5 sync T5 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L7;

when C0>=100 sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L6x: while C3 <= 50.5 & C0<=100 wait {m13'== 0 & m14'== 75 & m15'== 0 & m16'== 0 & C0'== 1 & C1'== 1 & C2' == 0 & C3' == 1 & C4'==1}

when C3 == 50.5 sync T5 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L12;

when C0>=100 sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L7x: while C0<=100 wait {m13'== 0 & m14'== 0 & m15'== 0 & m16'== 166 & C0'== 1 & C1'== 0 & C2' == 1 & C3' == 0 & C4'==0}

when C2 >= 0 sync T3 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== 0 & C2'== C2 & C3'== C3 & C4'==0 } goto L12x;

```
when C0>=100  sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L12x: while C4 <= 50.5 & C0<=100 wait {m13'== 200 & m14'== 0 & m15'== -200 & m16'== 166 & C0'== 1 & C1'== 1 & C2' == 0 & C3' == 0 & C4'==1 }

when C4 == 50.5  sync T6 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4} goto L1;

when C1 >= 0  sync T1 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== 0 & C3'== 0 & C4'==C4 } goto L15x;

when C0>=100  sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

loc L15x: while C4 <= 50.5 & C0<=100 wait {m13'== 0 & m14'== 0 & m15'== 0 & m16'== 166 & C0'== 1 & C1'== 0 & C2' == 1 & C3' == 1 & C4'==1 }

when C4 == 50.5 sync T6 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== C0 & C1'== C1 & C2'== C2 & C3'== C3 & C4'==C4 } goto L3;

when C0>=100  sync T13 do {m13'== m13 & m14'== m14 & m15'== m15 & m16'== m16 & C0'== 0 & C1'== 0 & C2'== 0 & C3'== 0 & C4'==0} goto final;

 loc final: while true  wait {m13'== 0 & m14'== 0 & m15'==0 & m16'== 0 & C0'== 0 & C1'== 0 & C2' == 0 & C3' == 0 & C4'==0}

initially L1 & C0== 0 & C1== 0 & C2==0 & C3==0 & C4==0  & m13==10100 & m14==5572 & m15==0 & m16==4350;

end

echo "analyse en avant";

analyse_avant = pipeline.reachable;

analyse_avant.print;
```

2- Espace atteignable de l'automate de la figure 3-4

Le résultat après la compilation du fichier PHAVer sur Cygwin est illustré comme suit :

```
L1 &
```

(34*C0 - 200*C2 + m16 == 4350 & m15 == 0 & 75*C0 - 200*C2 - m14 == -5572 & m13 = = 10100 & 2*C4 == 101 & 2*C3 == 101 & -C0 >= -100 & C2 >= 0 & 2*C1 >= 101 & C0 - C1 - C2 >= 0 | 34*C0 + m16 == 4350 & m15 == 0 & 75*C0 - m14 == -5572 & m13 == 1 0100 & C4 == 0 & C3 == 0 & C2 == 0 & C0 - C1 == 0 & C0 >= 0 & -C0 >= -100 | m16 == 4350 & m15 == 0 & m14 == 5572 & m13 == 10100 & C4 == 0 & C3 == 0 & C2 == 0 & C1 == 0 & C0 == 0),
L3 &
(41*C0 - m14 - m16 == -9922 & 200*C2 - m15 == 0 & 200*C2 + m13 == 10100 & 2*C4 = = 101 & C2 - C3 == 0 & -C0 >= -100 & -2*C1 >= -101 & 2*C1 + 2*C2 >= 101 & 75*C0 + 200*C1 - m14 >= 4528 & 125*C0 - 200*C1 - 200*C2 + m14 >= 5572 | 41*C0 - m14 - m16 == -9922 & 200*C2 - m15 == 0 & 200*C2 + m13 == 10100 & 2*C4 == 101 & C2 - C3 == 0 & -C0 >= -100 & C2 >= 0 & 2*C1 >= 101 & 75*C0 - m14 >= -5572 & 125*C0 - 20 0*C1 - 200*C2 + m14 >= 5572 | 34*C0 + m16 == 4350 & 200*C0 - 200*C1 - m15 == 0 & 75*C0 - m14 == -5572 & 200*C0 - 200*C1 + m13 == 10100 & C4 == 0 & C0 - C1 - C3 == 0 & C0 - C1 - C2 == 0 & C1 >= 0 & C0 - C1 >= 0 & -2*C0 + 2*C1 >= -101 & -C0 > = -100),
final &
(m14 + m16 == 14022 & m13 + m15 == 10100 & C4 == 0 & C3 == 0 & C2 == 0 & C1 == 0 & C0 == 0 & -m15 >= -10100 & -m16 >= -10850 & m16 >= 950 & m15 >= 0 | m14 + m16 == 14022 & m13 + m15 == 10100 & C4 == 0 & C3 == 0 & C2 == 0 & C1 == 0 & C0 == 0 & -m15 >= -10100 & -m16 >= -10850 & m15 + m16 >= 11050 | m14 + m16 == 14022 & m 13 + m15 == 10100 & C4 == 0 & C3 == 0 & C2 == 0 & C1 == 0 & C0 == 0 & -m15 - m16 >= -11050 & -m16 >= -10850 & m16 >= 950 & m15 >= 0 | m15 == 10100 & m14 + m16 = = 14022 & m13 == 0 & C4 == 0 & C3 == 0 & C2 == 0 & C1 == 0 & C0 == 0 & -m16 >= - 10850 & m16 >= 950 | m16 == 950 & m14 == 13072 & m13 + m15 == 10100 & C4 == 0 & C3 == 0 & C2 == 0 & C1 == 0 & C0 == 0 & -m13 >= -10100 & m13 >= 0 | m16 == 950 & m15 == 0 & m14 == 13072 & m13 == 10100 & C4 == 0 & C3 == 0 & C2 == 0 & C1 == 0 & C0 == 0),
L6 &
(41*C0 - m14 - m16 == -9922 & 200*C2 - m15 == 0 & 200*C2 + m13 == 10100 & C1 - t 4 == 0 & C1 + C2 - C3 == 0 & -C0 >= -100 & C2 >= 0 & C1 >= 0 & 75*C0 - m14 >= -5 572 & 125*C0 - 200*C1 + m14 >= 15672 & 2*C0 - 2*C1 - 2*C2 >= 101 | 41*C0 - m14 - m16 == -9922 & 200*C2 - m15 == 0 & 200*C2 + m13 == 10100 & C1 - C4 == 0 & C1 + C2 - C3 == 0 & -C0 >= -100 & C2 >= 0 & C1 >= 0 & 75*C0 - m14 >= -5572 & 125*C0 - 200*C1 - 200*C2 + m14 >= 15672 | 34*C0 + m16 == 4350 & 200*C2 - m15 == 0 & 75*t 0 - m14 == -5572 & 200*C2 + m13 == 10100 & C1 - C4 == 0 & C1 + C2 - C3 == 0 & C0 - C1 - C2 >= 0 & -2*C1 - 2*C2 >= -101 & C2 >= 0 & C1 >= 0 & -C0 >= -100),
L7 &
166*C0 - 200*C1 - m16 == 5750 & m15 == 10100 & 125*C0 - 200*C1 + m14 == 15672 & m13 == 0 & C4 == 0 & 2*C3 == 101 & C0 - C1 - C2 == 0 & -C0 >= -100 & 2*C0 - 2*C1 >= 101 & C1 >= 0,
L12 &
(34*C0 - 200*C1 - 200*C2 + m16 == -5750 & 200*C1 + m15 == 10100 & 75*C0 - 200*C1 - 200*C2 - m14 == -15672 & 200*C1 - m13 == 0 & C1 - C4 == 0 & 2*C3 == 101 & -C0 >= -100 & C1 >= 0 & 2*C2 >= 101 & C0 - C1 - C2 >= 0 | 34*C0 - 200*C1 - 200*C2 + m16 == -5750 & 200*C1 + m15 == 10100 & 75*C0 - 200*C1 - 200*C2 - m14 == -15672 & 200*C1 - m13 == 0 & C1 - C4 == 0 & 2*C3 == 101 & -2*C1 >= -101 & -2*C2 >= -101 & -C0 >= -100 & 2*C1 + 2*C2 >= 101 & C0 - C1 - C2 >= 0),
L15 &
(41*C0 - m14 - m16 == -9922 & 200*C1 + m15 == 10100 & 200*C1 - m13 == 0 & C1 + t 2 - C4 == 0 & C2 - C3 == 0 & -C0 >= -100 & C1 >= 0 & C2 >= 0 & 75*C0 - 200*C1 - 200*C2 - m14 >= -5572 & 125*C0 + m14 >= 15672 | 41*C0 - m14 - m16 == -9922 & 200 *C1 + m15 == 10100 & 200*C1 - m13 == 0 & C1 + C2 - C4 == 0 & C2 - C3 == 0 & 75*t 0 - 200*C2 - m14 >= -5572 & -2*C1 - 2*C2 >= -101 & 125*C0 + m14 >= 15672 & C2 >= 0 & -C0 >= -100 & -75*C0 + 200*C1 + 200*C2 + m14 >= 5572)}