UNIVERSITÉ PARIS VI - PIERRE ET MARIE CURIE

U.F.R. D'INFORMATIQUE

THÈSE DE DOCTORAT

En vue de l'obtention du grade de
docteur de l'université Paris VI
en informatique

Présentée par

Olivier MARCHETTI

DIMENSIONNEMENT DES MÉMOIRES POUR SYSTÈMES EMBARQUÉS

Soutenue le 4 Décembre 2006 devant le jury composé de :

M.	Bruno GAUJAL	rapporteur
M^{me}	Nathalie SAUER	rapporteur
M.	Jean-Marc DELOSME	examinateur
M^{me}	Nathalie DRACH-TEMAM	examinateur
M^{me}	Claire HANEN	examinateur
M^{me}	Alix MUNIER-KORDON	directrice de thèse
M.	Yves SOREL	examinateur
M.	Pascal URARD	examinateur

Cette thèse a été co-financée grâce un partenariat entre le CNRS et l'entreprise ST-Microelectronics (bourse CNRS BDI-entreprise). Aussi, je tiens à exprimer ici toute ma gratitude à ces deux entités.

Remerciements

Je tiens à exprimer toute ma gratidude...

... à Bruno Gaujal, directeur de recherche à l'INRIA, et à Nathalie Sauer, professeur à l'université de Metz, pour l'honneur qu'ils m'ont fait en acceptant d'être les rapporteurs de ce manuscrit...

... à Pascal Urard, *high level synthesis manager* à ST - Microelectronics, pour l'intérêt qu'il a manifesté à l'égard de mes travaux tout au long de ces trois années de doctorat, ainsi que pour sa participation à ce jury...

... à Jean-Marc Delosme, professeur à l'université d'Evry, et à Nathalie Drach-Temam, professeur à l'université Pierre et Marie Curie, pour avoir accepté d'être examinateurs de ce travail...

... à Claire Hanen, professeur à l'université de Paris X - Nanterre, pour sa participation à ce jury et pour son intérêt manifesté à l'égard de mes travaux depuis mon stage de DEA...

... à tous les membres de l'équipe AOSTE de l'INRIA Rocquencourt pour l'accueil qu'ils m'ont réservé lors de mes tout premiers pas dans le monde de la recherche informatique et tout particulièrement à Yves Sorel, directeur de recherche INRIA, pour m'avoir donné l'opportunité de venir régulièrement au sein de son équipe et pour avoir accepté d'examiner ce travail...

... à ma directrice de thèse Alix Munier-Kordon, professeur de l'université Paris XII - Créteil, très chaleureusement, d'une part pour m'avoir fait partager sa passion pour la recherche en informatique : ses conseils avisés ont été une aide précieuse durant ces trois années de doctorat ; d'autre part pour avoir considéré avec intérêt et patience toutes mes idées sur ce sujet de thèse ; enfin pour sa grande disponibilité et le soutien qu'elle m'a apporté en toutes circonstances...

... à Alain Greiner, professeur à l'université Pierre et Marie Curie, pour m'avoir accueilli au sein du laboratoire SoC (anciennement ASIM) et pour son intérêt concernant les techniques d'optimisations appliquées à la CAO...

... à Shahin et Chantale qui adoucissent toutes nos tracasseries administratives, mais aussi à Manuel et ses 10 ans d'expérience qui font toujours la différence, tout comme à Pierre pour ses coups de pouce techniques bienvenus, et à Emmanuel pour ses heureuses corrections orthographiques *in extremis*...

... à tous les membres de l'équipe ASIM et plus particulièrement à Roselyne, Patricia et Karine pour leurs précieux conseils aux moments-charnière de ma thèse...

... à Totol, auprès de qui j'ai beaucoup appris...

... à ma famille, qui m'a toujours soutenu et encouragé...

... enfin, à Sara, ma bonne étoile.

Résumé

Le dimensionnement des mémoires pour un système embarqué consiste à définir la taille à allouer pour chaque mémoire de sorte que la surface globale occupée par ces mémoires soit minimale et que le système puisse fonctionner sans blocage. Malgré l'importance industrielle de ce problème, les techniques d'optimisation actuelles sont rapidement mises en défaut par l'explosion combinatoire associée à l'usage des modèles mathématiques existants.

Dans cette thèse, nous nous proposons de définir des outils algorithmiques permettant de contourner cette contrainte afin de résoudre efficacement ce problème d'optimisation. Le modèle utilisé est celui des graphes d'événements généralisés (GEG en abrégé).

Dans une première partie, nous montrons que ce problème d'optimisation est étroitement lié au problème de la vivacité d'un GEG. Nous proposons une transformation des GEG, appelée normalisation, permettant de définir une condition suffisante de vivacité ainsi qu'un algorithme polynomial pour tester cette condition sur des GEG quelconques. En nous appuyant sur cette condition suffisante, nous déterminons également une borne inférieure sur la taille des mémoires. Nous définissons ensuite un algorithme polynomial qui résout de façon exacte le problème de marquage.

Dans la seconde partie, nous étudions ce problème d'optimisation en tenant compte du débit des transitions. Nous proposons des résultats nouveaux de complexité pour des problèmes d'optimisation bi-critère (*i.e.* surface globale / débit du système). Nous développons un algorithme polynomial 2-approché pour la minimisation de la surface pour un débit maximum intrinsèque.

Mots clefs

Dimensionnement des mémoires, graphe d'événements généralisé temporisé, ordonnancement cyclique, normalisation, vivacité , complexité théorique, algorithme d'approximation.

Abstract

The sizing buffer problem considered in this thesis consists in minimizing the surface area of the buffers such that no deadlock appears. Despite its industrial importance, most of optimization approaches do not seem to be well suited for this problem due to the combinatorial explosion. The aim of this thesis is to develop new algorithmic tools to solve efficiently the sizing buffer problem modelled using generalized event graphs (GEG in short).

It is shown in the first part that this problem is strongly related to the liveness of a GEG. An original transformation, called normalization, is then presented to simplify significantly the structure of the graph. A sufficient condition of liveness and a lower bound on the buffers sizes are then expressed. Lastly, a polynomial time algorithm is presented to solve the marking problem.

In the second part, the throughput optimization is considered. New theoretical complexity results for the bi-criteria optimization problem (*i.e.* surface area *vs.* throughput of the system) are provided. A 2-approximation algorithm is then presented for the minimization of the surface area with a maximum intrinsic throughput.

Keywords

Buffers sizing problem, timed generalized event graph, cyclic scheduling, normalisation, liveness, theoretical complexity, approximation algorithm.

En essayant continuellement on finit par réussir.
Donc : plus ça rate, plus on a de chances que ça marche.
(Devise shadok)

Table des matières

Table des figures

Chapitre 1

Introduction

De nos jours, les applications informatiques embarquées sont devenues omniprésentes : avionique, automobile, baladeur multimédia, téléphonie mobile... La conception de ces systèmes fait interagir de nombreux domaines de l'informatique et des télécommunications. Les spécifications de ces systèmes sont de plus en plus nombreuses et font intervenir des critères de performance tels que le débit de l'application. En outre, la vérification de ces spécifications est indispensable pour ce type d'applications. Le problème du dimensionnement des mémoires pour systèmes embarqués est devenu ces dernières années un problème crucial pour l'industrie. En effet, ces systèmes sont réalisés sur des puces électroniques dont les coûts de fabrication sont étroitement liés au coût du silicium. La surface occupée sur ces puces par les mémoires est très importante et leur consommation électrique est non négligeable ce qui peut être pénalisant par rapport aux spécifications initiales du système.

La nécessité de définir une méthodologie et des techniques algorithmiques efficaces s'est imposée afin de simplifier et de sécuriser la phase de conception. Depuis une vingtaine d'années, les modèles développés sont devenus adéquats pour spécifier la plupart des systèmes embarqués actuels. Cependant, malgré de nombreuses approches distinctes, les techniques algorithmiques employées pour la vérification des spécifications de ces systèmes se révèlent inopérantes sur des systèmes de grande taille. Les systèmes embarqués connaissent actuellement un développement important. La concurrence économique s'est également accrue entre les différents acteurs de ce marché. La mise au point de méthodes algorithmiques efficaces pour le développement de ces systèmes représente un serieux atout. Dans ce contexte, le problème du dimensionnement des mémoires pour systèmes embarqués est d'une importance majeure.

L'objectif de cette thèse est de proposer des méthodes de complexité polynomiale pour ce problème. Nous voulons également définir une approche nouvelle dans l'analyse du problème bi-critère qui consiste à minimiser la somme des tailles de mémoires tout en garantissant que l'application atteint un certain débit.

Le chapitre 2 est consacré à l'état de l'art. Nous présentons dans un premier temps trois modèles importants pour la description des systèmes à événements discrets. Nous examinons ensuite les travaux menés sur le problème du dimensionnement des mémoires par la communauté scientifique. Nous exposons les résultats obtenus sur un modèle équivalent utilisé dans la communauté réseaux de Petri.

Le chapitre 3 définit rigoureusement le problème du dimensionnement des mémoires pour systèmes embarqués. Nous introduisons également le modèle utilisé dans ce document (*i.e. le modèle des graphes d'événements généralisés*). Nous formulons également les résultats importants qui s'y rapportent. Nous montrons ensuite que ce modèle est particulièrement bien adapté pour modéliser les problèmes d'ordonnancement cyclique. Nous rappelons alors les résultats obtenus par Chrétienne [Chr83] sur le modèle des graphes d'événements temporisés ainsi que la généralisation de ces résultats faite par Munier [Mun91]. Nous évoquons ensuite les aspects algorithmiques de ces méthodes en soulignant les différences de complexité des méthodes entre le modèle des graphes d'événements et le modèle des graphes d'événements généralisés.

Ce mémoire s'articule autour de deux parties. La première partie se concentre sur les problèmes liés à la vivacité des graphes d'événements généralisés. Nous introduisons dans la seconde partie le débit dans notre analyse du problème du dimensionnement des mémoires. Nous cherchons alors à optimiser le dimensionnement des mémoires tout en maximisant le débit résultant de l'application.

Partie I : Vivacité

Dans cette partie, nous étudions le problème de la vivacité d'un graphe d'événements généralisé et nous en déduisons un algorithme polynomial pour construire un marquage initial vivant qui minimise la taille des mémoires.

Au chapitre 4, nous étudions le problème de la vivacité d'un graphe d'événements généralisé. La mise au point de techniques algorithmiques pour étudier cette question est un problème central. En effet, les applications embarquées ciblées dans ce mémoire sont spécifiées pour pouvoir s'exécuter durant une durée non bornée *a priori*. Le concepteur doit donc vérifier si le système embarqué vérifie bien cette propriété. L'analyse de la vivacité est un problème classique dans la communauté réseaux de Petri. Cependant, les techniques algorithmiques utilisées pour l'analyse de la vivacité de ces systèmes embarqués sont toutes limitées intrinsèquement par l'explosion combinatoire associée à leur complexité pseudo-polynomiale. Les instances pouvant être traitées sont donc de taille limitée.

Afin de pallier à cet inconvénient, nous avons mis au point une condition suffisante de vivacité vérifiable en temps polynomial. Pour cela, nous définissons au préalable une transformation du graphe d'événements généralisé : la normalisation. Cette transformation a pour but de simplifier l'analyse de la vivacité en observant que le nombre de jetons présents sur tout circuit d'un graphe normalisé demeure constant quelle que soit la séquence de franchissements valide considérée. Nous démontrons que tout graphe unitaire peut être normalisé en utilisant un algorithme polynomial de type algorithme de plus courts chemins. Nous proposons alors une condition suffisante de vivacité vérifiable en temps polynomial pour les graphes normalisés. Nous démontrons en outre que cette condition suffisante s'avère également être nécessaire pour les circuits composés de deux transitions.

Dans ce même chapitre, nous proposons également trois propriétés permettant de simplifier et de comparer les ensembles de contraintes de précédence induites par une place et son marquage initial. Nous définissons notamment la notion de jetons utiles qui permet de restreindre le nombre des marquages à considérer. Cette notion de marquages utiles est indispensable pour démontrer l'optimalité de certains résultats contenus dans cette thèse.

Dans le chapitre 5, nous définissons et résolvons le problème de marquage. Etant donnée une application embarquée, le problème de marquage consiste à définir une taille pour chaque mémoire ainsi que le nombre de données présentes initialement dans chaque mémoire de sorte que la taille globale des mémoires soit minimale et que l'application puisse s'exécuter indéfiniment sans blocage. Ce problème est important dans un contexte d'optimisation de la surface des mémoires. Il a été abordé par Adé dans sa thèse de doctorat [Adé96]. Pour résoudre ce problème, elle propose de résoudre le problème de marquage sur des motifs particuliers de graphes *synchronous dataflow* et élabore une heuristique basée sur une décomposition du graphe de l'application en motifs élémentaires.

Nous reformulons le problème de marquage à l'aide du modèle des graphes d'événements généralisés. Le problème de marquage consiste alors à définir une capacité pour chaque place ainsi qu'un marquage initial tel que le graphe d'événements généralisé soit vivant et tel que la somme des capacités soit minimum. Nous définissons formellement dans un premier temps la notion de graphe d'événements généralisé borné que nous caractérisons à l'aide de propriétés structurelles. Nous montrons alors que l'on peut modéliser la contrainte de capacité associée à une place $p = (t_i, t_j)$ en ajoutant simplement une place retour $p' = (t_j, t_i)$ avec un certain marquage initial. Les contraintes de précédence induites par l'ajout de cette place retour p' entre les transitions t_j et t_i correspondent alors aux contraintes de précédence induites par la capacité de la place p. Le graphe d'événements généralisé résultant de l'ajout de ces places retour est alors symétrique et est appelé graphe à capacité. En nous appuyant sur la condition nécessaire et suffisante de vivacité définie au chapitre précédent, nous déduisons une borne inférieure de la capacité à allouer à une place (ce résultat est analogue à ceux obtenus par [Adé96] et [Mur96]). En restreignant les marquages possibles, nous transformons le problème de marquage en un problème de valuation d'arcs sur un graphe orienté associé. Nous proposons enfin un algorithme polynomial pour la résolution du problème de marquage ainsi qu'une généralisation de cet algorithme pour les graphes non fortement connexes.

Partie II : Etude de problèmes d'optimisation bi-critère

Dans cette partie nous étudions plusieurs problèmes d'ordonnancement cyclique bi-critère. Nous étudions l'incidence de la limitation de la taille des mémoires d'un système embarqué sur le débit de l'application. Les critères considérés dans ces problèmes sont la minimisation de la taille globale des mémoires et la maximisation du débit des processus de l'application. Ces problèmes constituent une extension logique du problème initial de la minimisation de la taille globale des mémoires d'un système embarqué.

Nous étudions au chapitre 6 la complexité de problèmes d'ordonnancement cyclique bi-critère. Après avoir résolu le problème de marquage, nous avons voulu étendre notre analyse en incorporant le critère du débit des transitions. Afin de caractériser la difficulté algorithmique de ce problème, nous nous sommes restreint à l'étude du problème de décision pour le problème de marquage avec débit pour des instances non-généralisées. Nous avons alors exhibé une relation forte entre ce problème et le problème de décision Flot Ratio étudié par Minty [Min62]. En utilisant le lemme de Minty [Min62], nous avons obtenu la NP-complétude du problème de décision Flot Ratio. Nous en déduisons que la détermination d'un marquage minimalement borné vérifiant une contrainte sur le débit des transitions est un problème NP-complet. En combinant ce résultat au lemme de Minty, nous pouvons mettre en relation ce problème d'ordonnancement cyclique bi-critère au problème de la K-coloration ouvrant ainsi de nouvelles perspectives pour la mise au point d'heuristiques.

En caractérisant la complexité de ce problème d'ordonnancement cyclique relativement artificiel, nous pouvons en déduire la complexité du problème plus général de la conception d'un marquage utilisant un quantité limitée de jetons conférant aux transitions du graphe un certain débit. Nous démontrons également la NP-complétude de deux problèmes classiques en conception de chaîne d'assemblage : Optimisation de marquage et Débit maximum intrinsèque. Ces deux problèmes ont largement été étudiés dans leur version non-généralisée et résolus par des approches heuristiques. Toutefois, la complexité théorique de ces deux problèmes n'avait jamais été formellement établie. Nous montrons que ces deux problèmes sont effectivement NP-complet et nous proposons un algorithme polynomial 2-approché pour le problème Débit maximum intrinsèque.

Dans le chapitre 7, nous montrons que l'algorithme 2-approché défini pour le problème Débit maximum intrinsèque dans sa version non-généralisée fournit également une solution 2-approchée pour des instances généralisées. Nous identifions également un ensemble d'instances pour lesquelles l'algorithme proposé construit une solution optimale. Le problème Débit maximum intrinsèque dans sa version généralisée n'étant pas dans NP, la réalisation d'heuristiques efficaces est particulièrement difficile. De plus, les méthodes actuelles d'évaluation du débit pour ces instances étant de complexité pseudo-polynomiale, la solution fournie par notre algorithme constitue une base intéressante pour la réalisation d'heuristiques pour ces problèmes d'ordonnancement bi-critère.

*

Nos travaux ont montré que le problème du dimensionnement des mémoires pour systèmes embarqués était un problème difficile. Cependant, nos travaux mettent en lumière certaines structures qui étaient ignorées jusqu'à présent. L'introduction de la normalisation a permis de considérablement simplifier notre vision du problème et de mettre en évidence une condition suffisante de vivacité. Nous avons également exprimé la contrainte sur la taille d'une mémoire en utilisant la notion de place retour. De plus, nous avons montré que l'on pouvait résoudre efficacement le problème de marquage en contournant la difficulté algorithmique de la vivacité inhérente au modèle en utilisant judicieusement notre condition suffisante de vivacité.

Nous avons également établi la complexité théorique de problèmes d'ordonnancement cyclique bi-critères en montrant un lien fort avec le problème de la K-coloration. Nous avons développé un algorithme polynomial 2-approché pour le problème Débit maximum intrinsèque dans sa version généralisée. Ce résultat constitue une base solide pour la mise au point d'heuristiques d'optimisation efficaces. Cependant la non appartenance apparente à la classe NP de l'évaluation du débit rend cette conception délicate.

Nous espérons que nos travaux ouvriront de nouvelles perspectives pour l'analyse et la conception des systèmes embarqués en particulier dans la mise au point d'heuristiques efficaces pour ces problèmes.

Chapitre 2

Etat de l'art

Sommaire

Avec l'avènement des calculateurs modernes et de la micro-électronique, les chercheurs et les ingénieurs ont eu besoin de développer des modèles de plus en plus précis afin de formaliser la notion de calcul parallèle et distribué. Ces modèles ont dès lors subi des évolutions en interaction avec le développement des techniques. Nous présentons de façon synthétique dans une première section trois modèles généraux importants : les *réseaux de Petri*, les *computation graphs* et les *réseaux de Kahn*. Ces trois modèles pour le calcul parallèle définissent chacun un socle solide duquel découle la plupart des modèles développés ces trente dernières années. Nous présentons ensuite les modèles *dataflow* et *synchronous dataflow*. Enfin, nous évoquons le modèle des réseaux de Petri en soulignant les similitudes avec le modèle précédent.

2.1 Trois modèles pour les systèmes à événements discrets

Le modèle des réseaux de Petri a été introduit en 1962 par Carl Adam Petri [Pet62] dans sa thèse. A l'origine, les réseaux de Petri avaient été conçus pour proposer une alternative aux modèles des automates. Ils ont cependant connu un large succès de par leur formidable capacité à modéliser les systèmes à événements discrets. Les réseaux de Petri définissent un outil adapté pour l'analyse des propriétés des systèmes dynamiques. On distingue les propriétés structurelles (liées à la structure du réseau) et les propriétés comportementales (liées au fonctionnement du réseau). L'analyse de tels systèmes consiste alors à relier les propriétés structurelles aux propriétés comportementales.

Une caractéristique intéressante pour les réseaux de Petri est la vivacité ou l'absence de blocage. La vivacité d'un réseau de Petri permet de dire qu'il existe un fonctionnement de l'application modélisée qui est infini. Commoner *et al.* [CHEP71] ont établi une condition nécessaire et suffisante de vivacité pour une sous classe importante des réseaux de Petri : les graphes d'événements. Les graphes d'événements sont constitués de transitions et de places telles que chaque place possède au plus une transition en entrée et en sortie. Ce type de réseau de Petri est dit sans conflit car les jetons présents dans une place sont consommés par une unique place en aval. Il en résulte que cette classe de réseau de Petri ne permet pas de modéliser des problèmes dynamiques avec confilts sur les ressources. Une condition nécessaire et suffisante de vivacité pour un graphe d'événements est la présence d'au moins un jeton dans chaque circuit du graphe.

Dans [Lie76], Lien définit les réseaux de Petri généralisés (*Generalized Petri Nets*) comme étant des réseaux de Petri pouvant avoir des fonctions de marquage dans $\mathbb{N}^\star$. Il propose une étude importante sur les propriétés fondamentales de ces systèmes. En particulier, il caractérise les propriétés de conservation et de répétitivité d'un réseau de Petri généralisé qui décrivent des invariants du réseau. La propriété de conservation est une propriété d'invariance sur le nombre de jetons présents dans les places du réseau. La propriété de répétitivité montre qu'il existe un fonctionnement périodique du système. Lien montre que ces deux propriétés sont liées aux paramètres structurels du réseau et montre ainsi qu'il est possible de vérifier ces deux propriétés par le calcul. Dans sa thèse d'état [Chr83], Chrétienne s'intéresse aux réseaux de Petri temporisés. Dans ce modèle, le temps est pris en compte et permet de modéliser de nombreux problèmes d'ordonnancement. Chrétienne s'intéresse alors à la structure de l'ordonnancement au plus tôt des transitions d'un graphe d'événements temporisés. Chrétienne caractérise alors le fonctionnement optimal d'un tel système en analysant l'ordonnancement au plus tôt des transitions. Il montre que l'ordonnancement au plus tôt est K-périodique et propose un algorithme polynomial pour le calcul des fréquences des transitions. Avec ce même modèle, Ramamoorthy et Ho [RH80] montre parallèlement à Chrétienne qu'il existe un ordonnancement périodique qui atteint le débit maximum associé à un marquage initial donné. Ils proposent également un algorithme polynomial permettant de vérifier la valeur du débit des transitions d'un système.

*

Afin de proposer une alternative au modèle d'exécution séquentiel d'un programme, Karp et Miller [KM66] définissent les *computation graphs*. Dans ce modèle, un algorithme est décrit sous la forme d'un graphe orienté dont les sommets modélisent des étapes bien définies d'un programme informatique ou des sous-programmes. Les arcs représentent les échanges de données entre ces différentes étapes du programme. Lorsqu'une étape de l'algorithme s'achève, une quantité fixe de données est produite et stockée provisoirement dans des files d'attente représentées par les arcs du graphe. Ces données seront alors consommées utlérieurement lors des activations futures des sous-programmes concernés. La figure 2.1 représente un computation graph à deux sommets et un arc.

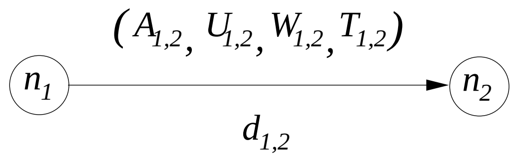


FIG. 2.1: Computation graph à deux sommets.

Le computation graph décrit sur cette figure contient deux sommets n_1 et n_2 reliés par un arc $d_{1,2}$. L'arc $d_{1,2}$ est valué par le quadruplet $(A_{1,2}, U_{1,2}, W_{1,2}, T_{1,2})$ tel que $A_{1,2}$ désigne le nombre de données présentes initialement dans cette file, $U_{1,2}$ (*resp.* $W_{1,2}$) désigne le nombre de données écrites (*resp.* le nombre de données lues ou consommées) par le sous-programme n_1 (*resp.* par n_2) dans la file $d_{1,2}$. Enfin, $T_{1,2} \geqslant W_{1,2}$ est le seuil de données nécessaires pour lancer une activation du sous-programme n_2. L'activation de chaque étape de l'algorithme est alors uniquement conditionnée par la présence d'une quantité suffisante de données sur tous les arcs situés en son entrée. Une particularité des computation graphs est qu'ils ne prennent pas en compte la sensibilité aux données des sous-parties du programme. La définition d'un programme informatique et de son fonctionnement dans ce modèle diffèrent de la définition classique pour laquelle les instructions d'un programme s'exécutent séquentiellement dans un ordre déterminé sous le contrôle d'un compteur ordinal. Cependant, les computation graphs représentent une façon naturelle de décrire certains programmes informatiques et permettent de mettre en évidence le parallélisme entre leurs différentes parties. Karp et Miller montrent alors que l'exécution d'un programme modélisé par un computation graph est déterministe et ce quelles que soient les durées des différentes sous-parties du programme. Ce déterminisme est un atout singulier par rapport aux autres modèles pour le calcul parallèle. Les auteurs s'intéressent également à l'existence d'un interblocage d'un computation graph. Ils montrent alors que tout computation graph a soit une exécution infinie (propriété de vivacité) soit une exécution finie. Karp et Miller montrent que le problème de l'existence d'un interblocage dans un computation graph appartient à NP (cependant le résultat n'est pas formulé ainsi car leur papier est antérieur aux résultats de Cook [Coo71] et de Karp [Kar72]). Ils proposent alors pour la vivacité une condition suffisante ainsi qu'une condition nécessaire portant sur la notion de gain associé aux circuits élémentaires du graphe (le gain $g(C)$ d'un circuit élémentaire C étant défini par $g(C) = \prod_{\forall d \in C} \frac{U_d}{W_d}$). Enfin, en s'appuyant sur cette notion de gain, les auteurs montrent que la taille des files est bornée si et seulement si le gain associé à chaque circuit est inférieur ou égal à 1. Cependant, ces travaux de Karp et Miller ont été entrepris dans l'optique de caractériser des programmes devant avoir une exécution finie. Dans [Rei68], Reiter s'intéresse à l'ordonnançabilité de programmes parallèles en utilisant le modèle des computation graphs. Il suppose alors que $U_d = W_d = T_d = 1$ pour tout arc d. De plus, ces arcs sont valués par un réel positif afin de modéliser la durée d'exécution du sous-programme situé en amont. Reiter montre qu'il existe pour tout computation graph une valeur dépendante des conditions initiales des files d'attente des circuits du graphe décrivant le fonctionnement asymptotique du système. Il montre également qu'un ordonnancement périodique atteignant ce débit peut être obtenu en résolvant un programme linéaire. Les travaux de Reiter constituent une première étape dans l'analyse des performances d'un programme modélisé par un computation graph.

*

L'article de Kahn [Kah74] définit un modèle pour la programmation basé sur une description graphique du programme. Dans ce modèle, on décrit un algorithme sous la forme d'un graphe orienté. Les sommets du graphe désignent les sous-parties de l'algorithme. Ces sous-parties sont amenées à s'échanger des données au cours de l'exécution de cet algorithme. Ces échanges sont réalisés par le biais de tampons pouvant stocker provisoirement ces données. Dans ce modèle, on considère l'écriture de données d'un processus sur un tampon comme non-bloquante. En revanche, la lecture de données d'un processus sur un tampon vide est bloquante ce qui entraîne la suspension du processus jusqu'à l'arrivée de données. La communication des différents processus d'un algorithme s'effectuant par des tampons, on emploie également le terme de réseau de processus de Kahn pour désigner ce modèle. La description d'un algorithme par un réseau de processus de Kahn est encore plus naturelle que dans le modèle des computation graphs de Karp et Miller. En outre, le fonctionnement d'un algorithme induit par la sémantique est déterministe. Cependant, la sémantique de ce modèle induit l'indécidabilité du problème de dimensionnement des tailles à allouer aux tampons. Pour pallier à cet inconvénient, on suppose que les tampons sont de tailles infinies. En conséquence, l'utilisation de ce modèle s'avère utile pour la description d'applications complexes mais la prédiction algorithmique des performances des systèmes modélisés est hors de portée.

2.2 Le modèle Data flow

Le modèle *dataflow* (*i.e.* flot de données) est un modèle inspiré des modèles des computation graphs et des réseaux de Kahn. Ce modèle s'appuie également sur une représentation graphique des programmes ou des architectures modélisés. Dans la définition originale du modèle [Den74, DK82], les sommets du graphe représentent les opérations élémentaires d'un programme informatique et les arcs entre deux sommets modélisent des dépendances de données entre ces deux opérations. On décrit également les conditions initiales du programme en disposant sur les arcs des jetons qui modélisent les données initiales du système. Contrairement au modèle réseau de Petri, ces jetons sont susceptibles de représenter une valeur de conditionnement. La sémantique du modèle défini par Dennis est simple. Pour qu'une opération (représentée par un sommet du graphe) puisse s'exécuter, il faut que des données soient disponibles sur toutes ses entrées. Lorsque l'opération est exécutée, celle-ci consomme un jeton sur chaque arc entrant et produit un jeton sur chaque arc sortant. La représentation dataflow est naturelle pour les programmes informatiques, les architectures multi-processeurs et la compilation parallèle. L'article de Najjar *et al.* [NGL99] dresse un état de l'art sur ce modèle et souligne son utilité actuelle. La réprésentation sous forme d'un graphe dataflow définissant seulement un ordre partiel pour l'exécution des instructions, celle-ci met en valeur le parallèlisme potentiel d'un programme ou d'une application. Cependant, le modèle dataflow se prête mal à la description d'application opérant sur des données à grosse granularité. L'analyse des dépendances de données est plus complexe et le graphe dataflow résultant est de taille conséquente.

*

Les systèmes informatiques et micro-électroniques devenant de plus en plus complexes, la modélisation d'une application par un graphe dataflow s'avérait fastidieuse. L'idée introduite par Lee et Messerschmitt [LM87b, LM87a] consiste à définir une application algorithmique comme un ensemble de blocs élémentaires communiquants interconnectés. La conception et la programmation de ces systèmes sont alors grandement simplifiées par une description basée sur une décomposition intuitive de l'application. Cette décomposition permet également de mettre en valeur le parallélisme potentiel de l'application. Par exemple, une application de traitement du signal met en œuvre des filtres et a recours à des processus calculatoires comme des calculs de FFT. Pour modéliser l'ensemble de l'application, la représentation synchronous dataflow définit les blocs élémentaires (filtres, calcul de FFT...) et identifie les échanges de données entres ces différentes parties de l'application. Il résulte de cette décomposition de l'application en blocs une granularité plus large des données échangées entre deux processus communiquants. De plus, cette description sous forme de blocs est hiérarchique et permet au concepteur de se situer à différents niveaux de granularité. Les blocs peuvent alors eux-mêmes représenter des graphes synchronous dataflow. Un bloc est dit synchrone lorsque les quantités de données produites et consommées par celui-ci sont constantes au cours du temps et connues à l'avance du concepteur. Une application est décrite dans le modèle synchronous dataflow sous la forme d'un graphe orienté dont les sommets modélisent les processus de cette application et dont les arcs relient deux processus devant s'échanger des données. Dans le cas d'une application physique, les sommets représentent des processus de l'application et les arcs modélisent les buffers qui permettent à ces deux processus de se communiquer des données. Chaque buffer est alors associé à un couple de processus afin de stocker provisoirement les données émises par un processus à destination d'un processus voisin. Les données présentes dans un buffer sont alors gérées comme dans une Fifo (*i.e.* first in, first out) et sont considérées comme homogènes en taille et en type. Les arcs du graphe synchronous dataflow sont valués par deux entiers. La figure 2.2 représente un graphe synchronous dataflow à deux blocs α et β.

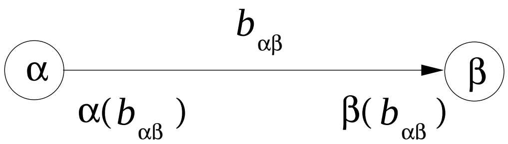


Fig. 2.2: Graphe synchronous dataflow à deux blocs.

Le bloc α communique des données au bloc β en écrivant des données dans le buffer $b_{\alpha\beta}$. Lorsque le bloc β est invoqué pour une exécution, il doit consommer $\beta(b_{\alpha\beta})$ données stockées dans le buffer $b_{\alpha\beta}$. Cette quantité de données doit être présente dans le buffer afin que le bloc β puisse être initié. De même, lorsque le bloc α termine son exécution, il écrit $\alpha(b_{\alpha\beta})$ données dans le buffer $b_{\alpha\beta}$ qui seront consommées ultérieurement par le bloc β. Une spécification importante des systèmes modélisés par un graphe synchronous dataflow est que la taille des buffers doit être finie.

Dans le contexte de l'électronique embarquée, les applications sont susceptibles de devoir fonctionner pendant une durée indéfinie. Afin de modéliser pleinement ce genre de systèmes, un graphe synchronous dataflow doit vérifier certaines propriétés :

1. Les blocs d'un graphe synchronous dataflow doivent pouvoir s'exécuter un nombre infini de fois. Cette caractéristique est particulièrement vraie pour des applications de traitement du signal.

2. Il existe un ordonnancement périodique valide pour lequel les tailles des buffers sont finies. Afin de faciliter la programmation de ces systèmes, les concepteurs définissent un ordonnancement périodique pour simplifier l'analyse et la vérification du comportement de l'application.

Dans [LM87a], Lee et Messerschmitt proposent une condition nécessaire pour la première des deux propriétés. Cette condition, appelée la consistance du graphe, est liée aux facteurs de lecture et d'écriture sur l'ensemble des buffers. Les facteurs d'écriture et de lecture étant constants, la tâche du concepteur consiste à générer un code informatique permettant de gérer les activations des processus et les échanges de données. Le concepteur définit également un ordonnancement des blocs sur une architecture mono ou multi-processeurs. L'application devant pouvoir s'exécuter infiniment longtemps, le concepteur construit alors un motif d'ordonnancement périodique des blocs tout en s'assurant que les tailles des buffers sont limitées.

Ces travaux sont à l'origine du *Ptolemy Project* initié par Lee et Messerschmitt au département d'informatique de l'université de Berkeley (*voir* http ://ptolemy.berkeley.edu/). Ce projet a pour objectifs de définir des modèles fiables décrivant des systèmes informatiques hétérogènes ainsi qu'une méthodologie pour la synthèse des systèmes embarqués. Le plate-forme java *PtolemyII* est un outil de conception assistée par ordinateur pour les systèmes embarqués. Depuis sa création, le *Ptolemy Project* s'est enrichi de nombreux modèles afin de s'adapter à une gamme d'applications de plus en plus large.

Une caractéristique majeure du modèle dataflow est qu'il modélise des systèmes déterministes. Cependant, ce modèle impose des restrictions importantes sur le comportement des blocs constitutifs d'une application. Par exemple, le modèle synchronous dataflow ne prend pas en compte les blocs pour lesquels le fonctionnement est dépendant des données lues sur ses buffers en amont. Dans sa thèse [Buc93], Buck définit les *Boolean dataflow graph* pour lesquels les blocs peuvent être sensibles à la valeur des données lues dans les buffers. En utilisant une réduction avec le *Halting problem* de Turing, Buck démontre l'indécidabilité du problème qui consiste à décider si un graphe booléen dataflow peut s'exécuter infiniment longtemps avec une taille finie pour l'ensemble des buffers.

Dans sa thèse [Bha94], Bhattacharyya étudie le problème du dimensionnement des mémoires pour les systèmes DSP (*i.e.* Digital Signal Processing). De nos jours, les DSP sont omniprésents dans les systèmes embarqués (avionique, automobile, baladeur mp3, téléphone mobile...). La modélisation synchronous dataflow est particulièrement bien adaptée à ce type d'application. Les travaux de Bhattacharyya ont été effectués dans le cadre du *Ptolemy Project* et sont imprégnés de la philosophie initiale du modèle synchronous dataflow. Lors de la conception d'un DSP, le concepteur définit un motif d'ordonnancement des blocs intervenant dans l'application, devant être répété infiniment souvent. Ce motif d'ordonnancement est stocké dans une mémoire particulière du DSP sous la forme d'un programme informatique définissant les activations de blocs les unes par rapport aux autres. Cependant la taille de cette mémoire est également limitée. Le concepteur doit alors judicieusement définir un motif d'ordonnancement de sorte que le programme qui le génère soit également de taille minimum. Dans [BL94, BBHL95, BML97], Bhattacharyaa expose des heuristiques efficaces pour la construction de motifs d'ordonnancements compacts qui minimisent la quantité de mémoire utilisée résultant du fonctionnement de l'application. Ces heuristiques reposent sur une décomposition du graphe synchronous dataflow et sur une analogie avec le problème de parenthésage d'un produit de matrices.

Dans sa thèse [Mur96], Murthy démontre que le problème étudié par Bhattacharrya est NP-complet. Etant donnée la complexité de ce problème de dimensionnement des buffers, Murthy *et al.* proposent une heuristique en $\mathcal{O}(n^3)$(où n désigne le nombre de sommets du graphe) pour le problème de dimensionnement des buffers d'un graphe synchronous dataflow acyclique. Murthy propose une borne inférieure de la taille à allouer à un buffer défini entre deux processus [MBL97]. Si nous considérons le buffer $b_{\alpha\beta}$ représenté sur la figure 2.2 page 12, la borne inférieure B_{inf} de la taille à allouer à ce buffer vaut :

$$B_{inf} \quad = \quad \alpha(b_{\alpha\beta}) + \beta(b_{\alpha\beta}) - pgcd(\alpha(b_{\alpha\beta}), \beta(b_{\alpha\beta})) \tag{2.1}$$

Murthy montre également que l'usage d'une mémoire partagée par tous les processus n'est pas forcément optimal. Si la gestion des écritures et des lectures sur cette mémoire est simplifiée par une utilisation judicieuse de pointeurs, la taille totale de cette mémoire partagée peut être sensiblement plus grande que la somme des tailles des buffers utilisés pour la même application mais avec un modèle synchronous dataflow (*i.e.* un buffer est associé à un arc du graphe). Dans [MB99], Murthy et Bhattacharyya exposent cependant une heuristique efficace qui consiste à fusionner certains buffers afin de minimiser la quantité totale de mémoires utilisées. Dans le cas où l'on considère que les mémoires ne peuvent être ni partagées ni fusionnées, Murthy prouve que le problème qui consiste à déterminer le taille minimum pour l'ensemble des buffers d'un graphe synchronous dataflow est un problème NP-complet.

*

Les problèmes d'optimisation traités au sein du Ptolemy Project sont donc fortement liés aux contraintes imposées par les DSP. Cependant, certains paramètres comme le temps d'exécution de chaque bloc n'est pas pris en compte. Le débit des ces applications n'est également pas considéré. En s'appuyant sur des techniques d'ordonnancement développés pour les compilateurs [DGN92], Govindarajan et Gao abordent le problème de la recherche d'un ordonnancement des blocs d'un graphe synchronous dataflow qui maximise le débit de l'application [GG93]. Les auteurs montrent que l'on peut associer au graphe synchronous dataflow un graphe dataflow homogène qui modélise l'ensemble des contraintes de précédence entre les différentes exécutions des processus du graphe initial. La recherche d'un ordonnancement de débit maximum est associé à un programme linéaire en nombre entiers dont les contraintes modélisent les contraintes de précédence du graphe dataflow homogène. Cependant la taille du graphe dataflow homogène étant pseudo-polynomiale par rapport à la taille de l'instance de départ, la résolution du programme linéaire est pseudo-polynomiale. Pour vérifier les performances attendues de ces applications, Ito et Parhi [IP95] proposent un algorithme exact du calcul du débit des processus d'un graphe synchronous dataflow. L'algorithme est basé sur le calcul du circuit de poids moyen minimum d'un graphe orienté pondéré. Pour cela, un graphe dataflow homogène est construit dans un premier temps selon la technique exposée dans [Par89]. Afin d'améliorer la complexité de l'algorithme, certains arcs redondants sont supprimés de ce graphe dataflow homogène. Cependant, la complexité de l'algorithme est fortement liée aux valeurs numériques de l'instance et s'avère donc être pseudo-polynomiale.

Govindarajan *et al.* [GGD94] ont alors étendu leurs méthodes afin de déterminer conjointement un ordonnancement qui soit de débit maximum pour l'application modélisée tout en minimisant la somme des tailles des buffers utilisés. Dans [ČP93], Čubrić et Panangaden exposent un algorithme permettant de déterminer un ordonnancement qui confère au système un débit maximum. Ils présentent également un algorithme qui minimise la somme des tailles de buffers utilisés. Contrairement aux recherches menées à l'université de Berkeley, les auteurs ne prennent pas en compte la taille du motif d'ordonnancement. Afin de concilier le critère du débit de l'application et de la quantité de mémoires utilisées, Govindarajan et Gao [GG95] ont analysé le problème de la construction d'un ordonnancement de débit maximum utilisant le moins de quantité de mémoire possible. En modélisant la durée de vie des données dans les buffers par un graphe d'intervalles cyclique, les auteurs montrent que l'on peut calculer une taille pour chaque buffer avec des algorithmes de coloration de graphes appropriés. Govindarajan *et al.* [GGD02] étendent cette méthode d'optimisation dans le cas d'applications utilisant une mémoire partagée. Dans ce modèle toutes les données échangées entre les différents processus transitent sur une mémoire partagée. De nouveau, en modélisant la durée de vie des données stockées dans la mémoire par un graphe d'intervalles cyclique, les auteurs proposent une méthode pour minimiser la taille de la mémoire utilisée. Cependant, du fait des transformations algorithmiques employées, les méthodes d'optimisation que nous avons présentées ne sont efficaces que pour des instances de petite taille.

$*$

Parallèlement aux travaux menés en Amérique du nord, des recherches sur les mêmes problèmes ont été entreprises dans divers pays européens. Dans [ALP94], Adé *et al.* proposent une heuristique pour le dimensionnement des buffers d'un DSP. Leur approche est différente de celle des universitaires de Berkeley. En effet, le problème d'optimisation traité dans la thèse d'Adé [Adé96] a pour objectif unique la minimisation de la somme des tailles de buffers utilisés. La recherche d'un motif ordonnancement de taille minimale et compatible avec les tailles définies pour les buffers n'est plus essentielle. Parallèlement à Murthy, Adé démontre la formule de la taille minimum qu'il faut allouer à un buffer (*voir* formule 2.1 page 14). Pour résoudre le problème de dimensionnement de buffers, Adé propose dans sa thèse une heuristique basée sur l'analyse de motifs particuliers de graphes synchronous dataflow. Dans [ALP96], Adé *et al.* montrent que le problème de la minimisation du dimensionnement des mémoires d'un graphe synchronous dataflow est un problème important lorsque l'application cible est réalisée sur FPGA (Field Programmable Gate Arrays). Pour ce problème, la ressource mémoire est très limitée et il convient de la minimiser au maximum. Contrairement à la problématique traitée par Bhattacharyya *et al.*, il n'est pas nécessaire de minimiser la taille du motif d'ordonnancement. Cependant, ces travaux ne prennent pas en compte des critères de performance comme le débit. Cette heuristique est implémentée dans la plateforme *Grape II* conçue pour la conception d'applications de traitement du signal [ALP95, LEAP95]. Lauwereins *et al.* [WELP95, EBLP95] ont complété la gamme des modèles de calcul en proposant une extension des graphes synchronous dataflow : les graphes cyclo-static dataflow. Ce modèle permet de modéliser des systèmes pour lesquels chaque bloc est susceptible d'adopter périodiquement un mode de fonctionnement particulier. Les facteurs de lecture et d'écriture associés à un couple « (processus,buffer) » sont multiples dans ce modèle et varient au cours du temps. Ce type de fonctionnement périodique des processus d'une application est difficilement modélisable par un graphe synchronous dataflow. Les auteurs définissent pour ce type de graphes une condition de consistance semblable à celle définie par Lee et Messerschmitt. Dans [PPL95], Parks *et al.* exposent une comparaison des modèles synchronous dataflow et cyclo-static dataflow. Le modèle cyclo-statique est alors enrichi par la définition des Cyclo Dynamic Dataflow [WELP96]. Ce modèle permet de modéliser des applications ayant des processus à fonctionnement périodique et également sensibles à la valeur des données.

Plus récemment, Geilen *et al.* [GBS05] ont étudié le problème de la minimisation de la somme des tailles des buffers utilisés pour une application embarquée fonctionnant sur une architecture parallèle. Dans ce contexte, le code définissant le motif d'ordonnancement périodique du système n'a pas à être stocké dans une mémoire. L'ordonnancement est construit de façon dynamique en appliquant une politique d'ordonnancement au plus tôt et par conséquent la taille du motif d'ordonnancement recherché n'est plus contrainte. Pour résoudre ce problème, Geilen *et al.* utilisent une méthode de modèle checking en décrivant la sémantique d'un graphe synchronous dataflow sous SPIN. Bien que cette approche semble fonctionnelle sur certain exemple, celle-ci souffre de la taille de l'espace d'état à explorer qui dépend des valeurs numériques du graphe dataflow. Dans [SGB06], ces mêmes auteurs développent leur démarche pour prendre en compte le problème bicritère qui consiste à minimiser la somme des tailles des buffers tout en garantissant un certain débit de l'application. Ces deux critères étant antagonistes, les auteurs proposent de construire algorithmiquement une courbe des optimums de Pareto. De nouveau, le succès de cette approche model checking est conditionné par une taille raisonnable de l'instance traitée. Le débit d'une application est un critère de performance important dans les applications temps réels. On pourra citer en exemple le cas de la télévision numérique haute définition pour lequel les images doivent être affichées à une certaine cadence afin de procurer un certain confort visuel pour l'utilisateur. Dans [GGS+06], les auteurs exposent une définition du débit global d'un graphe synchronous dataflow et proposent une méthode basée sur la simulation du graphe synchronous dataflow en explorant l'espace d'état des buffers. Ghamarian *et al.* [GGB+06] étudient le problème de la bornabilité des buffers d'un graphe synchonous dataflow ainsi que le problème de la vivacité et du débit. Ils proposent un algorithme qui étant donné un graphe synchronous dataflow soumis à une politique d'ordonnancement au plus tôt calcule le débit global de l'application et indique si les buffers du graphe peuvent être bornés. Les résultats explicités dans ce papier sont cependant tous des résultats classiques de la communauté réseau de Pétri, ce qui suggère un cloisonnement des communautés scientifiques. De plus, les algorithmes proposés dans ces deux articles ne sont pas de complexité polynomiale. Les instances pouvant être traitées par ces méthodes sont donc de taille limitée.

2.3 Le modèle réseau de Petri

Le modèle des graphes d'événements temporisés permet de modéliser naturellement de nombreux problèmes d'ordonnancement cyclique. Les transitions modélisent les tâches ou des ateliers d'une chaîne d'assemblage. Les places peuvent modéliser des contraintes de précédence entre des tâches ou alors des zones de stockage des produit en cours de fabrication sur la chaîne. Les jetons représentent les produits manufacturés en cours de fabrication sur la chaîne. Le débit des transitions est un critère de performance utile pour l'analyse et la conception d'une chaîne d'assemblage. Cependant, le nombre d'encours présents (modélisés par les jetons du marquage initial) initialement dans la chaîne d'assemblage doit être limité au maximum pour éviter des coûts économiques liés au stockage de ces produits. Le problème bi-critère qui consiste à déterminer un marquage initial de taille minimum (*i.e.* le nombre total d'encours présents au démarrage d'une chaîne d'assemblage) qui confère aux transitions du graphe un débit supérieur à une valeur donnée (*i.e.* une certain niveau de productivité de la chaîne) est un problème scientifique et industriel important. Hillion *et al.* [HP89] s'intéressent à la détermination d'un marquage initial de taille minimum qui confère aux transitions du graphe un débit maximum. Les auteurs montrent que ce problème peut être formulé comme un programme linéaire en nombre entiers et proposent une heuristique efficace pour sa résolution. Febbraro *et al.* [DFMS97] définissent un problème d'ateliers complexe qu'ils modélisent par un graphe d'événements temporisé. Le problème consiste alors à définir un marquage qui permette au système d'atteindre son débit maximum tout en utilisant le moins de jetons possible et en repectant une contrainte de marquage sur certains circuits. Les auteurs proposent alors une méthode arborescente *Branch and Bound* en résolvant des sous-problèmes relaxés. Giua *et al.* [GPS00, GPS02] proposent différentes approches (algorithme glouton, résolution de programmes linéaires mixtes) pour la résolution de ce problème en fonction de la structure des instances traitées. Dans [LPX92], Laftit *et al.* étudient un problème d'optimisation de marquage légèrement différent du problème bi-critère initial défini par

Hillion *et al.*. Le problème consiste alors à minimiser une combinaison linéaire du marquage initial (un invariant de place) tout en garantissant que le marquage résultant permet de définir un ordonnancement périodique d'un certain débit. Les auteurs élaborent alors une heuristique efficace et un algorithme exact pour le calcul du marquage initial. Proth *et al.* [PSX97] élaborent des bornes sur les marquages solutions du problème d'optimisation de marquage. Ils définissent alors un algorithme exact de type *Branch and Bound* capable de traiter des instances de taille industrielle.

$*$

En utilisant les propriétés du semi-corps $(\mathbb{R}, max, +)$, Cohen *et al.* parviennent à caractériser le comportement asymptotique des dates de début des transitions associées à l'ordonnancement au plus tôt d'un graphe d'événements temporisé. Cette caractérisation est obtenue en décrivant le graphe d'événements temporisé par un système d'équations de récurrence dans ce semi-corps. Ce modèle permet d'obtenir les résultats démontrés par Chrétienne et décrit une nouvelle méthode algorithmique pour la résolution de ce problème. En utilisant ce même formalisme, Gaubert [Gau90] étudie un problème d'optimisation bi-critère d'affectation de ressources modélisé par un graphe d'événements généralisé temporisé. Dans ce problème, certaines places du graphe sont initialement sans jeton et les autres ont des marquages fixes. L'objectif du problème est de définir un marquage de ces places non marquées qui confère aux transitions du graphe un débit au moins supérieur à une valeur cible tout en minimisant la somme des jetons ajoutés. L'auteur propose une heuristique efficace basée sur la relaxation d'un programme linéaire en nombres entiers. Bien que cette approche théorique semble peu naturelle pour traiter ces problèmes d'ordonnancement cyclique, celle-ci met en valeur la structure mathématique sous-jacente de ces problèmes. En outre, ce formalisme suggère des méthodes algorithmiques distinctes des méthodes classiques pour la résolution de ces problèmes.

$*$

Le modèle des graphes d'événements est cependant limité et ne permet pas de modéliser certains problèmes. Les graphes d'événements généralisés sont des graphes d'événements ayant des fonctions de marquage dans $\mathbb{N}^{\star}$. Une analyse des propriétés classiques de ces graphes (*i.e.* la vivacité, la bornabilité des marquages du graphe, la répétitivité et la conservation des jetons) est exposée dans [TCWCS92]. Teruel *et al.* proposent un algorithme afin de tester la vivacité d'un graphe d'événements généralisé marqué. Cependant, l'algorithme possède une complexité algorithmique dépendante des paramètres numériques du système initial. Les auteurs exhibent alors une condition suffisante de vivacité pour les circuits. Dans [CWR93], Chrzastowski-Wachtel et Raczunas prolongent les travaux de Teruel *et al.* [TCWCS92] sur la vivacité. Les auteurs s'intéressent en particulier à la détermination du plus petit marquage vivant pour un circuit. Leurs travaux établissent alors un lien fort entre la vivacité et le problème des équations diophantines de Frobenius. Dans [Mun93], Munier généralise les résultats obtenus par Chrétienne pour les graphes d'événements généralisés. Munier établit une condition nécessaire de vivacité portant sur les circuits élémentaires du graphe. Munier montre que tout graphe d'événements généralisé peut être associé à un graphe d'événements modélisant exactement les mêmes contraintes de précédence entre les différentes exécutions des transitions. Pour cela, elle définit une transformation : l'expansion. La vérification de la vivacité est alors possible sur le graphe d'événements généré par l'expansion. Cependant, l'expansion est une transformation qui dépend des valeurs numériques de l'instance de départ. Il en résulte que l'analyse de la vivacité et du débit des transitions dans l'ordonnancement au plus tôt est de complexité pseudo-polynomiale. Les méthodes suggérées par Munier sont donc efficaces pour des instances de tailles raisonnables. Chao *et al.* [CZW93] soulignent l'importance de l'évaluation du débit pour les systèmes modélisés par les graphes d'événements généralisés temporisés. Ils montrent alors qu'il est possible de déterminer le débit par la même méthode employée pour les graphes d'événements temporisés si le marquage initial vérifie une certaine condition. Cependant, cette méthode est d'une portée extrêmement limitée à cause de cette condition. Plus récemment, Sauer [Sau03] s'est intéressée au problème d'optimisation de marquage pour les graphes d'événements généralisés temporisés. Pour ce faire, Sauer s'appuie sur l'expansion des graphes d'événements généralisés. Afin de minimiser le nombre des encours sur chaque place, Sauer expose une méthode qui, étant donné un marquage, définit un marquage moindre sans toutefois modifier la structure des contraintes de précédence engendrée par le marquage initial. Sauer propose également une heuristique pour la résolution du problème d'optimisation de marquage. Toursi et Sauer [TS04] définissent une méthode de type *Branch and Bound* pour résoudre le problème d'optimisation de marquage. Toutefois, les méthodes algorithmiques exposées sont basées sur des méthodes de complexité pseudo-polynomiale. Il en résulte que l'efficacité de la résolution algorithmique est à nouveau dépendante de la taille de l'instance traitée.

Conclusion

De nombreux modèles ont été élaborés pour le calcul parallèle et distribué, permettant ainsi de décrire un large éventail d'applications informatiques et industrielles. Ces dernières années, l'importance du problème d'optimisation de la taille des buffers s'est accrue avec le développement fulgurant de l'électronique embarquée. De nombreux chercheurs appartenant à des communautés scientifiques distinctes ont étudié ce type de problèmes d'optimisation de ressources. Cependant, le succès des approches algorithmiques actuelles est limité par des considérations algorithmiques inhérentes à ces approches et aux modèles utilisés. En outre, l'analyse des problèmes d'ordonnancements cycliques bi-critère sous-jacents est délicate et peu d'auteurs s'y sont attaqués malgré leur importance industrielle actuelle. Pour relever ce défi technique, il convient de simplifier au maximum la définition des problèmes à étudier afin de définir de nouveaux outils de complexité algorithmique raisonnable.

Chapitre 3

Présentation du problème

Sommaire

Ce chapitre présente la problématique en détails. Par la suite, nous introduisons le modèle des graphes d'événements généralisé ainsi que les notations associées à ce formalisme. Nous présentons l'ensemble des concepts, définitions et théorèmes importants nécessaires à l'élaboration des nouveaux résultats de cette thèse. Enfin, nous évoquons dans la dernière section de ce chapitre les problèmes de complexité des méthodes utilisées pour la vérification algorithmique de ces propriétés.

3.1 Présentation du problème de dimensionnement des mémoires

Dans un premier temps, cette section présente de façon synthétique le modèle de fonctionnement d'une application embarquée. Ensuite, nous montrons que les situations critiques de fonctionnement de ces applications peuvent être assimilées aux situations de blocages. Enfin, nous énonçons clairement le problème du dimensionnement des mémoires.

3.1.1 Modèle de fonctionnement d'une application embarquée

Une application embarquée peut être vue comme un ensemble fini de processus communiquants asynchrones devant pouvoir s'exécuter sur une durée indéfiniment longue. Un processus est un programme, ne pouvant s'exécuter lui-même en parallèle, qui consomme et produit des données. On parle parfois de modèle producteur-consommateur.

Au cours du temps, les processus sont amenés à s'échanger des données. Du fait de l'asynchronisme, les données produites par un processus ne sont pas forcément consommées instantanément par un autre. L'usage de mémoires permet alors de stocker provisoirement les données produites pour être consommées ultérieurement par un autre processus. Une mémoire est définie entre deux processus telle qu'il existe un unique processus qui écrit des données dans cette mémoire et un unique processus les lisant. Cette configuration particulière mémoire-processus permet d'éviter tout conflit en lecture et en écriture. Un processus communique ses données à un processus voisin en effectuant des opérations d'écriture sur la mémoire définie entre eux. De même, ce processus voisin acquiert des données nécessaires au lancement d'une nouvelle exécution en effectuant une opération de lecture sur cette mémoire. Les données stockées dans une mémoire sont de type homogène. De plus, les opérations d'écriture et de lecture sur une mémoire sont séquentielles et les données présentes dans une mémoire sont gérées comme les éléments d'une FIFO (*i.e.* les données consommées par un processus dans la mémoire sont les données les plus anciennes stockées dans celle-ci). Le nombre maximum de données pouvant être stockées simultanément dans une mémoire est appelé la taille de la mémoire. Pour les applications réelles, la taille des mémoires doit être bornée donc finie.

Afin de lancer une nouvelle exécution, un processus doit consommer des données sur l'ensemble de ses mémoires situées en son entrée. Une exécution est lancée seulement s'il y a suffisamment de données présentes dans les mémoires concernées sinon l'exécution est retardée. Si ces conditions sont vérifiées, le processus lance alors une exécution après avoir lu les données nécessaires présentes dans les mémoires concernées. Une fois l'exécution finie, le processus produit une quantité fixe de données à destination d'autres processus et les écrit simultanément dans les mémoires dédiées à cet effet. Le nombre de données nécessaires pour initier l'exécution d'un processus est un paramètre statique propre à chaque port de ce processus. A l'issue de cette exécution, les données consommées sont effacées des mémoires d'entrée du processus exécuté.

L'exemple de la figure 3.1 récapitule les idées développées dans ce paragraphe. Soient deux processus communiquant P_1 et P_2 partageant la mémoire $M_{1,2}$. Chaque case vide représente un espace mémoire disponible. Une case est grisée lorsque celle-ci contient une donnée. On considère alors la séquence d'exécution P_2 puis P_1. La première figure représente l'état de la mémoire avant et pendant l'exécution de P_2. L'état de la mémoire immédiatement après l'exécution de P_2 est décrit sur la deuxième figure. La dernière figure montre l'état de la mémoire à l'issue de l'exécution du processus P_1.

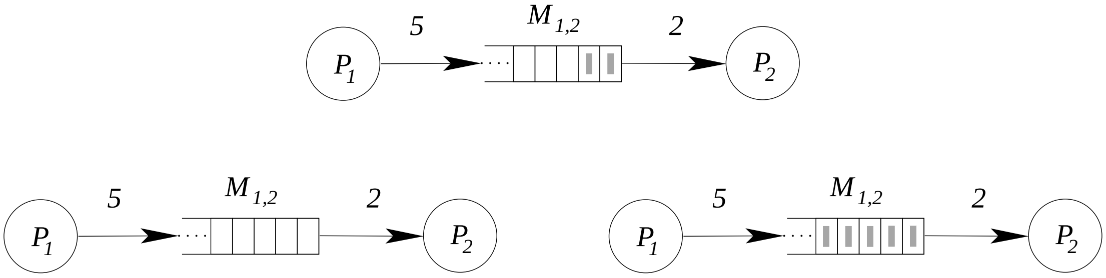


FIG. 3.1: Exemple de fonctionnement des écritures et des lectures de données sur une mémoire.

3.1.2 Blocage d'une application embarquée

Une application embarquée étant destinée à fonctionner sur une durée indéfinie, les processus mis en œuvre sont susceptibles d'être répétés infiniment souvent. Il convient donc d'éviter toute situation de blocage due aux lectures et écritures dans les mémoires. Dans un contexte embarqué, le non respect de cette exigence peut occasionner des problèmes allant de la simple erreur de calcul jusqu'à la perte physique du système. Compte tenu du contexte, ce type de comportement doit être proscrit. On distingue deux types de situation pouvant mener à un blocage complet du système :

- Un processus ne peut se terminer car il n'y a pas assez d'espaces disponibles dans l'une des mémoires situées en sa sortie. L'opération d'écriture est alors impossible et la tâche correspondante se met en attente. Dès que le problème sera résorbé par la consommation de données engendrée par l'exécution d'un processus voisin partageant cette mémoire, la tâche mise en attente reprendra son cours et procédera à l'écriture des données. Si, à un instant donné, il existe un ensemble de tâches bloquées formant un circuit, le système est dans une situation de blocage complet.

- Un processus ne peut pas être initié par manque de données sur l'une au moins des mémoires situées en son entrée. L'opération de lecture est impossible par manque de données sur la mémoire et par conséquent la tâche se met en attente. Dès que le problème sera résorbé par un apport suffisant de données sur les mémoires concernées, la tâche mise en attente reprendra son cours. De même, si, à un instant donné, il existe un ensemble de tâches bloquées formant un circuit, le système est dans une situation de blocage complet.

Les exemples de la figure 3.2 illustrent ces deux types de situations de blocage sur un système à trois processus. Ici les mémoires représentées sont finies. Le premier système est complètement bloqué car les mémoires $M_{1,2}$ et $M_{2,1}$ ont un trop plein de données qui empêchent les processus P_1 et P_2 de s'exécuter à nouveau. Le second système est dans une situation de blocage complet car aucun processus ne peut être initié à cause d'un manque de données dans les mémoires.

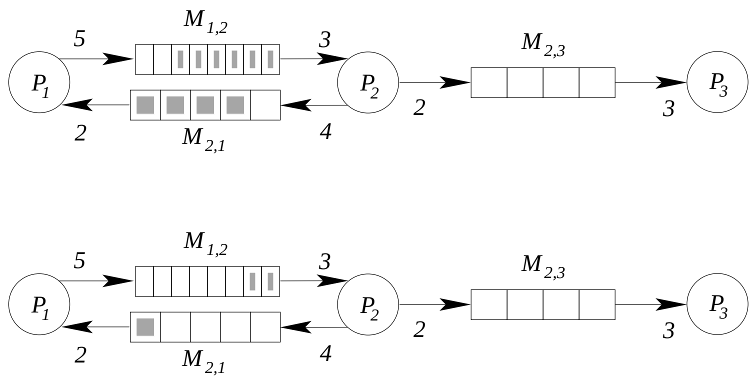


FIG. 3.2: Les deux types de blocage complet d'une application : trop plein de données et famine.

Le premier type de blocage dépend uniquement de la taille totale des mémoires qui a été initialement allouée. Le second type de blocage dépend lui de la configuration initiale des données dans les mémoires au moment du démarrage de l'application. Bien que étranger au problème de la taille des mémoires, l'analyse de ce dernier cas de figure s'avère décisive lors de la conception d'un système embarqué.

3.1.3 Le problème de dimensionnement des mémoires

Le problème de dimensionnement des mémoires d'un système embarqué consiste à déterminer une taille pour chacune des mémoires utilisées qui minimise la somme des tailles mémoires utilisées tout en garantissant l'absence de blocages.

Nous définissons formellement ce problème d'optimisation :

- Soit $P = \{P_1, P_2, \ldots, P_n\}$ un ensemble de n processus asynchrones communiquants.
- Soit $M = \{M_1, M_2, \ldots, M_m\}$ un sous-ensemble ordonné de $P \times P$ contenant m éléments appelés mémoires.
- Soit φ_i avec $i \in \{1, 2, \ldots, m\}$, la taille élémentaire d'une donnée stockée par la mémoire M_i. Suivant le contexte, le vecteur φ peut désigner également une fonction de coût croissante associée aux mémoires.

Déterminer une fonction $f : M \longrightarrow \mathbb{N}^\star$ qui associe à chaque mémoire de M une taille telle que la taille totale des mémoires utilisées, définie par :

$$\sum_{i=1}^{m} \varphi_i \cdot f(M_i)$$

soit minimum tout en garantissant l'absence de blocage de l'application.

Ce problème est pertinent à double titre. En effet, la surface occupée par une mémoire sur le silicium est proportionnelle à sa taille. Le coût du silicium et de la gravure sur silicium étant particulièrement onéreux, il convient alors de minimiser la somme des tailles des mémoires utilisées par l'application. De plus, la consommation électrique d'une mémoire est fonction croissante de sa taille ce qui pose problème dans un contexte embarqué où les ressources électriques sont limitées.

Par ailleurs, les méthodes qui ont cours dans l'industrie consistent peu ou prou à surdimensionner la taille des mémoires, faisant par la suite un usage intensif de simulations pour la vérification des spécifications du système embarqué. Ces systèmes sont de plus en plus complexes et les exigences techniques imposées par le marché sont de plus en plus fortes. Désormais, les concepteurs doivent faire face à des problèmes d'optimisation bi-critère comme la maximisation du débit d'une application tout en minimisant la taille des mémoires utilisées. Nous traiterons ce type de problèmes dans la deuxième partie de cette thèse.

3.2 Choix du formalisme

Dans la communauté informatique, de nombreux langages de description de systèmes complexes ont été mis au point. Certains de ces langages sont spécifiques à un cadre d'applications bien déterminé. L'approche *dataflow* permet de modéliser par un formalisme simple les échanges de données qui ont lieu entre les différents processus mis en œuvre au sein d'un système complexe. Dans le modèle dataflow, les données échangées sont atomiques. En revanche, dans le modèle des Synchronous Dataflow (SDF), mis au point par Lee et Messerschmitt à la fin des années 80 [LM87a], la granularité des données échangées est supposée quelconque. De plus, les paramètres de chaque processus utilisé dans une application sont statiques et connus du concepteur. Ce paradigme introduit par sa simplicité davantage de souplesse dans la modélisation et la conception des systèmes embarqués.

Cependant, cette approche est restée circonscrite à ce domaine d'applications. Les *réseaux de Petri* [Pet62] ont eux connu un plus large succès que le modèle dataflow. Aussi, nous faisons le choix d'utiliser dans cette thèse le formalisme des réseaux de Petri. La classe des graphes d'événements généralisés (GEG) est un sous-ensemble des réseaux de Petri pour laquelle chaque place a en entrée et en sortie au plus une transition. De ce fait, il n'y pas de conflit possible pour les franchissements de transitions et le stockage des jetons sur les places. Comme dans le modèle SDF, le franchissement d'une transition est conditionné par la présence ou l'absence d'une quantité suffisante de jetons dans ses places d'entrée. La sémantique de ces deux modèles est donc la même à ceci près que les jetons d'une place d'un réseau de Petri ne sont pas supposés être gérés comme les éléments d'une FIFO. Le modèle des GEG est en quelque sorte l'homologue du modèle SDF. D'autre part, dans le modèle SDF comme dans le modèle GEG, la taille d'une mémoire ou d'une place est supposée infinie.

Les graphes d'événements généralisés sont bien adaptés à la modélisation de ces systèmes :

- Les places représentent les mémoires ;
- les jetons représentent les données.

Le choix de ce modèle nous semble plus judicieux car il est transversal à plusieurs communautés donc plus standard que le modèle SDF. Il est aussi largement éprouvé et la variété des outils mis au point pour l'analyse des problèmes de décision ou d'optimisation surpasse de loin celle des SDF.

3.3 Notations et définitions de base

Dans cette section, nous introduisons les définitions formelles liées au modèle des graphes d'événements généralisés ainsi que des notations mathématiques standards. Nous rappelons ensuite quelques résultats structurels importants issus de la littérature. Enfin, nous présentons quelques uns des principaux résultats en ordonnancement cyclique.

3.3.1 Présentation du modèle

Nous exposons ici dans un premier temps les définitions sur les graphes d'événements.

Un réseau de Petri marqué est un tuple (P, T, F, M_0, W) où

- $P = \{p_1, p_2, \ldots, p_m\}$ est l'ensemble des places. Son cardinal est noté $|P| = m$.
- $T = \{t_1, t_2, \ldots, t_n\}$ est l'ensemble des transitions. Son cardinal est noté $|T| = n$.
- F est un ensemble d'arcs tel que :

$$F \subseteq (P \times T) \cup (T \times P)$$

  i.e. un arc ne peut pas être connecté entre 2 places ou 2 transitions.
- $M_0 : P \to \mathbb{N}$ est un vecteur à $|P|$ composantes entières qui définit le marquage initial de l'ensemble des places.
- $W : F \to \mathbb{N}^\star$ est une fonction de poids définie pour chaque arc de F qui représente soit la quantité de jetons consommés par une transition pour être franchie (ou tirée) soit la quantité de jetons produits sur les places de sortie d'une transition après son franchissement (ou tir).

Pour toute place $p \in P$, on définit l'ensemble des prédécesseurs noté $\mathcal{T}^-(p)$ et l'ensemble des successeurs noté $\mathcal{T}^+(p)$ de la place p comme suit :

$$\mathcal{T}^-(p) = \{t \in T, (t, p) \in F\} \quad \text{et} \quad \mathcal{T}^+(p) = \{t \in T, (p, t) \in F\}$$

On peut définir de façon analogue l'ensemble des prédécesseurs noté $\mathcal{P}^-(t)$ et l'ensemble des successeurs noté $\mathcal{P}^+(t)$ d'une transition $t \in T$:

$$\mathcal{P}^-(t) = \{p \in P, (p, t) \in F\} \quad \text{et} \quad \mathcal{P}^+(t) = \{p \in P, (t, p) \in F\}$$

Définition 1. *Un réseau de Petri est un graphe d'événements généralisé (noté GEG) si pour toute place $p \in P$:*

$$|\mathcal{T}^+(p)| = |\mathcal{T}^-(p)| = 1$$

Une place p étant définie entre un couple (t, t') de transitions, on notera $p = (t, t')$. De plus, la définition des arcs de F étant supposée sans ambiguïté, un graphe d'événements généralisé marqué sera désigné par $G = (T, P, M_0)$.

Nous pouvons aussi définir les graphes d'événements qui sont une sous-classe des graphes d'événements généralisés :

Définition 2. *Un graphe d'événements généralisé dont la fonction de poids W vaut 1 pour tout arc est appelé graphe d'événements (noté GE).*

Les graphes d'événements généralisés étant des réseaux de Petri, ils admettent une représentation graphique classique définie ainsi :

- Une transition $t \in T$ est représentée par un rectangle.
- Une place $p \in P$ est représentée par un cercle.

- Le marquage $M_0(p)$ initial d'une place p est usuellement représenté par la présence de $M_0(p)$ jetons sur la place p. Par souci de lisibilité des figures, nous n'adopterons pas cette représentation standard d'un marquage et nous écrirons la valeur de $M_0(p)$ dans le cercle représentant la place p.
- Soit une place $p = (t_i, t_j)$. La transition t_i relie p par un arc (t_i, p) valué par l'entier positif $w(p)$. De même la place p relie la transition t_j par un arc (p, t_j) valué par l'entier positif $v(p)$. Lorsque G sera un graphe d'événements (*i.e.* non-généralisé), la valuation des arcs ne sera pas représentée.

La figure 3.3 ci-dessous décrit un exemple de représentation graphique pour un graphe constitué d'un couple de transitions (t_i, t_j) et d'une place $p = (t_i, t_j)$.

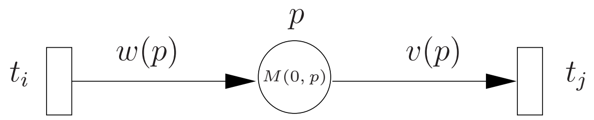


FIG. 3.3: Représentation graphique d'une place $p = (t_i, t_j)$ d'un graphe d'événements généralisé.

Les définitions suivantes sont des définitions générales et structurelles. Elles s'appliquent aussi bien aux graphes d'événements généralisés qu'aux graphes d'événements.
Un chemin μ est défini comme une séquence de n places telles que :

$$\mu = \{\, p_1 = (t_1, t_2), p_2 = (t_2, t_3), \ldots, p_n = (t_n, t_{n+1}) \,\}$$

Si $t_1 = t_{n+1}$ alors ce chemin est un circuit. De même, si toutes les transitions du circuit sont distinctes, le circuit est élémentaire. La notion classique de connexité de la théorie des graphes est, elle aussi, applicable aux réseaux de Petri. On dira qu'un graphe d'événements est fortement connexe si pour tout couple de transitions (t_i, t_j), il existe au moins un chemin allant de t_i à t_j.
Le poids d'un chemin μ (aussi appelé le gain) noté $W(\mu)$ est définit par le produit des rapports des fonctions d'écriture et de lecture des places de ce chemin, soit :

$$W(\mu) = \prod_{p \in P \cap \mu} \frac{w(p)}{v(p)}$$

On dira qu'un chemin μ est de poids unitaire si $W(\mu) = 1$. De même, un circuit C est unitaire si $W(C) = 1$. Un graphe d'événements généralisé fortement connexe est unitaire si tout ses circuits sont de poids 1.

3.3.2 Sémantique d'un GEG

Une transition t est franchissable pour un marquage M si :

$$M(p) \geqslant v(p) \quad \forall p \in \mathcal{P}^-(t)$$

Si la transition t est franchie (ou tirée), on note :

$$M \xrightarrow{t} M'$$

Le marquage M' résultant du franchissement de cette transition t vaut alors :

$$M'(p) = \begin{cases} M(p) - v(p) & \text{si } p \in \mathcal{P}^-(t) \\ M(p) + w(p) & \text{si } p \in \mathcal{P}^+(t) \\ M(p) & \text{sinon} \end{cases}$$

Si $M \xrightarrow{t_a} M'$ et $M' \xrightarrow{t_b} M''$, alors la séquence de tirs $\sigma = t_a t_b$ est valide à partir du marquage M et on note :

$$M \xrightarrow{t_a t_b} M''$$

3.3.3 Ecriture matricielle d'un GEG

Etant donnée une place $p = (t_i, t_j)$, les valeurs $w(p)$ et $v(p)$ sont respectivement appelées fonction d'écriture et de lecture de p. Par souci de clarté dans les notations, nous posons pour toute place $p \in P$:

1. Soit $pgcd(w(p), v(p)) = pgcd_p$ le plus grand commun diviseur de $w(p)$ et $v(p)$.
2. Soit $ppcm(w(p), v(p)) = ppcm_p$ le plus petit commun multiple de $w(p)$ et $v(p)$.

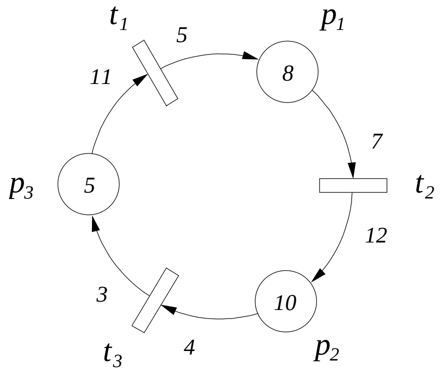
La matrice d'incidence d'un graphe d'événements généralisé (ou matrice d'incidence Pré-Post) permet une description synthétique d'un système. Elle est en quelque sorte l'analogue de la matrice d'incidence arcs-sommets en théorie des graphes. La matrice d'incidence d'un graphe d'événements généralisé G notée $\Gamma_G \in \mathcal{M}_{|P|\times|T|}$ est définie $\forall (p,t) \in P \times T$ comme suit :

$$\Gamma_G[p,t] = \begin{cases} +w(p) & \text{si} & p \in \mathcal{P}^+(t) \\ -v(p) & \text{si} & p \in \mathcal{P}^-(t) \\ 0 & \text{sinon} & \end{cases}$$

La figure 3.4 page suivante décrit un exemple d'un graphe d'événements à trois transitions avec l'écriture de sa matrice d'incidence.

Remarque 1. *Si un graphe d'événements possède une place $p = (t,t)$, alors la p-ième ligne de la matrice d'incidence du graphe ne comporte que des valeurs nulles. Cette ligne peut donc être supprimée. Nous verrons au chapitre 4 que nous pourrons faire de même pour le cas généralisé.*

Lorsqu'un graphe d'événements généralisé a une place $p = (t,t)$ marquée avec un jeton, on dit que celle-ci modélise une contrainte de non-réentrance sur t (*cf.* exemple de la figure 3.5 page suivante). Une contrainte de non-réentrance sur une transition t signifie que ses différentes occurrences doivent être franchies séquentiellement. Ce type de contrainte se révèle utile dans un contexte dit temporisé (*voir* section 3.5).



$$\Gamma_G = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{pmatrix} 5 & -7 & 0 \\ 0 & 12 & -4 \\ -11 & 0 & 3 \end{pmatrix} \end{array}$$

FIG. 3.4: Un graphe d'événements généralisé et sa matrice d'incidence.

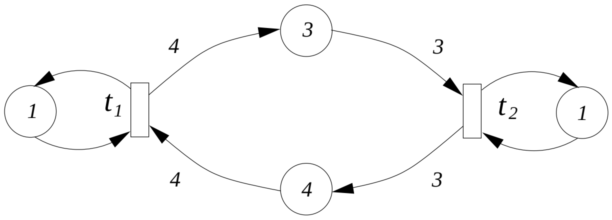


FIG. 3.5: Un graphe d'événements généralisé avec contraintes de réentrance sur ses transitions.

Convention : *Par la suite, les instances considérées auront toutes des contraintes de non-réentrance sur chacune de leurs transitions. Cependant, les contraintes de non-réentrance ne seront plus représentées afin d'alléger les figures.*

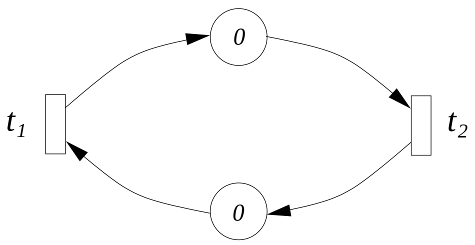
Soit σ une séquence de tirs. On note $\overline{\sigma}$ le vecteur dont les composantes $\overline{\sigma}(t)$ sont les nombres d'occurrences des transitions t dans la séquence de tirs σ. Ce vecteur est appelé vecteur caractéristique de la séquence de tirs σ et sa dimension est égale au nombre de transitions du graphe d'événements généralisé.
La variation du marquage d'un graphe d'événements généralisé G consécutive au franchissement d'une séquence de tirs σ telle que $M \xrightarrow{\sigma} M'$ peut être décrite mathématiquement à l'aide de son vecteur caractéristique et de la matrice d'incidence de G :

$$M' = M + \Gamma_G \cdot \overline{\sigma}$$

Cette équation, associée au couple de marquages (M, M'), est appelée équation fondamentale du graphe d'événements G.

Remarque 2. *L'existence d'un vecteur $\overline{\sigma}$ à coordonnées entières reliant un couple de marquages (M, M') par l'équation fondamentale n'implique pas l'existence d'une séquence de tirs valide reliant M à M'.*

En effet, observons l'exemple de la figure 3.6 page ci-contre.



$$\Gamma_G = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

FIG. 3.6: Un graphe d'événements et sa matrice d'incidence.

Si on considère $M = M' = 0$ alors le vecteur $(1,1)$ vérifie l'équation fondamentale de ce graphe d'événements. Cependant, aucune transition n'est franchissable donc il n'existe aucune séquence de tirs valide de vecteur caractéristique $(1,1)$.

3.4 Propriétés structurelles et comportementales

Cette section est consacrée à la présentation de propriétés essentielles des graphes d'événements généralisés. Ces propriétés sont de deux natures. Les propriétés de nature structurelle ne dépendent que de la structure du système modélisé alors que les propriétés comportementales dépendent également de l'état initial du système. Les propriétés structurelles permettent par exemple de détecter la non consistance d'un système. Les critères qualitatifs comme le débit d'un système sont dans une large mesure dépendants des propriétés comportementales. L'étude de ces propriétés est une étape capitale dans la conception de nombreux systèmes industriels.

Dans un premier temps, nous présentons les principales propriétés structurelles et comportementales d'un graphe d'événements généralisé. Les définitions d'invariants de place et de transition sont exposées par la suite. Enfin, après avoir rappelé quelques résultats fondamentaux sur l'analyse des graphes d'événements, nous présentons un outil pour l'analyse des propriétés comportementales d'un graphe d'événements généralisé.

3.4.1 Propriétés des GEG : vivacité et graphes K-bornés

Lorsqu'une transition t dans un graphe d'événements généralisé peut être tirée une infinité de fois, on dira que celle-ci est vivante. De même, si toutes les transitions du graphe d'événements généralisé sont vivantes, on dira que le graphe est vivant. On parlera de propriété de vivacité du système. En revanche, si pour toutes places $t \in T$ on a :

$$M(p) < v(p) \quad \forall p \in \mathcal{P}^-(t)$$

alors aucun franchissement de transition n'est possible. On parlera de situation de blocage du système.

Propriété 1. *Etant donné un marquage initial, un graphe d'événements généralisé fortement connexe est soit vivant soit toutes les séquences de tirs valides conduisent à une situation de blocage.*

Commoner *et al.* [CHEP71] prouvent le résultat suivant sur la vivacité des graphes d'événements :

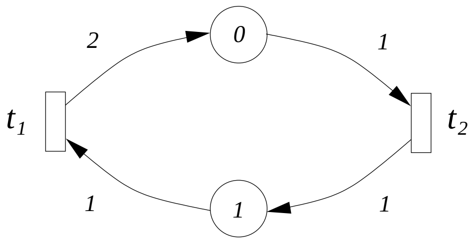
Théorème 1. *Soit G un graphe d'événements. G est vivant si et seulement si tout les circuits de G ont au moins un jeton.*

Pour tester la vivacité d'un graphe, il suffit alors de supprimer les places marquées et de tester si le graphe résultant est acyclique. Ce faisant, la vivacité des graphes d'événements est décidable en temps polynomial.
Dans [Mun93, TCWCS92], les auteurs expriment une condition nécessaire de vivacité d'un graphe d'événements généralisé en fonction du poids de ses circuits.

Propriété 2. *Soit G un graphe d'événements généralisé. Si G est vivant alors le poids de tout circuit de G est supérieur ou égal à* 1.

Cette condition n'est pas suffisante. En effet, dans le cas d'un graphe d'événements dont le marquage initial est nul sur toutes les places (*voir* l'exemple 3.6 page précédente), la condition nécessaire est vérifiée et cependant aucune transition n'est franchissable. On notera également que le circuit C de la figure 3.4 page 30 est de poids $W(C) = \frac{45}{77} < 1$. Donc, quel que soit le marquage initial de ce circuit, le système n'est pas vivant.

En pratique, les jetons représentent des données échangées dans un système informatique ou des produits dans un système manufacturier. Les places modélisent alors des zones de stockage. On cherche à s'assurer que le nombre de données ou de produits stockés dans un système soit fini, et ce quel que soit son fonctionnement. Considérons l'exemple de la figure 3.7 ci-dessous.



$$\Gamma_G = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$$

FIG. 3.7: Un graphe d'événements et sa matrice d'incidence.

Il est facile de voir que toute séquence valide de tirs de taille infinie contient une infinité de franchissements de t_1. Or lorsque la transition t_1 est franchie, le nombre total de jetons présents dans ce système augmente d'une unité. En revanche, le nombre total de jetons présents dans les places reste inchangé quand la transition t_2 est tirée. Le marquage total de ce graphe tend vers l'infini quelle que soit la séquence de tirs valide considérée. L'étude de cette propriété pour les réseaux de Petri s'avère importante pour la modélisation de systèmes réels.

Définition 3. *Un réseau de Petri est K-borné si quels que soient le marquage initial M et la séquence de tirs valide σ telle que $M \xrightarrow{\sigma} M'$:*

$$\exists K \in \mathbb{N}^m \quad \textit{tel que} \quad M' \leqslant K$$

Une caractérisation de cette propriété peut être obtenue par l'étude du poids des circuits d'un réseau :

Propriété 3. *Un graphe d'événements généralisé fortement connexe est K-borné si et seulement si le poids de tout ses circuits est inférieur ou égal à 1.*

Les propriétés 2 et 3 s'interprètent intuitivement de la manière suivante : le poids d'un circuit peut être vu comme un taux d'accroissement des jetons sur le circuit. Si le poids est strictement supérieur à 1, le nombre de jetons tend vers l'infini lorsque la taille de la séquence tend vers l'infini. De même, si le poids du circuit vaut 1, il y a en quelque sorte conservation des jetons dans le circuit. En revanche, si le poids est strictement inférieur à 1, il y a une décroissance progressive du nombre de jetons menant systématiquement à une situation de blocage. Seuls les deux premiers cas sont propices à l'existence d'une séquence de tirs infinie et seuls les deux derniers cas permettent d'avoir un marquage total du réseau fini.
Pour vérifier qu'un graphe d'événements généralisé possède bien ces deux dernières propriétés (*i.e.* existence d'une séquence de tir infinie et marquage total du réseau borné), nous disposons de l'algorithme polynomial conçu par Munier [Mun93]. Cet algorithme permettra d'écarter les instances ne vérifiant pas l'une ou l'autre de ces deux propriétés.

3.4.2 Invariants de place et de transition

Une notion utile pour l'étude des réseaux de Petri en général est l'étude des invariants du réseau.

Définition 4. *Soit G un réseau de Petri de matrice d'incidence Γ_G :*

1. *On appelle composante conservative ou encore P-semiflot tout vecteur Y à $|P|$ composantes tel que :*

$$^{\intercal}Y \cdot \Gamma_G = 0$$

pour toute séquence de franchissements valide σ telle que :

$$M \xrightarrow{\sigma} M'$$

selon l'équation fondamentale, nous avons alors :

$$\begin{array}{rcl} M' & = & M + \Gamma_G \cdot \overline{\sigma} \\ ^{\intercal}Y \cdot M' & = & ^{\intercal}Y \cdot M + ^{\intercal}Y \cdot \Gamma_G \cdot \overline{\sigma} \\ ^{\intercal}Y \cdot M' & = & ^{\intercal}Y \cdot M \end{array}$$

La quantité totale de jetons du réseau est constante selon cette combinaison linéaire. Le vecteur Y est un invariant de place.

2. *On appelle composante répétitive ou encore T-semiflot, tout vecteur X à $|T|$ composantes tel que :*
$$\Gamma_G \cdot X = 0$$
Selon l'équation fondamentale, si X est de plus le vecteur caractéristique d'une séquence de tirs valide alors celle-ci peut être répétée infiniment souvent conférant ainsi un fonctionnement cyclique à ce réseau. Le vecteur X est un invariant de transition.

On considère généralement les vecteurs semiflots de support minimal. Ces invariants décrivent à la fois la structure et le comportement d'un réseau. Munier [Mun93] propose un algorithme polynomial de calcul du T-semiflot de support minimal dans le cas des graphes d'événements généralisés unitaires. De plus, ces invariants peuvent être considérés comme des facteurs de pondération pertinents pour les fonctions objectifs dans un contexte d'optimisation. Pour les problèmes d'ateliers où intervient la minimisation des encours, il est d'usage de considérer comme fonction objectif le produit scalaire d'un marquage donné et d'un P-semiflot (*cf.* [LPX92, Sau03]).

3.4.3 Notion de contrainte de précédence

La sémantique des graphes d'événements généralisés indique que le franchissement d'une transition est conditionné par la présence d'une quantité suffisante de jetons dans ses places d'entrée. Une place $p = (t_i, t_j)$ et son marquage initial $M_0(p)$ induisent donc des contraintes entre les différentes exécutions de t_i et t_j.

Explicitons au préalable la notion de marquage instantané pour une place $p = (t_i, t_j)$ ayant $M_0(p)$ jetons initialement.

Notation : *Soient t une transition et ν un entier strictement positif. Le ν-ième franchissement de la transition t est noté $< t, \nu >$.*

Après ν_i franchissements de t_i, $\nu_i \cdot w(p)$ jetons sont ajoutés sur p. De même après ν_j initiations de la transition t_j, $\nu_j \cdot v(p)$ jetons sont retirés de la place p. Le marquage instantané résultant de ces franchissements de t_i et t_j vaut :

$$M(p) = M_0(p) + \nu_i \cdot w(p) - \nu_j \cdot v(p)$$

On dira qu'il existe une *contrainte de précédence* entre $< t_i, \nu_i >$ et $< t_j, \nu_j >$ si le ν_j-ième franchissement de la transition t_j ne peut avoir lieu qu'après le ν_i-ième franchissement de la transition t_i.

Dans le cas d'un graphe d'événements, Chrétienne [Chr83] note qu'une place $p = (t_i, t_j)$ avec un marquage initial $M_0(p)$ définit une contrainte de précédence entre $< t_i, k >$ et $< t_j, k + M_0(p) >$ quel que soit l'entier naturel k. Dans ce cas, le marquage d'une place peut être interprété comme un décalage potentiel maximal entre les différentes occurrences des transitions t_i et t_j.

Dans le cas généralisé, Munier [Mun93] montre qu'une place et son marquage initial modélisent une famille de contraintes de précédence entre les différents franchissements des transitions adjacentes. Plus formellement, une place $p = (t_i, t_j)$ de marquage initial $M_0(p)$ définit une contrainte de précédence entre $< t_i, \nu_i >$ et $< t_j, \nu_j >$ si et seulement si les deux conditions suivantes sont vérifiées :

1. après le ν_i-ième franchissement de t_i, il est possible de franchir t_j pour la ν_j-ième fois. Cette condition est équivalente à :

$$M_0(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geqslant 0$$

2. avant le ν_i-ième franchissement de t_i, on peut effectuer le $\nu_j - 1$-ième franchissement de t_j mais pas le ν_j-ième. Cette condition est équivalente à :

$$v(p) > M_0(p) + w(p) \cdot (\nu_i - 1) - v(p) \cdot (\nu_j - 1) \geqslant 0$$

En combinant ces deux inégalités, Munier [Mun93] aboutit au résultat suivant :

Lemme 1. *Soient une place $p = (t_i, t_j)$ de marquage initial $M_0(p)$ et deux entiers ν_i et ν_j. Il existe une contrainte de précédence entre $< t_i, \nu_i >$ et $< t_j, \nu_j >$ si et seulement si :*

$$w(p) > M_0(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geqslant \max(w(p) - v(p), 0)$$

Ce résultat simple met donc en relation les différentes occurrences de franchissement des transitions t_i et t_j. En revanche, le lien entre les différentes occurrences des transitions t_i et t_j apparaît à présent beaucoup moins simple. Afin de faciliter l'analyse des graphes d'événements généralisés, Munier propose un outil puissant : l'expansion.

3.4.4 L'expansion : un outil pour l'étude des GEG

Dans [Mun93], Munier montre qu'à tout graphe d'événements généralisé correspond un graphe d'événements modélisant les mêmes contraintes de précédence. Pour cela, Munier définit une transformation qui associe à chaque place marquée $p = (t_i, t_j)$ du graphe d'événements généralisé un graphe d'événements. Cette transformation est appelée expansion.

L'idée sous-jacente consiste à observer que l'ensemble infini des contraintes de précédence induites par une place p et son marquage initial possède une structure répétitive finie. Cette structure est largement dépendante des valeurs du vecteur de sa composante répétitive. En effet, soient $p = (t_i, t_j)$ une place de marquage initial $M_0(p)$ et (N_i, N_j) un T-semiflot de p. S'il existe une contrainte de précédence entre $< t_i, \nu_i >$ et $< t_j, \nu_j >$ alors :

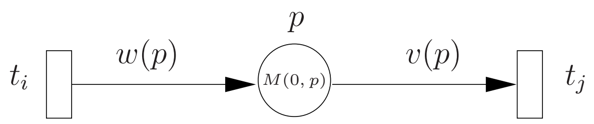
$$w(p) > M_0(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geqslant \max(w(p) - v(p), 0)$$

Par ailleurs, comme (N_i, N_j) est un T-semiflot (*i.e.* $N_i \cdot w(p) - N_j \cdot v(p) = 0$), on a :

$$w(p) > M_0(p) + w(p) \cdot (\nu_i + N_i) - v(p) \cdot (\nu_j + N_j) \geqslant \max(w(p) - v(p), 0)$$

Cette dernière inégalité implique qu'il existe également une contrainte de précédence entre $< t_i, \nu_i + N_i >$ et $< t_j, \nu_j + N_j >$. On explicite alors par le calcul une famille de contraintes de précédence qui génère cet ensemble infini de contraintes.

Plus formellement, soit $p = (t_i, t_j)$ une place avec initialement $M_0(p)$ jetons. On note (N_i, N_j) le plus petit T- semiflot positif de la place p.



L'expansion d'une place marquée $p = (t_i, t_j)$ correspond à la construction suivante :

1. La transition t_i est dupliquée en N_i transitions notées ${}_1t_i$, ${}_2t_i$, … , ${}_{N_i}t_i$.

   $\Rightarrow$ Pour $k \in \{1, \ldots, N_i - 1\}$, on ajoute les places $r_k = ({}_kt_i\ ,\ {}_{k+1}t_i)$ avec $M_0(r_k) = 0$ jeton et la place $r_{N_i} = ({}_{N_i}t_i\ ,\ {}_1t_i)$ avec $M_0(r_{N_i}) = 1$ jeton.
2. De même, la transition t_j est dupliquée en N_j transitions notées ${}_1t_j$, ${}_2t_j$, … , ${}_{N_j}t_j$.

   $\Rightarrow$ Pour $k \in \{1, \ldots, N_j - 1\}$, on ajoute les places $s_k = ({}_kt_j\ ,\ {}_{k+1}t_j)$ avec $M_0(s_k) = 0$ jeton et la place $s_{N_j} = ({}_{N_j}t_j\ ,\ {}_1t_j)$ avec $M_0(s_{N_j}) = 1$ jeton.

Notons que cette définition des marquages initiaux modélise la contrainte de non-réentrance des transitions initiales. On distingue alors deux cas selon le signe de $w(p) - v(p)$:

- Si $w(p) > v(p)$, alors N_i places sont ajoutées.
  Pour $k = 1, \ldots, N_i$, calculer a_k et b_k tels que :

  $$\begin{cases} b_k \in \{1, \ldots, N_j\} \\ a_k \cdot N_j + b_k = \left\lfloor \frac{M_0(p) + w(p) \cdot (k-1)}{v(p)} \right\rfloor + 1 \\ a_k \in \mathbb{N} \end{cases}$$

  Ajouter une place $p^k = ({}_kt_i\ ,\ {}_{b_k}t_j)$ telle que $M_0(p^k) = a_k$.
- Si $w(p) \leqslant v(p)$, alors N_j places sont ajoutées.
  Pour $k = 1, \ldots, N_j$, calculer c_k et d_k tels que :

  $$\begin{cases} d_k \in \{1, \ldots, N_i\} \\ c_k \cdot N_i + d_k = \left\lceil \frac{k \cdot v(p) - M_0(p)}{w(p)} \right\rceil \\ c_k \in \mathbb{Z}^- \end{cases}$$

  Ajouter une place $q^k = ({}_{d_k}t_i\ ,\ {}_kt_j)$ telle que $M_0(q^k) = -c_k$.

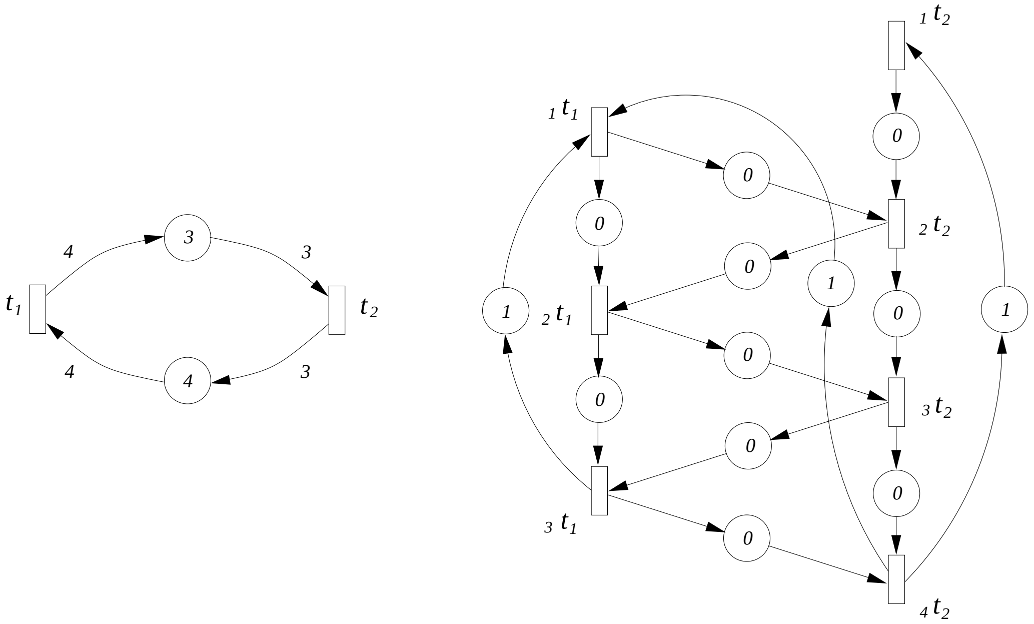


FIG. 3.8: Expansion d'un graphe d'événements généralisé.

Le graphe d'événements ainsi obtenu reflète cet aspect répétitif de la structure des contraintes. Etant donnée une transition duplicata ${}_k t_i$, son n-ième franchissement modélise le $((n-1) \cdot N_i + k)$-ième franchissement de t_i. La figure 3.8 ci-dessous décrit l'expansion d'un graphe d'événements généralisé à deux places.

L'étude des circuits du graphe expansé permet alors de déterminer la vivacité d'un graphe d'événements généralisé. Dans l'exemple de la figure 3.8 ci-dessus, tous les circuits du graphe expansé ont au moins un jeton ce qui implique, d'après le théorème 1, que le graphe d'origine est vivant.

Dans le cas général, Munier [Mun93] montre le résultat suivant qui relie les aspects structurels et comportementaux d'un système :

Théorème 2. *Soit G un graphe d'événements généralisé fortement connexe. Les trois propriétés suivantes sont équivalentes :*

1. *G est K-borné.*
2. *G est unitaire.*
3. *Il existe un T-semiflot $X > 0$ à composantes entières tel que $\Gamma_G \cdot X = 0$ (i.e. G est expansible).*

L'expansion minimale, en nombre de transitions duplicata, est alors calculée en fonction du plus petit T-semiflot de G. Les aspects algorithmiques de cette transformation seront évoqués dans la dernière section de ce chapitre.

3.5 Cas temporisé

Après une brève introduction sur les problèmes d'ordonnancement, nous montrons que le formalisme des GEG temporisés permet de modéliser efficacement des problèmes d'ordonnancement cyclique. Nous présentons ensuite quelques uns des principaux résultats obtenus dans ce domaine avec leurs extensions pour le cas généralisé.

3.5.1 Introduction à l'ordonnancement cyclique

Soit $T = \{t_1, \ldots, t_n\}$ un ensemble fini de tâches ayant chacune une durée d'exécution déterminée notée $l(t_i)$ $\forall i \in \{1, \ldots, n\}$. Soit $C \subseteq (T \times T)$ l'ensemble des contraintes de précédence sur T. Une contrainte de précédence $(t_i, t_j) \in C$ signifie que la tâche t_j ne peut être exécutée avant la fin de la tâche t_i. La date de début d'une tâche t_i est notée $s(t_i)$ et sa date de fin $e(t_i)$. Un ordonnancement est un ensemble de dates de début et de dates de fin. Si de plus, les dates de cet ordonnancement vérifient l'ensemble des contraintes de précédence :

$$i.e.\quad \forall (t_i, t_j) \in C \ : \ e(t_i) \leqslant s(t_j)$$

alors l'ordonnancement est dit valide. Un problème d'ordonnancement consiste à déterminer un ordonnancement valide. Par la suite on considère que les tâches sont non préemptives ($i.e.$ $e(t_i) = s(t_i) + l(t_i)$). Dans ce cas, un ordonnancement est entièrement caractérisé par les dates de début des tâches.

Un problème d'ordonnancement est dit cyclique dès lors qu'au moins une tâche de T doit être exécutée un nombre infini de fois. Une tâche $t \in T$ est appelée tâche générique tandis que ses différentes exécutions seront appelées occurrences de la tâche t. Pour représenter les différentes occurrences d'une même tâche, on adoptera la notation définie précédemment à la sous-section 3.4.3. Aussi, la date de début de l'occurrence $< t_i, \nu_i >$ sera notée $s(t_i, \nu_i)$. Lorsque les différentes exécutions successives d'une même tâche doivent être séquentielles, on dira que la tâche générique est non-réentrante. Par la suite, on considérera les problèmes d'ordonnancement cyclique à tâches génériques non-réentrantes. Ces problèmes trouvent de nombreuses applications dans des domaines comme l'informatique [HM94] ou les problèmes d'ateliers [PX95].

Les réseaux de Petri permettent de représenter naturellement des systèmes dynamiques à événements discrets. Chrétienne et Carlier [CC88] ont montré que de nombreux problèmes d'ordonnancement comme les problèmes d'ordonnancement cyclique peuvent être modélisés simplement à l'aide des graphes d'événements temporisés.

Définition 5. *Un graphe d'événements de marquage initial M_0 noté $G = (T, P, M_0)$ est temporisé s'il existe une fonction l telle que :*

$$l : T \to \mathbb{R}^{+\star}$$

qui associe à chaque transition de T une durée de franchissement. Un graphe d'événements temporisé (GET) sera noté $G = (T, P, M_0, l)$. Cette définition s'étend aux graphes d'événements généralisés que l'on notera GEGT.

Il existe plusieurs politiques d'ordonnancement d'un système selon la nature des contraintes exprimées sur l'ensemble des tâches. Parmi ces politiques, deux grandes tendances sont à distinguer : statique *vs.* dynamique. Une politique d'ordonnancement statique consiste à attribuer à chaque tâche une date de début. Une politique dynamique est un ensemble de règles simples permettant de déterminer à chaque instant quelles tâches exécuter. Pour les systèmes asynchrones comme les réseaux de Petri, une tâche peut être exécutée dès que les contraintes associées à celle-ci sont satisfaites. Cette politique d'ordonnancement dynamique est appelée ordonnancement au plus tôt. Une propriété intéressante de cette politique est la dominance des ordonnancements au plus tôt selon le critère du débit maximum. Le débit (ou la fréquence d'exécution) d'une tâche $t \in T$ noté $\lambda(t)$ étant défini comme la limite du rapport suivant :

$$\lambda(t) = \lim_{n \to \infty} \frac{n}{s(t,n)}$$

Le critère du débit est un critère d'évaluation majeur. Il reflète la cadence d'une chaîne de montage ou d'une application informatique.

3.5.2 Quelques résultats fondamentaux en ordonnancement cyclique

Cette sous-section est consacrée à la présentation du *problème central répétitif* et aux résultats principaux. Cette présentation se veut introductive et n'a pas pour objectif l'exhaustivité. Aussi, le lecteur intéressé pourra se référer aux travaux de Chrétienne [Chr83] ainsi qu'à l'ouvrage de Carlier et Chrétienne [CC88] dédié aux problèmes d'ordonnancement.

Le problème central répétitif proposé par Chrétienne [Chr83] consiste à caractériser le comportement asymptotique de l'ordonnancement au plus tôt d'un problème d'ordonnancement modélisable par un graphe d'événements temporisé. Dans notre contexte, le problème est de calculer le débit des applications modélisées et de dégager éventuellement une structure de l'ensemble des dates d'initiation des processus.

Dans le cas d'un problème cyclique, un ordonnancement est un ensemble infini de dates de début. La notion de K-périodicité introduite par Chrétienne [Chr83] permet de donner une caractérisation concise de la structure des dates de début d'un ordonnancement.

Définition 6. *Une suite u_n est K-périodique s'il existe trois entiers K, q et N tels que $u_{n+K} = u_n + q$ pour $n \geqslant N$. K est le facteur de périodicité, q la période et $\frac{K}{q}$ la fréquence. Si le facteur de périodicité vaut 1, on dira que la suite u_n est périodique.*

La figure 3.9 décrit un exemple de suite 3-périodique de période 8. Les trois premiers termes (*i.e.* $0, 3$ et 5) de la suite décrivent complètement cette suite. La fréquence de cette suite vaut $\frac{3}{8}$.

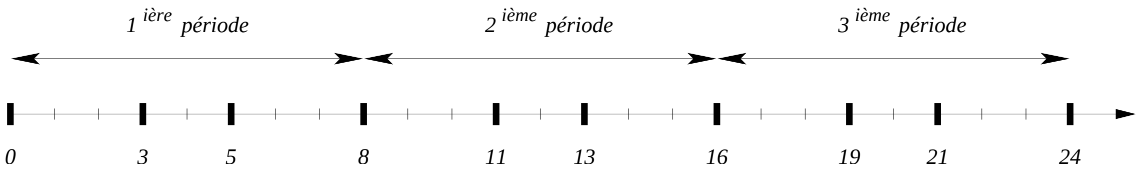


FIG. 3.9: Représentation graphique d'une suite 3-périodique de période 8.

En s'appuyant sur le lemme de Koenig [BBC92], Chrétienne [Chr83] parvient à caractériser le débit de l'ordonnancement au plus tôt d'un graphe d'événements temporisé G par l'analyse de ses circuits élémentaires. Soit C_G l'ensemble des circuits élémentaires de G. On appelle la hauteur d'un circuit C notée $H(C)$ la quantité suivante :

$$H(C) = \sum_{p \in C} M_0(p)$$

La longueur d'un circuit C notée $L(C)$ est définie par :

$$L(C) = \sum_{t \in C} l(t)$$

L'indice $\alpha(C)$ d'un circuit $C \in C_G$ est défini par le rapport de la longueur de ce circuit sur sa hauteur, soit :

$$\alpha(C) = \frac{L(C)}{H(C)}$$

On définit alors l'indice d'un graphe d'événements $\alpha(G)$ comme étant la valeur maximale des indices de ses circuits, soit :

$$\alpha(G) = \max_{C \in C_G} \{\alpha(C)\}$$

Un circuit de G est dit critique si son indice est égal à celui du graphe. Chrétienne montre alors le résultat suivant sur la structure de l'ordonnancement au plus tôt et son débit :

Théorème 3. *Soit G un graphe d'événements temporisé fortement connexe ayant un marquage initial vivant. Il existe un instant $\tau \in \mathbb{R}^+$ à partir duquel les dates de début de l'ordonnancement au plus tôt forment des suites K-périodiques de période $\alpha(G) \cdot K$ où K est le produit des hauteurs des circuits critiques de G. De plus, le débit du système de marquage $M(G)$ noté $\lambda(M(G))$ vaut :*

$$\lambda(M(G)) = \frac{1}{\alpha(G)}$$

Dans [CDQV85], Cohen *et al.* ont caractérisé davantage la valeur du facteur de périodicité K. Celui-ci est en effet égal au plus grand commun diviseur des hauteurs des circuits critiques. Dans le cas général (*i.e.* non fortement connexe), le facteur de périodicité est égal au plus petit commun multiple des facteurs de périodicité de chaque composante fortement connexe.

D'après les notations, le débit d'un graphe d'événements temporisé est aussi égal à :

$$\lambda(M(G)) = \min_{C \in C_G} \frac{H(C)}{L(C)}$$

Le calcul du débit d'un graphe d'événements temporisé se réduit alors à un calcul de circuit de poids moyen minimum dans un graphe orienté bi-valué auxiliaire défini comme suit :

- l'ensemble des transitions du graphe d'événements définit l'ensemble des sommets du graphe auxiliaire ;
- chaque place $p = (t_i, t_j)$ est remplacée dans le graphe auxiliaire par un arc (t_i, t_j) bi-valué par le marquage initial $M_0(p)$ et $l(t_i)$.

Les problèmes de calcul de circuit de poids moyen minimum sont importants et interviennent en théorie des graphes [Kar78], en CAO [IP95], en ordonnancement [LK98] et bien entendu en théorie des systèmes à événements discrets. De nombreux algorithmes polynomiaux ont été développés à cet effet. Dasdan *et al* [DIG99] proposent une étude comparative de différentes méthodes de calcul.

Pour le cas d'un graphe quelconque, Chrétienne généralise le théorème 3 en s'appuyant sur une décomposition du graphe en ses composantes fortement connexes. Cette généralisation est établie par une analyse du graphe réduit G_R dont les sommets représentent les composantes fortement connexes notées $C_1, \ldots, C_k$ du graphe initial. Un arc (C_u, C_v) appartient à G_R si et seulement s'il existe une place $p = (t_i, t_j)$ telle que $t_i \in C_u$ et $t_j \in C_v$. Chrétienne montre alors que les dates de franchissement au plus tôt deviennent K-périodiques à partir d'une certaine date $\tau \in \mathbb{R}^+$. De plus, si $\Gamma^{-1}(C_u)$ désigne l'ensemble des prédécesseurs du sommet C_u dans G_R, alors la fréquence optimale λ_u des transitions de C_u vérifie :

$$\lambda_u = \min \left(\frac{1}{\alpha(C_u)}, \min_{C_v \in \Gamma^{-1}(C_u)} \lambda_v \right)$$

Ces deux résultats sont fondamentaux pour l'analyse et la conception des systèmes à événements discrets. Toutefois, le modèle des GET ne permet pas de prendre en compte des contraintes de précédence linéaires entre les différentes occurrences de transitions. Ce type de contraintes de précédence est fréquent dans les problèmes informatiques. L'usage du modèle des GET semble mal adapté pour l'étude de certains problèmes industriels comme celui traité dans ce mémoire. Une partie du travail de thèse de Munier [Mun91] a consisté à étendre ces résultats aux graphes d'événements généralisés temporisés.

3.5.3 Extension aux graphes d'événements généralisés temporisés

Dans [Mun93], Munier propose une extension des résultats de Chrétienne pour le cas généralisé. Cette généralisation est obtenue par le biais de l'expansion qui permet d'exprimer toutes les contraintes de précédence d'un graphes d'événements généralisé sous la forme d'un graphe d'événements. Ainsi le théorème suivant est l'analogue du théorème 3 pour le cas généralisé.

Théorème 4. *Soit G un graphe d'événements généralisé temporisé unitaire fortement connexe. L'expansion minimale de G est notée G^X et N désigne un T-semiflot de support minimal. Il existe une date $\tau \in \mathbb{R}^+$ à partir de laquelle les dates de franchissement au plus tôt de chaque transition $t_i \in T$ deviennent K-périodiques de fréquence $\frac{N_i}{\alpha(G^X)}$.*

Le théorème 2 de la page 37 justifie l'hypothèse de restriction au cas unitaire. En effet, si le graphe n'est pas unitaire alors il ne peut pas être à la fois vivant et K-borné. Le théorème 3 permet de caractériser l'ordonnancement au plus tôt des transitions de l'expansion du graphe initial. Munier montre alors que les fréquences optimales de franchissement des transitions du graphe initial sont étroitement liées à celles des transitions duplicata dans le graphe expansé G^X. En effet, on notera que la fréquence optimale d'une transition générique t_i est obtenue en multipliant N_i par la fréquence optimale de ses transitions duplicata.

De façon similaire, Munier généralise les résultats de Chrétienne aux GEGT de structure quelconque. Cette généralisation est établie par une analyse des chemins du graphe réduit G_R dont les sommets représentent les composantes unitaires (*i.e.* les classes d'équivalence de la relation « deux transitions t_i et t_j appartiennent à la même composante unitaire si et seulement ces transitions se trouvent sur un circuit unitaire ») notées $C_1, \ldots, C_k$ du GEGT initial. Un arc (C_u, C_v) appartient à G_R si et seulement s'il existe une place $p = (t_i, t_j)$ telle que $t_i \in C_u$ et $t_j \in C_v$. Munier montre ensuite que certains arcs de G_R, considérés comme non-critiques, peuvent être supprimés de G_R tout en préservant la structure initiale de l'ordonnancement au plus tôt. De là, on calcule de façon analogue à celle du problème central répétitif les fréquences optimales de franchissement des transitions génériques. Les fréquences obtenues correspondent à un calcul de plus longs chemins à origine unique dans le graphe réduit.

Le modèle des GEGT permet de modéliser naturellement certains problèmes d'ordonnancement cyclique ou des problèmes d'ateliers. Cependant, les remarques de la section suivante limitent la portée des outils à notre disposition pour l'analyse de tels systèmes.

3.6 Aspects algorithmiques

Nous évoquons à présent les aspects calculatoires de l'ensemble des méthodes algorithmiques proposées dans les sections précédentes. Nous abordons dans un premier temps l'expansion. La complexité des méthodes utilisées pour le problème de la vivacité ou de l'évaluation du débit d'un système découle de celle de l'expansion. Ces problèmes sont alors traités à la suite.

3.6.1 Expansion

La méthode de l'expansion associe à un graphe d'événements généralisé un graphe d'événements non-généralisé. Nous avons vu que cette transformation simplifie l'étude de la vivacité du système ainsi que l'analyse de son débit.

Les valeurs du vecteur d'expansion sont celles du plus petit T-semiflot du graphe d'événements généralisé considéré. Celles-ci peuvent être obtenues en résolvant un problème de plus courts chemins à origine unique d'un graphe orienté auxiliaire. Le lecteur intéressé pourra se référer à l'article [Mun93] pour avoir davantage de détails sur la méthode. La complexité de ce test est alors celle de l'algorithme de plus courts chemins utilisé.

Cependant, l'expansion n'est pas une transformation polynomiale en la taille de l'instance de départ (*i.e.* le graphe d'événements généralisé). En effet, chaque transition t_i est remplacée par N_i transitions duplicata. La taille du graphe expansé est de l'ordre de $\mathcal{O}(m \cdot \max_{t_i \in T} N_i)$.

1. La construction du graphe expansé est de complexité pseudo-polynomiale.
2. La taille du graphe expansé est d'ordre pseudo-polynomial par rapport à la taille de l'instance initiale.

L'expansion est donc une méthode performante sur des instances pour lesquelles les valeurs des fonctions d'écriture $w(p)$ et de lecture $v(p)$ de chaque place p sont raisonnables. Dès lors que ces valeurs deviennent grandes ou que la taille du graphe d'événements généralisé est importante, l'expansion est mise en défaut par la taille de la structure qu'elle génère.

3.6.2 Vivacité

Les méthodes utilisées pour tester la vivacité dépendent de sa nature *i.e.* généralisé ou non-généralisé.

- Dans ce dernier cas, nous avons vu qu'une condition nécessaire et suffisante de vivacité était la présence d'au moins un jeton par circuit. On peut donc tester efficacement la vivacité d'un graphe d'événements en supprimant de ce graphe toutes les places marquées et en testant si le graphe résultant est acyclique. Déterminer la vivacité d'un graphe d'événements non-généralisé se fait donc en $\mathcal{O}(m)$.
- Dans le cas généralisé, nous vérifions la vivacité en utilisant la méthode précédente sur le graphe expansé. Bien que cette dernière méthode soit de complexité polynomiale, la structure de donnée sur laquelle on l'applique est de taille pseudo-polynomiale par rapport à la taille de l'instance initiale. Déterminer la vivacité d'un

graphe d'événements généralisé se fait alors en $\mathcal{O}\left((m \cdot \max_{t_i \in T} N_i)^2\right)$. Du fait de l'absence de certificat de taille polynomial, nous ignorons si le problème de la vivacité d'un graphe d'événements généralisé appartient à la classe NP.

- En revanche, la complexité liée au test de la condition nécessaire de vivacité de la propriété 2 page 32 est polynomiale. Ce test repose sur la méthode de calcul des valeurs du vecteur d'expansion. A nouveau, la complexité de ce test est celle de l'algorithme de calcul des valeurs d'expansions (elle est donc de l'ordre de $\mathcal{O}(m)$).

3.6.3 Débit

Nous étudions à présent la complexité des méthodes d'analyse de débit. Tout comme pour le problème de la vivacité d'un GEG, on observe un fossé de complexité entre le cas généralisé et le cas non généralisé. Cet écart de complexité est dû à la fois à l'expansion et à la taille du graphe expansé qu'elle génère. La technique d'analyse du comportement asymptotique des GEGT est limitée par l'expansion. En effet, bien que les algorithmes utilisés pour déterminer les fréquences optimales des transitions soient de complexité polynomiale, ces derniers sont appliqués au graphe expansé dont la taille est proportionnelle au plus petit T-semiflot du graphe d'origine. Dans sa thèse de doctorat, Munier [Mun91] montre par un formalisme équivalent que la détermination des fréquences optimales des transitions d'un graphe d'événements généralisé temporisé est de l'ordre de $\mathcal{O}\left(m^5 + (m \cdot \max_{t_i \in T} N_i)^3 \cdot \log(m \cdot \max_{t_i \in T} N_i)\right)$. Ces méthodes sont donc de complexité pseudo-polynomiale.

Conclusion

Nous avons exposé dans cette section le problème de dimensionnement des mémoires pour les applications embarquées. Ce problème est formulé comme un problème d'optimisation. Nous avons ensuite justifié l'adoption du formalisme des graphes d'événements généralisés pour traiter ce problème. De là, nous rappelons quelques résultats et propriétés importantes concernant ce modèle comme notamment le propriété de vivacité. L'expansion d'un graphe d'événements généralisé est ensuite présentée. Comme l'a montré Munier, cette transformation est un concept central pour l'analyse des propriétés des graphes d'événements généralisés. Nous avons également évoqué l'importance des graphes d'événements généralisés pour la modélisation des problèmes d'ordonnancement cyclique. Enfin, nous soulignons les difficultés de résolution pour des problèmes réels de grande taille en examinant la complexité algorithmique des méthodes utilisées.

Première partie

Vivacité

Chapitre 4

Normalisation et vivacité des GEG

Sommaire

Ce chapitre traite de la vivacité des graphes d'événements généralisés fortement connexes. Le problème de la vivacité des graphes d'événements généralisés est un problème difficile et déterminant pour le problème du dimensionnement des mémoires. Contrairement au cas non-généralisé, il semble qu'il n'existe pas de certificat de taille polynomiale pour le problème de la vivacité sur des graphes d'événements généralisés. Actuellement, les algorithmes pour le calcul de la vivacité sont tous de complexité pseudo-polynomiale [Mun93, TCWCS92, ČP93]. Ces algorithmes sont donc limités par la taille des instances à traiter. Les systèmes embarqués actuels étant de plus en plus complexes, il convient de mettre au point des techniques algorithmiques efficaces pour vérifier leur vivacité. Nous proposons dans ce chapitre une condition suffisante de vivacité et un algorithme efficace permettant de vérifier cette condition suffisante.

Nous présentons d'abord trois résultats importants permettant de préciser l'incidence du marquage d'une place sur l'ensemble des contraintes de précédence engendrées. Nous montrons que l'on peut se restreindre à considérer les marquages d'une place p qui sont multiples du plus grand commun diviseur des fonctions d'écriture $w(p)$ et de lecture $v(p)$. Nous montrons ensuite qu'une place et son marquage peuvent être remplacés par une autre place dont les attributs (*i.e.* marquage initial et fonctions d'écriture et de lecture) sont tous multiples des attributs initiaux tout en préservant la structure des contraintes de précédence.

Nous présentons ensuite un nouveau concept simplifiant la manipulation des graphes d'événements généralisés : la normalisation. La normalisation est une transformation du graphe qui préserve la structure des contraintes de précédence. Nous observons alors que le graphe normalisé associé à un système a un nombre de jetons constant sur chacun de ses circuits quelle que soit la séquence valide de franchissements considérée. Cette propriété d'invariance est exploitée pour établir une condition suffisante de vivacité sur les graphes normalisés. De plus, cette condition suffisante s'avère être également nécessaire dans le cas des circuits à deux transitions. Enfin nous élaborons un algorithme polynomial pour la vérification de cette condition suffisante de vivacité.

4.1 Résultats préliminaires

Nous présentons dans cette section trois résultats donnant une description approfondie de la relation existant entre le marquage d'une place et l'ensemble des contraintes de précédence que cette place marquée génère. Ces résultats sont en outre nécessaires à l'élaboration de résultats permettant une simplification des problèmes d'optimisation sur ce type de système.

4.1.1 Marquages et structure des contraintes de précédence

Le résultat suivant montre qu'en retirant une même quantité de jetons à deux places modélisant les mêmes contraintes de précédence, nous obtenons deux places marquées qui engendrent de nouveau des contraintes de précédence identiques.

Lemme 2. *Soient $p_1 = (t_i, t_j)$ et $p_2 = (t_i, t_j)$ deux places telles que $v(p_1) = v(p_2)$ et $w(p_1) = w(p_2)$ décrivant le même ensemble de contraintes précédence entre les occurrences de t_i et t_j (voir figure 4.1 de la présente page). Soit $\alpha \in \mathbb{N}$ tel que :*

$$0 \leq \alpha \leq \min\left(\left\lfloor \frac{M_0(p_1)}{pgcd_{p_1}} \right\rfloor, \left\lfloor \frac{M_0(p_2)}{pgcd_{p_2}} \right\rfloor\right)$$

Alors, les places p_1 et p_2 de marquages initiaux respectifs $M_0'(p_1) = M_0(p_1) - \alpha \cdot pgcd_{p_1}$ et $M_0'(p_2) = M_0(p_2) - \alpha \cdot pgcd_{p_2}$ décrivent le même ensemble de contraintes de précédence.

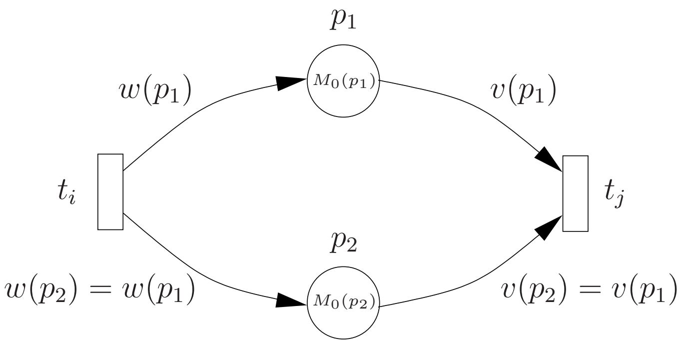


FIG. 4.1: Deux places $p_1 = (t_i, t_j)$ et $p_2 = (t_i, t_j)$ partageant les mêmes transitions.

Démonstration. Pour toute place p, $S(p, M_0(p))$ désigne l'ensemble des contraintes de précédence induites par p et son marquage initial $M_0(p)$. En respectant les hypothèses du lemme, nous montrons que $S(p_1, M_0'(p_1)) = S(p_2, M_0'(p_2))$.

Pour cela, supposons que la place p_1 et son marquage initial $M_0'(p_1)$ induisent une contrainte de précédence entre les occurrences $< t_i, \nu_i >$ et $< t_j, \nu_j >$. Alors, d'après le lemme 1 page 35, nous avons :

$$w(p_1) > M_0(p_1) - \alpha \cdot pgcd_{p_1} + w(p_1) \cdot \nu_i - v(p_1) \cdot \nu_j \geq \max(w(p_1) - v(p_1), 0)$$

D'après l'identité de BÉZOUT, il existe un couple $(A, B) \in \mathbb{Z} \times \mathbb{Z}$ tel que $A \cdot w(p_1) + B \cdot v(p_1) = pgcd_{p_1}$. Considérons la plus petite valeur $\beta \in \mathbb{Z}$ telle que :

$$\beta > \max\left(w(p_1) \cdot \left(A - \frac{\nu_i}{\alpha}\right), v(p_1) \cdot \left(-B - \frac{\nu_j}{\alpha}\right)\right)$$

$$\text{avec} \quad \beta \equiv 0(w(p_1)) \quad \text{et} \quad \beta \equiv 0(v(p_1))$$

On observe que $pgcd_{p_1} = \left(A - \frac{\beta}{w(p_1)}\right) \cdot w(p_1) + \left(B + \frac{\beta}{v(p_1)}\right) \cdot v(p_1)$. Alors, en posant $\nu_i' = \nu_i - \alpha \cdot \left(A - \frac{\beta}{w(p_1)}\right)$ et $\nu_j' = \nu_j + \alpha \cdot \left(B + \frac{\beta}{v(p_1)}\right)$, nous obtenons :

$$w(p_1) > M_0(p_1) + w(p_1) \cdot \nu_i' - v(p_1) \cdot \nu_j' \geq \max(w(p_1) - v(p_1), 0)$$

De par la définition de l'entier β, nous avons $(\nu_i', \nu_j') \in \mathbb{N} \times \mathbb{N}$. L'inégalité précédente induit alors une contrainte de précédence entre les occurrences $< t_i, \nu_i' >$ et $< t_j, \nu_j' >$. De plus, nous avons supposé que $S(p_1, M_0(p_1)) = S(p_2, M_0(p_2))$, d'où :

$$w(p_2) > M_0(p_2) + w(p_2) \cdot \nu_i' - v(p_2) \cdot \nu_j' \geq \max(w(p_2) - v(p_2), 0)$$

Comme $\nu_i' = \nu_i - \alpha \cdot \left(A - \frac{\beta}{w(p_2)}\right)$ et $\nu_j' = \nu_j + \alpha \cdot \left(B + \frac{\beta}{v(p_2)}\right)$, nous obtenons alors :

$$w(p_2) > M_0(p_2) - \alpha \cdot pgcd_{p_2} + w(p_2) \cdot \nu_i - v(p_2) \cdot \nu_j \geq \max(w(p_2) - v(p_2), 0)$$

La place p_2 de marquage initial $M_0(p_2)$ engendre également une contrainte de précédence entre le ν_i-ième franchissement de t_i et le ν_j-ième franchissement de t_j. Ceci complète la preuve. $\square$

Ce résultat sera utilisé ultérieurement pour comparer des ensembles de contraintes de précédence induites par des places ayant des marquages moindres.

4.1.2 Notion de jetons utiles

Le lemme suivant montre que l'on peut se restreindre à considérer pour toute place p des marquages qui soient multiples de $pgcd_p$. Etant donné un marquage, il est alors possible de retirer un certain nombre de jetons du marquage initial sans modifier la structure des contraintes de précédence entre les différentes occurrences des transitions du graphe. Les jetons ainsi retirés sont superflus pour la représentation de cet ensemble de contraintes de précédence. Nous montrons à la fin de ce chapitre qu'il s'agit du nombre maximum de jetons que l'on peut retirer.

Lemme 3. *Le marquage initial $M_0(p)$ d'une place $p = (t_i, t_j)$ peut être remplacé par $M_0^*(p) = \left\lfloor \frac{M_0(p)}{pgcd_p} \right\rfloor \cdot pgcd_p$ sans incidence sur l'ensemble des contraintes de précédence induites par la place p.*

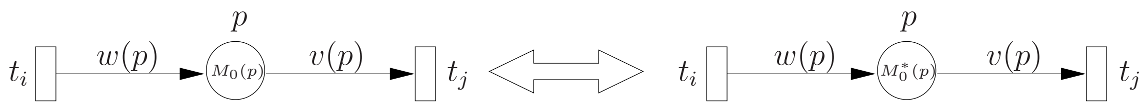


Fig. 4.2: Notion de jetons utiles

Démonstration. Soit $p = (t_i, t_j)$ une place de G. Nous désignons par A (*resp.* B) l'ensemble des contraintes de précédence induites par la place p avec $M_0(p)$ jetons initiaux (*resp.* $M_0^*(p)$). Nous démontrons alors que $A = B$.

Pour cela, nous nous servons de la division euclidienne de $M_0(p)$ par $pgcd_p$, soit :

$$M_0(p) = M_0^*(p) + R_{pgcd}(M_0(p))$$

où $R_{pgcd}(M_0(p))$ désigne le reste de cette division.

- $A \subseteq B$: Considérons dans un premier temps une contrainte de précédence de A entre les occurrences $< t_i, \nu_i >$ et $< t_j, \nu_j >$. D'après le lemme 1, nous avons :

  $$w(p) > M_0(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geq \max(w(p) - v(p), 0)$$

  Par substitution de $M_0(p)$, nous obtenons alors :

  $$w(p) > M_0^*(p) + R_{pgcd}(M_0(p)) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geq \max(w(p) - v(p), 0)$$

  Il s'ensuit que :

  $$w(p) > M_0^*(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j$$

  D'autre part, nous observons que :

  $$M_0^*(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \equiv 0(pgcd_p)$$

  $$\max(w(p) - v(p), 0) \equiv 0(pgcd_p)$$

  $$\text{et} \quad R_{pgcd}(M_0(p)) \in \{0, \ldots, pgcd_p - 1\}$$

  Nous en déduisons donc que :

  $$M_0^*(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geq \max(w(p) - v(p), 0)$$

  Ce faisant, la contrainte de précédence définie entre les occurrences $< t_i, \nu_i >$ et $< t_j, \nu_j >$ appartient également à B.

- $B \subseteq A$: Considérons à présent une contrainte de précédence de B définie entre les occurrences $< t_i, \nu_i >$ et $< t_j, \nu_j >$.
  D'après le lemme 1, nous avons :

  $$w(p) > M_0^*(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geq \max(w(p) - v(p), 0)$$

  Nous en déduisons aussi :

  $$M_0^*(p) + R_{pgcd}(M_0(p)) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geq \max(w(p) - v(p), 0)$$

Or $M_0^*(p) + \nu_i \cdot w(p) - \nu_j \cdot v(p) \equiv 0(pgcd_p)$, d'où :

$$w(p) - pgcd_p \geq M_0^*(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j$$

Sachant que $R_{pgcd}(M_0(p)) < pgcd_p$, nous avons alors :

$$w(p) > M_0^*(p) + R_{pgcd}(M_0(p)) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geq \max(w(p) - v(p), 0)$$

Il s'ensuit que la contrainte de précédence définie entre les occurrences $< t_i, \nu_i >$ et $< t_j, \nu_j >$ appartient aussi à A.

$\square$

Ce résultat permet de supprimer des jetons des marquages initiaux tout en préservant la structure des contraintes de précédence induites par ces places. Ces suppressions préservent donc la propriété de vivacité du système considéré. Les jetons restant sont qualifiés d'utiles et définissent le marquage utile de la place. De plus, la notion de marquage utile permet de limiter sensiblement la combinatoire liée à la définition des marquages initiaux. Désormais, nous considérerons des instances ayant des marquages utiles.

4.1.3 Places équivalentes

Deux places $p_1 = (t_i, t_j)$ et $p_2 = (t_i, t_j)$ sont dites équivalentes si elles induisent les mêmes contraintes de précédence entre les différentes occurrences de franchissement de leurs transitions adjacentes.

Lemme 4. *Deux places $p_1 = (t_i, t_j)$ et $p_2 = (t_i, t_j)$ telles que $\frac{w(p_2)}{w(p_1)} = \frac{v(p_2)}{v(p_1)} = \frac{M_0(p_2)}{M_0(p_1)} = \Delta \in \mathbb{R}^{+\star}$ sont équivalentes.*

Démonstration. Nous notons A_1 (*resp.* A_2) l'ensemble des contraintes de précédence entre les occurrences de t_i et t_j induites par la place p_1 (*resp.* p_2). Considérons alors une contrainte de précédence $a \in A_1$ définie entre $< t_i, \nu_i >$ et $< t_j, \nu_j >$. Alors d'après le lemme 1 page 35, nous obtenons :

$$w(p_1) > M_0(p_1) + w(p_1) \cdot \nu_i - v(p_1) \cdot \nu_j \geqslant \max(w(p_1) - v(p_1), 0)$$

$$\Updownarrow \times\Delta$$

$$\Delta \cdot w(p_1) > \Delta \cdot [M_0(p_1) + w(p_1) \cdot \nu_i - v(p_1) \cdot \nu_j] \geqslant \Delta \cdot \max(w(p_1) - v(p_1), 0)$$

$$\Updownarrow$$

$$w(p_2) > M_0(p_2) + w(p_2) \cdot \nu_i - v(p_2) \cdot \nu_j \geqslant \max(w(p_2) - v(p_2), 0)$$

Cette dernière inégalité est équivalente à $a \in A_2$, ce qui complète la preuve. $\square$

Ce résultat montre qu'il existe plusieurs représentations possibles d'un ensemble de contraintes de précédence sous forme d'une place marquée d'un graphe d'événements généralisé. Ce résultat est utile pour l'étude de la propriété de vivacité.

4.2 La normalisation

Dans cette section, nous montrons que tout graphe d'événements généralisé peut être transformé en un autre graphe d'événements normalisé équivalent *i.e.* un graphe d'événements généralisé pour lequel toutes les fonctions de marquages adjacentes à une transition t_i sont égales. La normalisation est un outil central dans cette thèse qui permet de simplifier l'approche de certains problèmes comme celui de la vivacité.

Nous présentons dans un premier temps une définition formelle de la normalisation. Nous montrons ensuite que tout graphe unitaire peut être normalisé. Enfin, nous mettons en lumière le lien existant entre la normalisation et l'expansion d'un graphe unitaire.

4.2.1 Définition et transformation du problème

Définition 7. *Une transition t_i est dite normalisée s'il existe $Z_i \in \mathbb{N}^\star$ tel que :*

$$\begin{array}{lllllllll} \forall(p_1,p_2) & \in & \mathcal{P}^+(t_i) & \times & \mathcal{P}^-(t_i), & w(p_1) & = & v(p_2) & = Z_i \\ \forall(p_1,p_2) & \in & \mathcal{P}^+(t_i) & \times & \mathcal{P}^+(t_i) & w(p_1) & = & w(p_2) & = Z_i \\ \forall(p_1,p_2) & \in & \mathcal{P}^-(t_i) & \times & \mathcal{P}^-(t_i) & v(p_1) & = & v(p_2) & = Z_i \end{array}$$

La valeur Z_i est appelée valeur de normalisation de la transition t_i. On dira qu'un graphe d'événements généralisé est normalisé si toutes ses transitions sont normalisées.

D'après le lemme 4 sur les places équivalentes, chaque place $p_a \in P$ peut être remplacée par une place équivalente p'_a obtenue en multipliant $w(p_a)$, $v(p_a)$ et $M_0(p_a)$ par un entier α_a. L'idée est de construire pour tout graphe unitaire $G = (T, P)$ fortement connexe un graphe d'événements généralisé $G' = (T, P')$ en utilisant cette propriété et tel que toutes les transitions soient normalisées. Le graphe d'événements généralisé résultant de cette transformation conserve la structure des contraintes de précédence du graphe initial et donc la propriété de vivacité.

Considérons alors une transition $t_i \in T$. D'après la définition 7, cette transition peut être normalisée si et seulement s'il existe un vecteur $\alpha \in \{\mathbb{N}^{\star +}\}^{|P|}$ tel que :

vw Pour tout couple $(p_a, p_b) \in \mathcal{P}^-(t_i) \times \mathcal{P}^+(t_i)$,

$$\alpha_a \cdot v(p_a) = \alpha_b \cdot w(p_b) \quad \Leftrightarrow \quad \left\{ \begin{array}{lll} \frac{\alpha_a}{\alpha_b} & \leqslant & \frac{w(p_b)}{v(p_a)} \\ \frac{\alpha_b}{\alpha_a} & \leqslant & \frac{v(p_a)}{w(p_b)} \end{array} \right.$$

ww Pour tout couple $(p_a, p_b) \in \mathcal{P}^+(t_i) \times \mathcal{P}^+(t_i)$,

$$\alpha_a \cdot w(p_a) = \alpha_b \cdot w(p_b) \quad \Leftrightarrow \quad \left\{ \begin{array}{lll} \frac{\alpha_a}{\alpha_b} & \leqslant & \frac{w(p_b)}{w(p_a)} \\ \frac{\alpha_b}{\alpha_a} & \leqslant & \frac{w(p_a)}{w(p_b)} \end{array} \right.$$

vv Pour tout couple $(p_a, p_b) \in \mathcal{P}^-(t_i) \times \mathcal{P}^-(t_i)$,

$$\alpha_a \cdot v(p_a) = \alpha_b \cdot v(p_b) \quad \Leftrightarrow \quad \left\{ \begin{array}{ccc} \frac{\alpha_a}{\alpha_b} & \leqslant & \frac{v(p_a)}{v(p_b)} \\ \frac{\alpha_b}{\alpha_a} & \leqslant & \frac{v(p_b)}{v(p_a)} \end{array} \right.$$

Pour résoudre un tel système, nous allons exprimer l'ensemble de ces contraintes comme un système de contraintes de potentiel (*voir* [CLR90]). Nous explicitons à présent la construction de ces contraintes ainsi que celle du graphe de potentiel $\mathcal{G} = (V \cup \{s\}, E)$ associé :

- Chaque sommet $a \in V$ correspond à une place $p_a \in P$;
- Chaque arc $e = (a, b) \in E$ est associé à un couple de places adjacentes (p_a, p_b) (*i.e.* p_a et p_b sont adjacentes si elles ont au moins une transition en commun). Soit t_e une transition commune à ce couple de places. Si nous posons :

  $$\beta(t_e, p_a) = \left\{ \begin{array}{lll} w(p_a) & \text{si} & p_a \in \mathcal{P}^+(t_e) \\ v(p_a) & \text{sinon} & \end{array} \right.$$

  alors, nous pouvons associer à l'arc e la contrainte suivante :

  $$\log \alpha_b - \log \alpha_a \leqslant \mathcal{B}_{(a,b)}$$

  où $\mathcal{B}_{(a,b)} = \log \frac{\beta(t_e,p_a)}{\beta(t_e,p_b)}$. L'arc $(a, b) \in E$ est alors valué par $\mathcal{B}_{(a,b)}$;
- $\forall a \in V$, nous ajoutons les arcs (s, a) valués par 0.

Soit μ un chemin dans $\mathcal{G}$. La valeur $\mathcal{B}(\mu)$ de ce chemin est définie par :

$$\mathcal{B}(\mu) = \sum_{e \in \mu \cap E} \mathcal{B}_e$$

La figure 4.3 page ci-contre illustre la définition du graphe valué $\mathcal{G}$.

Nous avons ainsi montré que le système de contraintes initial se réduit à un système de contraintes de potentiel. On exprime alors ce dernier par un graphe de potentiel $\mathcal{G}$. Le vecteur $\alpha \in \{\mathbb{N}^{\star +}\}^{|P|}$ existe si et seulement si $\mathcal{G}$ ne possède pas de circuit de valeur strictement négative [CLR90]. De plus, ce vecteur peut être déterminé en temps polynomial en utilisant un algorithme de plus courts chemins tel que l'algorithme de Bellman-Ford [CLR90].

Après avoir présenté un exemple sur la construction d'un graphe de potentiel, nous démontrons que tout graphe unitaire admet des valeurs de normalisation.

4.2.2 Exemple

Nous présentons un exemple complet décrit sur la figure 4.4 page suivante. On constate que le graphe d'événements généralisé initialement présenté n'est pas normalisé. On pose alors dans un premier temps les équations associées à la normalisation de ce système.

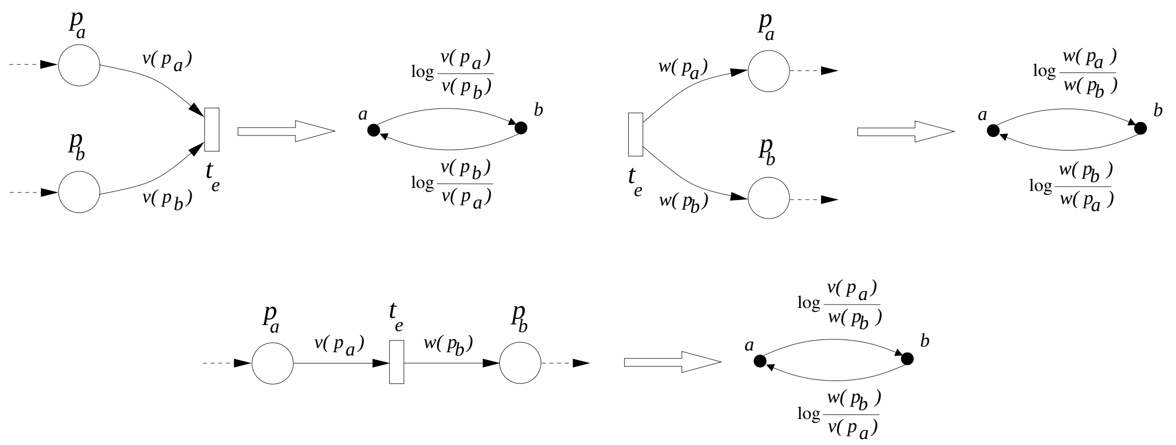
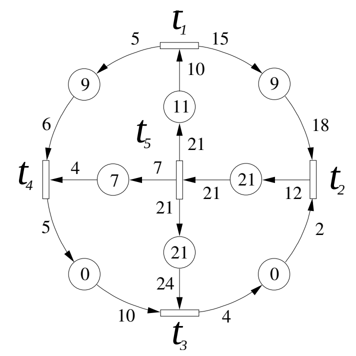


FIG. 4.3: Soient p_a et p_b deux places adjacentes ayant la transition t_e en commun. Nous décrivons les trois cas possibles pour la construction de $\mathcal{G}$.



$$\left\{\begin{array}{lclclcl}
15 \cdot \alpha_{p_{1,2}} & = & 10 \cdot \alpha_{p_{5,1}} & = & 5 \cdot \alpha_{p_{1,4}} & & \\
18 \cdot \alpha_{p_{1,2}} & = & 12 \cdot \alpha_{p_{2,5}} & = & 2 \cdot \alpha_{p_{3,2}} & & \\
4 \cdot \alpha_{p_{3,2}} & = & 24 \cdot \alpha_{p_{5,3}} & = & 10 \cdot \alpha_{p_{4,3}} & & \\
5 \cdot \alpha_{p_{4,3}} & = & 4 \cdot \alpha_{p_{5,4}} & = & 6 \cdot \alpha_{p_{1,4}} & & \\
21 \cdot \alpha_{p_{2,5}} & = & 21 \cdot \alpha_{p_{5,1}} & = & 21 \cdot \alpha_{p_{5,3}} & = & 7 \cdot \alpha_{p_{5,4}}
\end{array}\right.$$

FIG. 4.4: Soit G un graphe unitaire et le système d'équations associées à la normalisation de G.

Dans cet exemple, une place définie entre les transitions t_i et t_j est notée $p_{i,j}$. La i-ième ligne du système d'équations est associée à la transition t_i. La figure 4.5 page suivante présente le graphe associé $\mathcal{G}$. Pour des raisons de lisibilité, le sommet s et ses arcs adjacents ne sont pas représentés.

4.2.3 Théorème principal pour la normalisation

Nous montrons à présent que $\mathcal{G}$ ne possède pas de circuit de valeur strictement négative. Il est donc possible de calculer un potentiel sur les sommets de ce graphe [CLR90]. Les valeurs ainsi calculées permettrons de construire un graphe équivalent normalisé.

Nous démontrons dans un premier temps que tout circuit de taille 2 de $\mathcal{G}$ est de valeur nulle. De même, nous montrons par la suite que tous les circuits de $\mathcal{G}$ sont de valeur nulle.

Lemme 5. *Tout circuit de $\mathcal{G}$ ayant $q = 2$ sommets est de valeur nulle.*

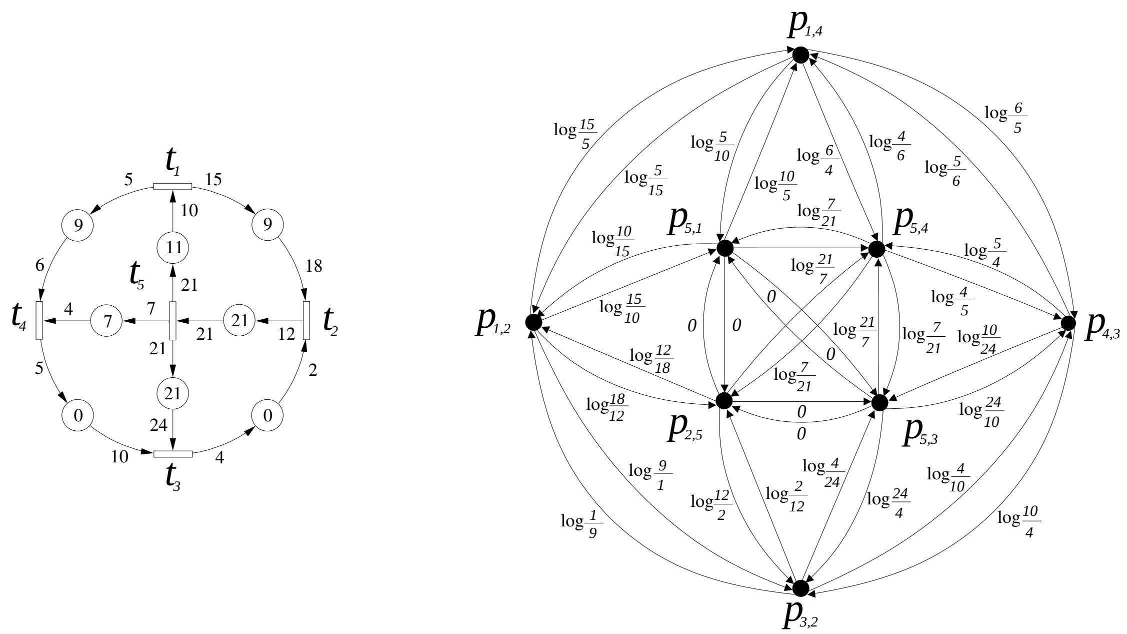


FIG. 4.5: Le graphe unitaire G et $\mathcal{G}$ son graphe associé. Chaque place $p = (t_i, t_j)$ de $G = (T, P)$ est associée à un sommet $p_{i,j}$ dans $\mathcal{G}$.

Démonstration. Soient (a, b, a) un circuit de taille 2 du graphe $\mathcal{G}$ et p_a (*resp.* p_b) la place associée de G au sommet a (*resp.* b). Nous distinguons alors trois cas pouvant engendrer un circuit de taille $q = 2$ dans le graphe valué $\mathcal{G}$:

1. Les places $p_a = (t_1, t_2)$ et $p_b = (t_2, t_1)$ forment un circuit dans G (*voir* figure 4.6 page 57). Alors d'après les règles de construction de $\mathcal{G}$, il existe deux arcs (a, b) dans $\mathcal{G}$ valués respectivement par $\log \frac{w(p_a)}{v(p_b)}$ et $\log \frac{v(p_a)}{w(p_b)}$. Le graphe d'événements généralisé G étant unitaire, nous avons :

$$\frac{w(p_a)}{v(p_a)} . \frac{w(p_b)}{v(p_b)} = 1$$

   Nous en déduisons que ces deux arcs ont la même valeur. De même, les deux arcs (b, a) étant tout deux valués par $\log \frac{v(p_b)}{w(p_a)}$, nous en déduisons que la valeur des circuits correspondants dans $\mathcal{G}$ est nulle.

2. Les places $p_a = (t_1, t_2)$ et $p_b = (t_1, t_2)$ de G partagent les mêmes transitions finales et initiales (comme décrit sur la figure 4.7 page suivante). Alors, selon les règles de construction de $\mathcal{G}$, il existe deux arcs (a, b) dans $\mathcal{G}$ valués respectivement par $\log \frac{w(p_a)}{w(p_b)}$ et $\log \frac{v(p_a)}{v(p_b)}$.
   Or, G étant un graphe unitaire fortement connexe, il existe un chemin ν reliant les transitions t_2 et t_1 tel que :

$$W(\nu) \cdot \frac{w(p_a)}{v(p_a)} = W(\nu) \cdot \frac{w(p_b)}{v(p_b)} = 1$$

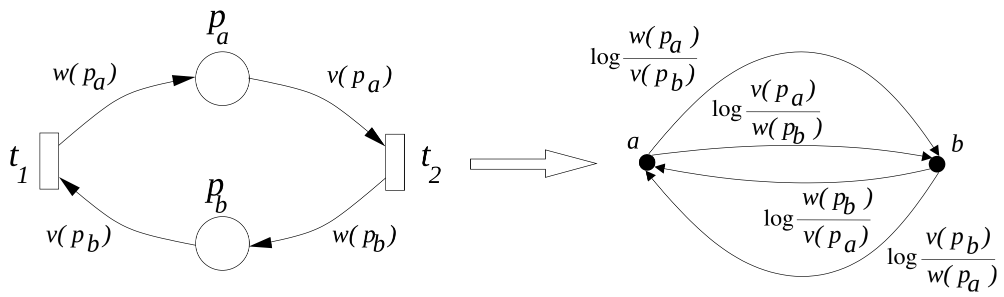


FIG. 4.6: Les places $p_a = (t_1, t_2)$ et $p_b = (t_1, t_2)$ forment un circuit dans G. Le graphe $\mathcal{G}$ correspondant possède alors quatre arcs.

Nous en déduisons que les deux arcs (a, b) de $\mathcal{G}$ sont de même valeur. De la même manière, les arcs (b, a) sont valués par $\log \frac{w(p_b)}{w(p_a)}$. Il s'ensuit que les valeurs des circuits correspondants de $\mathcal{G}$ sont nulles.

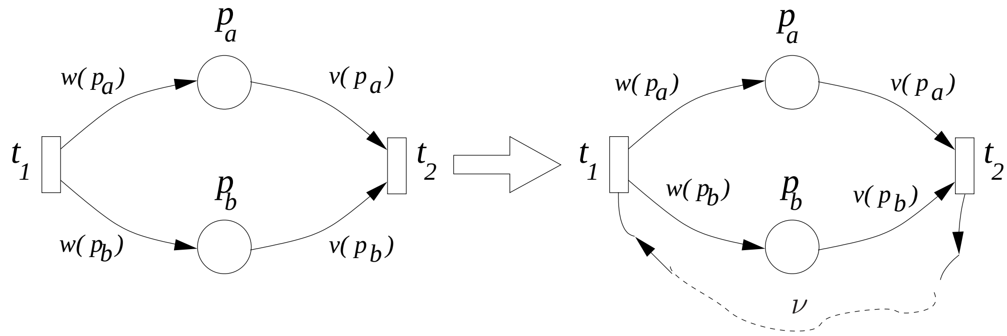


FIG. 4.7: Les places $p_a = (t_1, t_2)$ et $p_b = (t_1, t_2)$ partagent les mêmes transitions initiales et finales.

3. Les places p_a et p_b ont en commun une unique transition t_e. La figure 4.3 page 55 décrit les trois sous-cas découlant de cette situation ainsi que les trois circuits $C = (a, b, a)$ correspondants dans $\mathcal{G}$. On remarque alors que $\mathcal{B}(C) = \mathcal{B}_{(a,b)} + \mathcal{B}_{(b,a)} = 0$.

□

Le résultat suivant établit qu'il existe pour tout circuit $\mathcal{C}$ de $\mathcal{G}$ un circuit élémentaire de même valeur dans $\mathcal{G}$.

Lemme 6. *Soit i un sommet du circuit $\mathcal{C} = (1, \ldots, q, 1)$ avec $q > 2$ tel que les arcs $e_i = (i-1, i)$ et $e_{i+1} = (i, i+1)$ correspondent à la même transition de G (i.e. $t_{e_i} = t_{e_{i+1}}$). Alors, $\mathcal{C}' = (1, 2, \ldots, i-1, i+1, \ldots, q, 1)$ est aussi un circuit de $\mathcal{G}$ et $\mathcal{B}(\mathcal{C}') = \mathcal{B}(\mathcal{C})$.*

Démonstration. Selon l'énoncé du lemme, nous désignons par p_{i-1}, p_i et p_{i+1} les places respectivement associées aux sommets $i-1$, i et $i+1$ de $\mathcal{G}$. D'après les règles de construction de $\mathcal{G}$ et comme $t_{e_i} = t_{e_{i+1}}$, nous avons :

$$\begin{aligned} \mathcal{B}_{e_i} + \mathcal{B}_{e_{i+1}} &= \log \frac{\beta(t_{e_i}, p_{i-1})}{\beta(t_{e_i}, p_i)} + \log \frac{\beta(t_{e_i}, p_i)}{\beta(t_{e_i}, p_{i+1})} \\ &= \log \frac{\beta(t_{e_i}, p_{i-1})}{\beta(t_{e_i}, p_{i+1})} \end{aligned}$$

Les places p_{i-1} et p_{i+1} étant adjacentes à la transition t_{e_i}, il existe un arc $(i-1, i+1)$ dans le graphe $\mathcal{G}$ valué par $\log \frac{\beta(t_{e_i}, p_{i-1})}{\beta(t_{e_i}, p_{i+1})}$ (*voir* figure 4.8 de la présente page). Nous en

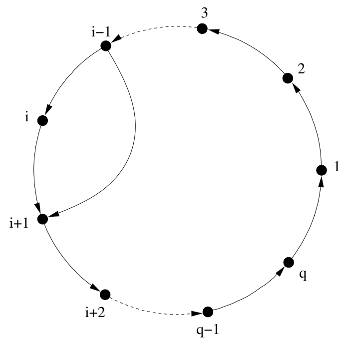


FIG. 4.8: Illustration du lemme 6 page précédente.

déduisons qu'il existe un sous-circuit $\mathcal{C}' = (1, 2, \ldots, i-1, i+1, \ldots, q, 1)$ ayant même valeur que le circuit $\mathcal{C}$. □

Nous pouvons à présent démontrer que les graphes unitaires sont normalisables.

Théorème 5. *Tout circuit de $\mathcal{G}$ est de valeur nulle.*

Démonstration. Soit $\mathcal{C} = (1, \ldots, q, 1)$ un circuit de $\mathcal{G}$. En utilisant le lemme 6, nous pouvons supposer sans perte de généralité que le circuit $\mathcal{C}$ vérifie l'une des propriétés suivantes :

Propriété 1 $q = 2$;

Propriété 2 $q > 2$ et deux arcs consécutifs de $\mathcal{C}$ sont associés à des transitions distinctes :

$$t_{e_1} \neq t_{e_q} \quad \text{et} \quad \forall i \in \{1, \ldots, q-1\},\ t_{e_i} \neq t_{e_{i+1}}$$

D'après le lemme 5, si la propriété 1 est vérifiée, alors le circuit $\mathcal{C}$ est de valeur nulle. Nous pouvons donc supposer que le circuit $\mathcal{C}$ vérifie la propriété 2. D'autre part, nous rappelons que chaque sommet i du circuit $\mathcal{C}$ correspond à une place $p_i \in P$. Nous notons les arcs du circuit $\mathcal{C}$ par $e_i = (i-1, i)$ pour $i \in \{2, \ldots, q\}$ et $e_1 = (q, 1)$. Afin de simplifier les équations, nous posons $e_{q+1} = e_1$. Par définition de $\mathcal{G}$, nous avons :

$$\begin{aligned} \mathcal{B}(\mathcal{C}) &= \sum_{i=2}^{q} \log \frac{\beta(t_{e_i}, p_{i-1})}{\beta(t_{e_i}, p_i)} + \log \frac{\beta(t_{e_1}, p_q)}{\beta(t_{e_1}, p_1)} \\ &= \sum_{i=1}^{q-1} \log \frac{\beta(t_{e_{i+1}}, p_i)}{\beta(t_{e_i}, p_i)} + \log \frac{\beta(t_{e_1}, p_q)}{\beta(t_{e_q}, p_q)} \end{aligned}$$

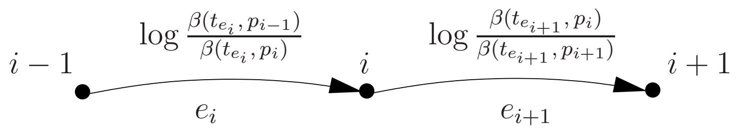


FIG. 4.9: Deux arcs valués du graphe $\mathcal{G}$

A présent, la place p_i est adjacente aux transitions t_{e_i} et $t_{e_{i+1}}$ avec $t_{e_i} \neq t_{e_{i+1}}$. Nous pouvons alors associer un circuit C dans G comme suit :

- Les transitions $t_{e_1}, \ldots, t_{e_q}$ appartiennent à C ;
- Chaque place p_i est adjacente aux transitions t_{e_i} et $t_{e_{i+1}}$. Nous devons considérer deux sous-cas :
  (a) Si $p_i = (t_{e_i}, t_{e_{i+1}})$, nous obtenons $\dfrac{\beta(t_{e_i},\, p_i)}{\beta(t_{e_{i+1}},\, p_i)} = \dfrac{w(p_i)}{v(p_i)} = W(p_i)$. Dans ce cas, nous ajoutons la place p_i au circuit C ;
  (b) Si $p_i = (t_{e_{i+1}}, t_{e_i})$ alors $\dfrac{\beta(t_{e_i},\, p_i)}{\beta(t_{e_{i+1}},\, p_i)} = \dfrac{v(p_i)}{w(p_i)}$. Le graphe G étant unitaire et fortement connexe, il existe un chemin μ_i issu de la transition t_{e_i} vers la transition $t_{e_{i+1}}$ tel que $W(\mu_i) = \frac{w(p_i)}{v(p_i)}$. Nous ajoutons alors le chemin μ_i au circuit C.

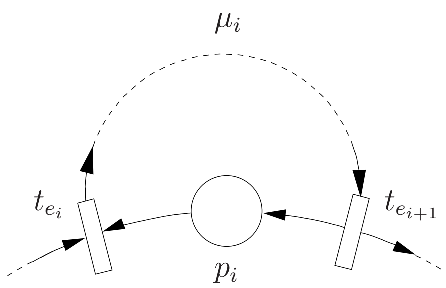


FIG. 4.10: Construction d'un circuit dans G

En posant

$$U_1 = \{i \in \{1, \ldots, q\} / p_i = (t_{e_i}, t_{e_{i+1}})\} \text{ et } U_2 = \{1, \ldots, q\} - U_1$$

nous parvenons à :

$$\begin{aligned} \log(W(C)) &= \sum_{i \in U_1} \log \tfrac{w(p_i)}{v(p_i)} + \sum_{i \in U_2} \log W(\mu_i) \\ &= \sum_{i \in U_1} \log \tfrac{\beta(t_{e_i}, p_i)}{\beta(t_{e_{i+1}}, p_i)} + \sum_{i \in U_2} \log \tfrac{\beta(t_{e_i}, p_i)}{\beta(t_{e_{i+1}}, p_i)} \\ &= -\beta(\mathcal{C}) \end{aligned}$$

Le graphe G étant unitaire, tout circuit C est de poids $W(C) = 1$ donc $\beta(\mathcal{C}) = 0$. □

Ce dernier théorème nous assure l'existence du vecteur $\alpha \in \{\mathbb{N}^{\star+}\}^{|P|}$ permettant de normaliser un graphe unitaire. En outre ce vecteur peut être calculé en temps polynomial en utilisant l'algorithme de BELLMAN-FORD [CLR90].

4.2.4 Normalisation et Expansion

Étonnamment, il existe une relation forte entre la normalisation et l'expansion d'un graphe d'événements unitaire. Cette relation se révèle être une deuxième démonstration de la normalisation des graphes unitaires.

Pour cela, nous rappelons une définition alternative de l'expansion proposée par Munier [Mun93] :

Définition 8. *Soit G un graphe d'événements généralisé. G est expansible s'il existe $(N_1, \dots, N_n) \in \{\mathbb{N}^*\}^n$ tel que :*

$$\forall p = (t_i, t_j) \in P, \quad \frac{N_i}{N_j} = \frac{v(p)}{w(p)}$$

Nous pouvons à présent énoncer le théorème suivant.

Théorème 6. *Soit G un graphe d'événements généralisé. G est expansible si et seulement si G est normalisable.*

Démonstration. $A \Rightarrow B$ Si G est expansible, alors il existe un vecteur $(N_1, \dots, N_n) \in \{\mathbb{N}^*\}^n$ tel que $\forall p = (t_i, t_j) \in P, \dfrac{N_i}{N_j} = \dfrac{v(p)}{w(p)}$. Nous posons alors $N = ppcm_{i \in \{1,\dots,n\}}(N_i)$, $\lambda = ppcm_{a \in \{1,\dots,m\}}(w(p_a), v(p_a))$ et $\forall i \in \{1, \dots, n\}, Z_i = \dfrac{N \cdot \lambda}{N_i}$. Pour toute place $p_a = (t_i, t_j)$, nous posons $\alpha_a = \dfrac{Z_i}{w(p_a)}$. Nous montrons alors que le vecteur α est une solution au système précédent :

1. Par définition de Z_i, $\alpha_a \in \mathbb{N}^\star$.
2. Pour tout couple $(p_a, p_b) \in \mathcal{P}^-(t_i) \times \mathcal{P}^+(t_i)$,

$$\alpha_b \cdot w(p_b) = Z_i = \frac{N \cdot \lambda}{N_i}$$

   Soit une transition t_j telle que $p_a = (t_j, t_i)$. Nous obtenons alors :

$$\frac{N \cdot \lambda}{N_i} = \frac{N \cdot \lambda}{N_j} \cdot \frac{v(p_a)}{w(p_a)} = \frac{Z_j}{w(p_a)} \cdot v(p_a) = \alpha_a \cdot v(p_a)$$

   Nous en déduisons que $\alpha_b \cdot w(p_b) = \alpha_a \cdot v(p_a)$.
3. Pour tout couple $(p_a, p_b) \in \mathcal{P}^+(t_i) \times \mathcal{P}^+(t_i)$,

$$\alpha_a \cdot w(p_a) = Z_i = \alpha_b \cdot w(p_b)$$

4. Pour tout couple $(p_a, p_b) \in \mathcal{P}^-(t_i) \times \mathcal{P}^-(t_i)$ avec $p_a = (t_j, t_i)$,

$$\alpha_a \cdot v(p_a) = \frac{Z_j}{w(p_a)} \cdot v(p_a) = \frac{N \cdot \lambda}{N_j} \cdot \frac{v(p_a)}{w(p_a)} = \frac{N \cdot \lambda}{N_i} = Z_i$$

d'où, $\alpha_a \cdot v(p_a) = Z_i = \alpha_b \cdot v(p_b)$.

$B \Rightarrow A$ Réciproquement, supposons que α est une solution du système précédent. Alors, pour tout entier $i \in \{1, \cdots n\}$ et $\forall (p_a, p_b) \in \mathcal{P}^-(t_i) \times \mathcal{P}^+(t_i)$, $\alpha_a \cdot v(p_a) = \alpha_b \cdot w(p_b) = Z_i$.

En posant $Z = ppcm_{i \in \{1,\ldots,n\}}(Z_i)$, nous montrons que $\forall i \in \{1, \ldots, n\}$, les valeurs $N_i = \frac{Z}{Z_i}$ vérifient les équations de l'expansion. En effet, pour toute place $p_a = (t_i, t_j) \in P$,

$$\frac{N_i}{N_j} = \frac{Z_j}{Z_i} = \frac{v(p_a)}{w(p_a)}$$

Ceci complète la preuve.

□

Il existe également un autre lien tout aussi fort entre l'expansion et la normalisation. Le corollaire suivant décrit la relation entre les valeurs des composantes des vecteurs d'expansion et de normalisation.

Corollaire 1. *Soit G un graphe d'événements généralisé fortement connexe. G est normalisable si et seulement si G est unitaire. De plus, si G est normalisé et pour tout $i \in \{1, \cdots, n\}$, Z_i désigne la valeur de normalisation de la transition t_i alors :*

$$N_i \cdot Z_i = K$$

Démonstration. Soit G un graphe d'événements généralisé fortement connexe. D'après [Mun93], G est unitaire si et seulement si G est expansible. D'après le théorème 6, nous obtenons la première partie du corollaire. Si G est normalisé, alors pour toute place $p = (t_i, t_j) \in P$, $Z_i = w(p)$ et $Z_j = v(p)$, d'où :

$$\frac{Z_i}{Z_j} = \frac{w(p)}{v(p)} = \frac{N_j}{N_i}$$

G étant fortement connexe, nous obtenons le résultat. □

4.2.5 Fin de l'exemple sur la normalisation

Reprenons le graphe d'événements généralisé $G = (T, P, M_0)$ considéré initialement (*voir* graphe de gauche sur la figure 4.11 page suivante). Nous rappelons dans un premier temps les équations de normalisation qui génère le système de contraintes de potentiel

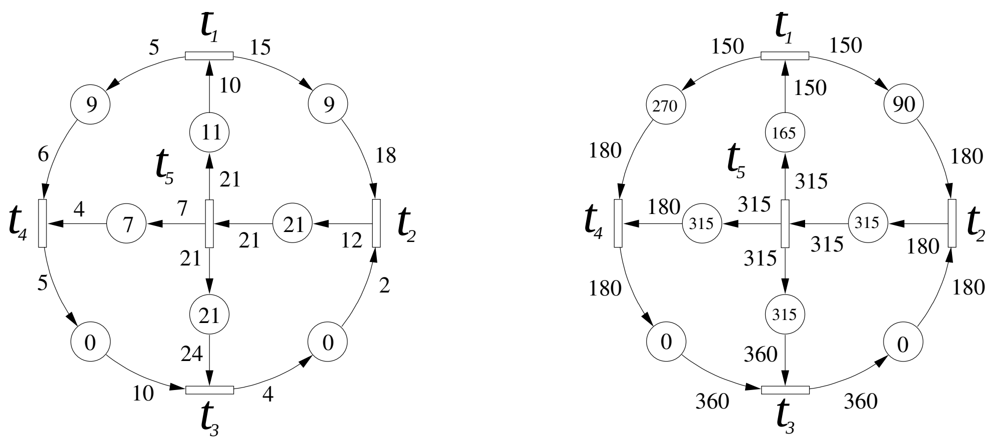


FIG. 4.11: Sur la gauche, un graphe d'événements généralisé unitaire G. Sur la droite, son graphe normalisé équivalent.

associé à ce graphe unitaire.

$$\left\{\begin{array}{lllllll} 15\cdot\alpha_{p_{1,2}} & = & 10\cdot\alpha_{p_{5,1}} & = & 5\cdot\alpha_{p_{1,4}} & & \\ 18\cdot\alpha_{p_{1,2}} & = & 12\cdot\alpha_{p_{2,5}} & = & 2\cdot\alpha_{p_{3,2}} & & \\ 4\cdot\alpha_{p_{3,2}} & = & 24\cdot\alpha_{p_{5,3}} & = & 10\cdot\alpha_{p_{4,3}} & & \\ 5\cdot\alpha_{p_{4,3}} & = & 4\cdot\alpha_{p_{5,4}} & = & 6\cdot\alpha_{p_{1,4}} & & \\ 21\cdot\alpha_{p_{2,5}} & = & 21\cdot\alpha_{p_{5,1}} & = & 21\cdot\alpha_{p_{5,3}} & = & 7\cdot\alpha_{p_{5,4}} \end{array}\right.$$

Une solution de ce système est donnée par $\alpha_{p_{1,2}} = 10$, $\alpha_{p_{1,4}} = 30$, $\alpha_{p_{2,5}} = 15$, $\alpha_{p_{3,2}} = 90$, $\alpha_{p_{4,3}} = 36$, $\alpha_{p_{5,1}} = 15$, $\alpha_{p_{5,3}} = 15$, et $\alpha_{p_{5,4}} = 45$. Le graphe d'événements généralisé décrit sur la droite de la figure 4.11 est le graphe normalisé associé aux valeurs précédemment fournies. Le vecteur de normalisation vaut ${}^tZ = (150, 180, 360, 180, 315)$. De même, nous écrivons le système d'équations associé à l'expansion de ce graphe :

$$\Gamma_G \cdot N = 0 \quad \text{avec} \quad \Gamma_G = \begin{array}{c} \\ p_{1,4} \\ p_{1,2} \\ p_{2,5} \\ p_{3,2} \\ p_{4,3} \\ p_{5,1} \\ p_{5,3} \\ p_{5,4} \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} 5 & 0 & 0 & -6 & 0 \\ 15 & -18 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 & -21 \\ 0 & -2 & 4 & 0 & 0 \\ 0 & 0 & -10 & 5 & 0 \\ -10 & 0 & 0 & 0 & 21 \\ 0 & 0 & -24 & 0 & 21 \\ 0 & 0 & 0 & -4 & 7 \end{pmatrix} \end{array}$$

On vérifie alors que le vecteur ${}^tN = (84, 70, 35, 70, 40)$ est une solution du système précédent et que par ailleurs pour tout $i \in \{1, \ldots, 5\}$, $N_i \cdot Z_i = 12600$. Désormais, nous considérons que les graphes unitaires sur lesquels nous travaillons sont normalisés.

4.3 Vivacité d'un graphe d'événements généralisé

Nous établissons dans un premier temps une condition suffisante de vivacité pour les GEG normalisés. Ensuite, nous montrons que cette condition est également nécessaire pour tout circuit normalisé à deux transitions. Enfin, nous proposons une propriété supplémentaire décrivant les marquages utiles.

4.3.1 Une condition suffisante de vivacité

Le lemme suivant est une conséquence directe de la normalisation d'un graphe unitaire. L'intérêt de la normalisation apparaît alors comme évident.

Lemme 7. *Soit G un graphe d'événements généralisé normalisé. Le nombre de jetons présents sur chaque circuit reste constant quelle que soit la séquence de franchissements considérée.*

Démonstration. Soit $C = (t_1, \ldots, t_q, t_1)$ un circuit de G et $\forall i \in \{1, \ldots, q\}$, ν_i désigne le nombre de franchissements de la transition t_i. Pour simplifier les équations, nous posons $\nu_{q+1} = \nu_1$. De même, nous posons $p_i = (t_i, t_{i+1})$ pour tout $i \in \{1, \ldots, q-1\}$ et $p_q = (t_q, t_1)$. Le marquage total du circuit à l'issue de ces franchissements vaut alors :

$$\sum_{i=1}^{q} M(p_i) = \sum_{i=1}^{q} (M_0(p_i) + \nu_i \cdot w(p_i) - \nu_{i+1} \cdot v(p_i))$$

Or G étant normalisé, $v(p_i) = w(p_{i+1})$ pour tout $i \in \{1, \ldots, q-1\}$ et $v(p_q) = w(p_1)$. Il s'ensuit que $\sum_{i=1}^{q} (\nu_i \cdot w(p_i) - \nu_{i+1} \cdot v(p_i)) = 0$. Le lemme est ainsi prouvé. □

Dans la littérature, l'étude de la vivacité est souvent abordée avec les P-semiflots. Pour le cas d'un graphe d'événements généralisé unitaire, un P-semiflot peut être utilisé pour construire un graphe d'événements généralisé équivalent pour lequel le nombre total de jetons est constant quelle que soit la séquence de franchissements considérée. Cependant, le graphe résultant de cette transformation n'est pas forcément normalisé et le lemme 7 ne peut donc s'appliquer.

Néanmoins, depuis les travaux pionniers de Karp [KM66], la plupart des auteurs ont montré la vivacité des circuits était une question centrale pour l'étude de la vivacité d'un système. Comme nous allons le voir dans le théorème suivant, la normalisation est un outil utile pour l'analyse de la vivacité et semble la simplifier.

Nous pouvons à présent énoncer une condition suffisante de vivacité basée sur le lemme précédent.

Théorème 7 (Condition suffisante). *Soit G un graphe d'événements généralisé normalisé. G est vivant si pour tout circuit C de G :*

$$\sum_{p \in C \cap P} M_0(p) > \sum_{p \in C \cap P} (v(p) - pgcd_p)$$

Démonstration. Cette preuve est obtenue par contradiction. Supposons que G ne soit pas vivant et que l'inégalité soit vérifiée pour tous les circuits C de G. Le graphe d'événements généralisé G étant non vivant, toute séquence de tirs valide conduit à une situation de blocage. A l'issue de chacune de ces séquences de tirs, il existe un circuit C_{Bl} dans G pour lequel aucune transition n'est franchissable par manque de jetons sur les places du circuit, soit :

$$\forall p \in C_{Bl} \cap P, M(p) < v(p)$$

Le nombre de jetons présents sur une place p pouvant être considéré comme un multiple du $pgcd_p$ (*cf.* lemme 3 page 50), nous obtenons :

$$\forall p \in C_{Bl} \cap P, M(p) \leq v(p) - pgcd_p$$

En additionnant toutes ces inégalités, nous avons :

$$\sum_{p \in C_{Bl} \cap P} M(p) \leq \sum_{p \in C_{Bl} \cap P} (v(p) - pgcd_p)$$

Le graphe d'événements généralisé G étant normalisé, d'après le lemme 7, $\sum_{p \in C_{Bl} \cap P} M(p) = \sum_{p \in C_{Bl} \cap P} M_0(p)$ soit une contradiction. □

Le membre de droite de cette condition suffisante s'interprète comme étant la somme des marquages du marquage bloquant le plus grand possible du circuit considéré. Or, d'après le lemme 7 page précédente, le nombre de jetons sur un graphe normalisé demeure constant sur chaque circuit quelque soit la séquence de franchissements considérée. Il suffit que la somme des marquages initiaux de l'ensemble des places d'un circuit donné soit plus grande que la somme des marquages de son plus grand marquage bloquant pour en déduire que ce circuit est vivant.

4.3.2 Condition nécessaire et suffisante de vivacité pour les circuits à deux transitions

La condition suffisante précédente s'avère être aussi nécessaire pour le cas des circuits à deux transitions.

Théorème 8 (Cas particulier). *Soit C un circuit normalisé composé de deux places p_1 et p_2 et de deux transitions distinctes (voir figure 4.12 page suivante). Le circuit C est vivant si et seulement si :*

$$M_0(p_1) + M_0(p_2) > v(p_1) + v(p_2) - 2 \cdot pgcd_{p_1}$$

Démonstration. Le circuit C étant normalisé, nous avons $w(p_1) = v(p_2)$ et $v(p_1) = w(p_2)$.

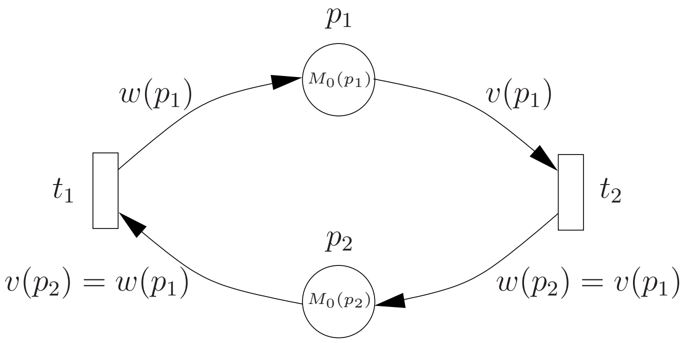


FIG. 4.12: Un graphe normalisé à deux places.

- Si $w(p_1) = w(p_2) = w$, alors, $pgcd_{p_1} = pgcd_{p_2} = w$. Dans ce cas, d'après le lemme 4 page 52 sur les places équivalentes, le circuit C est équivalent à un graphe d'événements ayant un marquage initial pour les places p_1 et p_2 respectivement égal à $\dfrac{M_0(p_1)}{w}$ et $\dfrac{M_0(p_2)}{w}$. Ce circuit est donc vivant si et seulement si

$$\frac{M_0(p_1)}{w} + \frac{M_0(p_2)}{w} > 0$$

  Les marquages initiaux $M_0(p_1)$ et $M_0(p_2)$ étant multiples de w, nous retrouvons la condition énoncée dans le théorème.

- Supposons à présent que $w(p_1) \neq w(p_2)$. Afin d'alléger nos notations, nous posons $p = p_1$ pour le reste de la preuve. Nous démontrons alors par un raisonnement par contradiction que la condition est également nécessaire. Supposons que l'inégalité ne soit pas vérifiée et que cependant le circuit C soit vivant. Nous définissons un état du circuit C comme étant un couple de marquages instantanés $(M(p_1), M(p_2))$. Soit Λ le nombre d'états distincts atteignables pour le circuit C et son marquage initial.

  (a) Nous supposons que le circuit C est vivant. Le nombre d'états atteignables étant borné, une séquence de franchissements contenant au moins un état deux fois peut être construite. Il existe alors deux entiers ν_1 et ν_2 tels que :

$$\left\{ \begin{array}{rclcrcl} M(p_1) + \nu_1 \cdot w(p) - \nu_2 \cdot v(p) & = & M(p_1) & \Longrightarrow & \nu_1 \cdot w(p) & = & \nu_2 \cdot v(p) \\ M(p_2) + \nu_2 \cdot v(p) - \nu_1 \cdot w(p) & = & M(p_2) & \Longrightarrow & \nu_2 \cdot v(p) & = & \nu_1 \cdot w(p) \end{array} \right.$$

  Les plus petites valeurs vérifiant ces conditions sont $\nu_1^* = \frac{ppcm_p}{w(p)}$ et $\nu_2^* = \frac{ppcm_p}{v(p)}$. Nous en déduisons que le nombre d'états Λ est borné par :

$$\Lambda \geq \frac{ppcm_p}{w(p)} + \frac{ppcm_p}{v(p)} - 1$$

  (b) D'autre part, nous avons supposé que le nombre total de jetons initiaux du circuit est inférieur ou égal à $\sigma = w(p)+v(p)-2 \cdot pgcd_p$. Selon le lemme 3 page 50, le nombre de jetons présents dans une place p est un multiple de $pgcd_p$. De plus, le nombre total de jetons présents dans un circuit normalisé demeurant constant, il est possible d'énumérer les couples d'entiers (X, Y) à composantes multiples de $pgcd_p$ et telles que $X + Y = \sigma$, soit :

$$\underbrace{\begin{array}{c|c|c|c|c|c} 0 & pgcd_p & 2 \cdot pgcd_p & \dots & \sigma - pgcd_p & \sigma \\ & & & & & \\ \sigma & \sigma - pgcd_p & \sigma - 2 \cdot pgcd_p & \dots & pgcd_p & 0 \end{array}}_{\text{Couple } (X,Y) \text{ avec } X+Y=\sigma}$$

$$\text{soit au total } \frac{\sigma}{pgcd_p} + 1 \text{ couples}$$

Il existe alors exactement $\frac{\sigma}{pgcd_p} + 1$ couples différents de marquages pouvant être construit à partir de σ jetons. Cependant, l'énumération précédente prend en considération le couple de valeurs $M(p_1) = v(p) - pgcd_p$ et $M(p_2) = v(p_2) - pgcd_p = w(p) - pgcd_p$ qui représente une situation de blocage du circuit. Nous en déduisons donc que le nombre Λ d'états vivants distincts pouvant être construit avec σ jetons vérifie :

$$\Lambda \leq \frac{\sigma}{pgcd_p} = \frac{w(p)}{pgcd_p} + \frac{v(p)}{pgcd_p} - 2$$

Néanmoins, nous avons :

$$v(p) \cdot w(p) = ppcm_p \cdot pgcd_p$$

Nous en déduisons que :

$$\frac{w(p)}{pgcd_p} = \frac{ppcm_p}{v(p)} \quad \text{et} \quad \frac{v(p)}{pgcd_p} = \frac{ppcm_p}{w(p)}$$

Il s'ensuit alors :

$$\frac{w(p)}{pgcd_p} + \frac{v(p)}{pgcd_p} - 1 \leq \Lambda \leq \frac{w(p)}{pgcd_p} + \frac{v(p)}{pgcd_p} - 2$$

d'où une contradiction.

□

Il est donc possible de tester en temps polynomial la vivacité d'un circuit à deux transitions. Ce dernier résultat s'avère très utile pour la résolution du problème étudié au prochain chapitre. Par ailleurs, nous allons pouvoir caractériser plus en détails la notion de jetons utiles présentée au début de ce chapitre.

4.3.3 Propriété des jetons utiles

La condition nécessaire et suffisante de vivacité du théorème 8 a pour conséquence de préciser la notion de jetons utiles évoquée dans la section 4.1 de ce chapitre. En effet, le théorème suivant caractérise le nombre maximum de jetons $R(p)$ pouvant être supprimés de la place p sans modifier la structure des contraintes de précédence générée par la place p. Dans [Sau03], Sauer présente un algorithme de complexité pseudo-polynomiale pour calculer $R(p)$. Le théorème suivant fournit une évaluation simple de $R(p)$ pouvant être calculée en temps constant.

Théorème 9. *Soient* $p = (t_i, t_j)$ *une place de marquage initial* $M_0(p)$ *et* $R_{pgcd_p}(M_0(p))$ *le reste de la division euclidienne de* $M_0(p)$ *par* $pgcd_p$. *Alors* $R(p) = R_{pgcd_p}(M_0(p))$.

Démonstration. Selon le lemme 3 sur les jetons utiles, nous avons $R(p) \geq R_{pgcd_p}(M_0(p))$. Nous prouvons que $R(p) \leq R_{pgcd_p}(M_0(p))$.

En effet, supposons que les places $p_1 = (t_i, t_j)$ et $p_2 = (t_i, t_j)$ avec $w(p_1) = w(p_2)$ et $v(p_1) = v(p_2)$ induisent les mêmes contraintes de précédence et que $M_0(p_1) > M_0(p_2) + R_{pgcd_{p_1}}(M_0(p_1))$. D'après le lemme 3 sur les jetons utiles, nous pouvons supposer que $M_0(p_1) = k_1 \cdot pgcd_{p_1}$ et $M_0(p_2) = k_2 \cdot pgcd_{p_2}$ avec $(k_1, k_2) \in \mathbb{N} \times \mathbb{N}$. Nous en déduisons que $R_{pgcd_{p_1}}(M_0(p_1)) = 0$ et $k_1 > k_2$.

Alors d'après le lemme 2, les marquages initiaux $M_0'(p_1) = (k_1 - k_2) \cdot pgcd_{p_1}$ et $M_0'(p_2) = 0$ pour les places p_1 et p_2 induisent également les mêmes contraintes de précédence entre les différentes occurrences des transitions adjacentes. On construit alors une place $p_3 = (t_j, t_i)$ telle que $w(p_3) = v(p_1)$, $v(p_3) = w(p_1)$ et $M_0'(p_3) = v(p_1) + w(p_1) - 2 \cdot pgcd_{p_1}$ afin d'obtenir deux circuits à deux transitions (*voir* figure 4.13 ci-dessus).

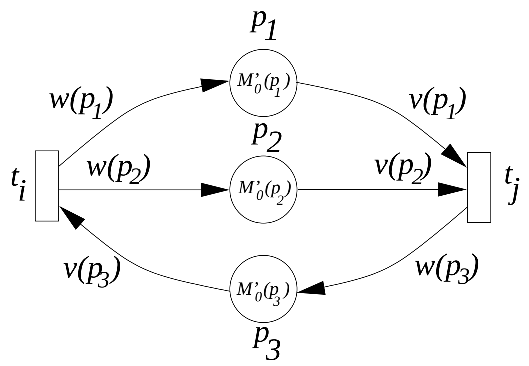


FIG. 4.13: Les places p_1, p_2 et p_3

D'après la condition nécessaire et suffisante de vivacité énoncée au théorème 8, le circuit $(t_i, p_1, t_j, p_3, t_i)$ est vivant alors que le circuit $(t_i, p_2, t_j, p_3, t_i)$ ne l'est pas. Nous en déduisons alors que les places p_1 et p_2 de marquage initial respectif $M_0(p_1)$ et $M_0(p_2)$ n'induisent pas les mêmes contraintes de précédence. □

4.4 Un algorithme polynomial pour la condition suffisante

Nous présentons dans cette dernière section un contre-exemple à la condition de vivacité puis nous détaillons un algorithme permettant de tester efficacement cette condition sur tout graphe unitaire.

4.4.1 Un contre-exemple pour la condition suffisante de vivacité

Dans le cas général, la condition suffisante de vivacité présentée au théorème 7 n'est pas nécessaire : en effet, considérons pour cela le graphe d'événements normalisé représenté sur la figure 4.14. La séquence de franchissements s décrite en détails dans le tableau 4.1 peut être répétée infiniment souvent. Le système est alors vivant et néanmoins la condition suffisante du théorème 7 n'est pas vérifiée car la somme des marquages initiaux de ce système vaut $\sum_{i=1}^{3} M_0(p_i) = 28$ et $\sum_{i=1}^{3}(v(p_i) - pgcd_{p_i}) = 29$.

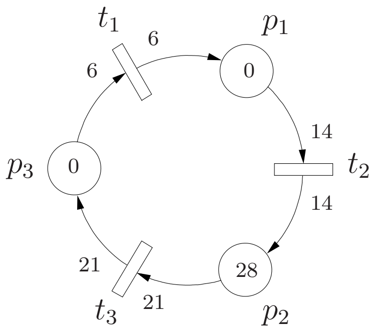


Fig. 4.14: Le graphe d'événements généralisé G est vivant alors que la condition suffisante du théorème 7 n'est pas vérifiée.

			t_3		$3.t_1$		t_2		t_3		$4.t_1$		$2.t_2$	
$M(p_1)$	:	0		0		18		4		4		28		0
$M(p_2)$	:	28		7		7		21		0		0		28
$M(p_3)$	:	0		21		3		3		24		0		0

Tab. 4.1: La séquence de franchissements $s = t_3t_1t_1t_1t_2t_3t_1t_1t_1t_1t_2t_2$ pour l'exemple de la figure 4.14 est valide.

4.4.2 Un algorithme efficace pour tester la condition suffisante de vivacité

La vérification algorithmique de la condition suffisante de vivacité se fait en $\mathcal{O}(n \cdot m)$:

- L'étape de normalisation du graphe unitaire se fait en $\mathcal{O}(n \cdot m)$ par l'algorithme de Bellman-Ford.
- On construit ensuite un graphe valué $\mathcal{G}' = (T, V)$ tel que pour toute place du graphe normalisé $p' = (t_i, t_j) \in P'$ est associée à un arc $(i, j) \in V$ valué par $v_{ij} = M_0(p') - v(p') + pgcd_{p'}$ (*voir* la sous-section suivante pour la suite de l'exemple de la page 56).

  A l'aide de l'algorithme de Bellman-Ford, nous vérifions l'existence d'un vecteur Δ à n composantes qui satisfait à :

$$\forall (i,j) \in V, \Delta_j - \Delta_i \leq v_{ij}$$

  (a) Si aucun vecteur Δ n'existe, alors il existe un circuit C tel que $\sum_{(i,j)\in C} v_{(i,j)} < 0$. Nous en déduisons que le circuit C ne vérifie pas la condition suffisante du théorème 7 . Dans ce cas, nous ne pouvons conclure pour la vivacité du graphe unitaire G.

  (b) Dans le cas contraire, nous construisons un sous-graphe partiel $\mathcal{G}''$ de $\mathcal{G}'$ en supprimant tous les arcs non critiques de V *i.e.* les arcs tels que $\Delta_j - \Delta_i < v_{ij}$.

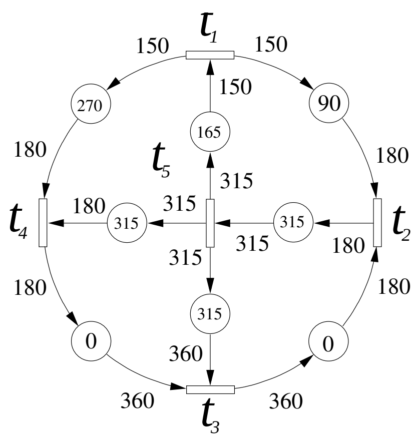
    i. Alors, si $\mathcal{G}''$ est acyclique, la condition du théorème 7 est satisfaite et nous en déduisons que le graphe d'événements généralisé G est vivant.

    ii. Dans le cas contraire nous ne pouvons rien conclure.

  La présence de circuits dans le graphe $\mathcal{G}''$ peut être testée en $\mathcal{O}(m)$ avec l'algorithme de Recherche en profondeur.

4.4.3 Exemple pour la vivacité

Reprenons le graphe d'événements généralisé obtenu après sa normalisation. Nous lui associons un graphe orienté valué $\mathcal{G}'$. Ce graphe n'ayant pas de circuit de valeur négative, nous pouvons affirmer que G est vivant.



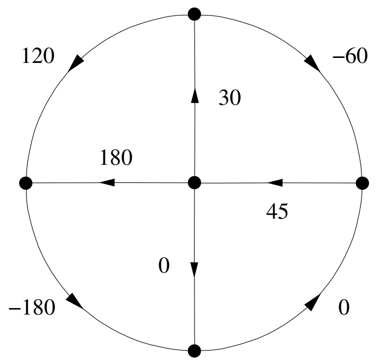


Fig. 4.15: A gauche le graphe unitaire normalisé. On présente à droite le graphe $\mathcal{G}'$ associé.

Conclusion

Nous avons introduit dans ce chapitre un ensemble de transformations pour les graphes d'événements généralisés permettant de faciliter l'étude de la vivacité.

Tout d'abord, nous avons décrit deux résultats permettant de comparer les ensembles de contraintes de précédence induites par une place et son marquage initial. La notion de places équivalentes permet de transformer une place marquée en une autre place marquée tout en préservant les contraintes de précédence générées par cette place. Nous avons également réduit la combinatoire des marquages à prendre en considération en introduisant les notions de jetons utiles et de marquages utiles. La notion de jetons utiles permet ainsi de restreindre l'ensemble des marquages à considérer conférant ainsi une certaine granularité des marquages possibles du système.

Nous avons aussi défini un outil important pour aborder les problèmes d'optimisation sur les graphes d'événements généralisés : la normalisation. La normalisation est un outil permettant d'obtenir un graphe équivalent dont le nombre total de jetons par circuit est un invariant. En outre, la normalisation semble simplifier la définition mathématique des problèmes sur ce modèle.

En combinant ces résultats, nous aboutissons à l'élaboration d'une condition suffisante de vivacité vérifiable en temps polynomial. Les complexités en temps de calcul des méthodes existantes pour vérifier la vivacité rendent ces méthodes inefficaces sur des instances de grande taille. Notre algorithme polynomial permet donc de tester rapidement la condition suffisante de vivacité pour des instances quelconques. Enfin, nous avons démontré que cette condition suffisante s'avère également nécessaire pour le cas des circuits à deux transitions.

Chapitre 5

Définition et résolution du problème de marquage

Sommaire

Ce chapitre est consacré à la résolution du *problème de marquage*. Ce problème consiste à déterminer un marquage initial et des capacités pour chaque place tels que la somme des capacités soit minimum et que le système soit vivant tout en respectant les contraintes de capacités des places.

Dans le problème de marquage, le concepteur a toute latitude pour définir les conditions initiales (*i.e.* le marquage initial) de l'application embarquée étudiée. Certains auteurs [ČP93, MBL97, BML99] ont considéré le problème qui consiste à déterminer les tailles des mémoires d'une application dont les conditions initiales sont une donnée du problème. Dans [Mur96], Murthy démontre que ce problème est NP-complet même pour le cas des graphes d'événements. Les résolutions algorithmiques de ce problème reposent alors sur des heuristiques [ALP94, Adé96, GGD02]. Cependant, dans le cas des graphes d'événements généralisés, Adé [Adé96] note qu'il est parfois nécessaire au concepteur de modifier sensiblement ces conditions initiales afin de pouvoir s'assurer de la vivacité du système donné. Les difficultés rencontrées dans l'analyse de la vivacité des graphes d'événements généralisés justifient en quelque sorte l'intérêt du problème de marquage.

Dans le contexte de la conception d'ateliers, les problèmes de minimisation des marquages initiaux vérifiant certaines propriétés sont des sujets de recherche classiques [HP89, LPX92, DFMS97, GPS02, Sau03]. Les transitions représentent alors les ateliers, les jetons modélisent les produits en cours de fabrication et les places désignent les zones de stockage de ces produits entre deux transformations successives. Dans ce domaine, le problème de marquage consiste à déterminer les conditions initiales des produits sur la chaîne de production de sorte à minimiser la taille de l'ensemble des zones de stockage.

Nous montrons dans ce chapitre qu'il existe un algorithme polynomial pour le problème de marquage. Le chapitre est organisé comme suit : la section 5.1 pose formellement le problème de marquage. Dans la section 5.2, une série de résultats préliminaires sont établis afin de caractériser précisément les graphes d'événements pour lesquels le problème de marquage est pertinent. Nous déterminons également dans cette section la capacité minimum à allouer à une place. Nous procédons dans la section 5.3 à une simplification du problème de marquage indispensable à la méthode de résolution que nous proposons. Les deux dernières sections de ce chapitre sont axées sur la mise au point d'algorithmes polynomiaux pour ce problème.

5.1 Présentation et définition du problème de marquage

Cette section a pour objectif de proposer une définition formelle du problème de marquage.

5.1.1 Définitions

Au chapitre 3, nous avons présenté une définition des réseaux de Petri K-bornés (*voir* définition 3 page 33). Nous introduisons à présent une définition légèrement différente pour laquelle l'existence d'au moins une séquence de franchissements valide suffit. Nous présentons d'abord la notion de place $M^\star$-bornée.

Définition 9. *Une place $p \in P$ est dite $M^\star$-bornée s'il existe une séquence de franchissements de taille infinie pour laquelle le nombre de jetons contenus dans la place p reste borné par $M^\star(p) \in \mathbb{N}$. $M^\star(p)$ est appelé la capacité de la place p.*

Nous définissons de façon analogue les graphes bornés :

Définition 10. *Soit $G = (T, P)$ un graphe d'événements généralisé. G est un graphe borné si $\forall p \in P$, il existe un marquage initial M_0 et une capacité $M^\star$ tels que :*

- *le graphe d'événements généralisé marqué $G' = (T, P, M_0)$ est vivant,*
- *pour toute place $p \in P$, le nombre de jetons contenus dans la place p reste borné par $M^\star(p)$.*

On remarque alors que notre définition des graphes bornés est distincte de celle des réseaux de Petri K-bornés. En effet, dans notre modèle la contrainte de capacité des places est une contrainte imposée alors que dans le modèle des places K-bornées, le fait qu'un graphe soit K-borné est avant toutes choses une propriété structurelle et comportementale inhérente au système.

5.1.2 Formulation du problème de marquage

Le problème peut être formalisé comme suit :

Soit $G = (T, P)$ un graphe borné et $f : \mathbb{N}^m \to \mathbb{N}$ une fonction croissante. Le problème consiste à déterminer un marquage initial M_0 et des capacités $M^\star$ pour chacune des places tels que :

- le graphe d'événements généralisé marqué $G' = (T, P, M_0)$ est vivant même lorsque l'on impose que le nombre de jetons d'une place p doit rester borné par $M^\star(p)$.
- $f(M^\star(p_1), \cdots, M^\star(p_m))$ est minimum.

Nous avons introduit volontairement dans la définition de ce problème une distinction entre le graphe d'événements (noté G) en tant que structure et le graphe d'événements qui est associé à un marquage initial donné (noté G'). Cette distinction sera faite tout au long de ce chapitre afin de dégager clairement les aspects structurels des aspects comportementaux de ces systèmes.

5.2 Résultats préliminaires

Comme nous l'avons vu précédemment, les graphes bornés forment une sous-classe des graphes d'événements généralisés pour laquelle la capacité des places doit être bornée. L'objectif de cette section est de fournir une caractérisation de cette sous-classe. Nous présentons également une borne inférieure de la capacité à allouer à une place.

Dans un premier temps, nous démontrons que les boucles du graphe d'événements généralisé peuvent être négligées. Par la suite, nous démontrons que la capacité d'une place peut être modélisée par un couple de places ayant un marquage initial vérifiant une condition précise. Nous montrons alors qu'on peut associer à chaque graphe borné un graphe d'événements généralisé appelé graphe à capacité qui modélise ces contraintes de capacité. Nous exposons ensuite une borne inférieure de la capacité d'une place. Enfin, nous donnons une caractérisation structurelle des graphes bornés.

5.2.1 Elimination des boucles

Définition 11. *Une place $p = (t_i, t_i)$ est appelée boucle (voir figure 5.1).*

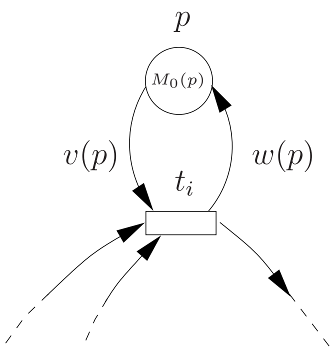


FIG. 5.1: Exemple d'une place boucle $p = (t_i, t_i)$.

Les boucles d'un graphe d'événements généralisé peuvent être supprimées : en effet, comme les systèmes étudiés ici ont une structure de graphe d'événements, le nombre de jetons présents dans une boucle $p = (t_i, t_i)$ dépend uniquement du nombre de franchissements de la transition t_i. Or, la transition t_i doit pouvoir être franchie infiniment souvent tout en gardant un marquage borné sur toutes les places du graphe. Il est facile de voir que $w(p) \geq v(p)$ est une condition nécessaire de vivacité pour la transition t_i et que $w(p) \leq v(p)$ est une condition nécessaire pour le marquage de la place p reste borné. Il s'ensuit alors que $w(p) = v(p)$. Cette dernière égalité implique que le nombre de jetons présents sur la place p est toujours égal à $M_0(p)$ et par conséquent $M_0^\star(p) = M_0(p)$.

De la même manière, $M_0(p) \geq v(p)$ est une condition nécessaire de vivacité pour la transition t_i. Nous en déduisons que si toutes les conditions précédentes sont satisfaites alors une place boucle n'inhibe aucun franchissement de la transition t_i. Le nombre maximum de jetons présents dans une place boucle p étant toujours égal à son marquage initial, nous pouvons poser $M^\star(p) = v(p)$.

Ces conditions devant être toutes vérifiées par les places boucles, nous pouvons limiter notre étude sans perte de généralité aux graphes d'événements généralisés dépourvus de place boucle.

5.2.2 Modélisation de la contrainte de capacité d'une place

Nous démontrons dans cette sous-section qu'une place $M^\star$-bornée d'un graphe d'événements généralisé G peut être modélisée par un couple de places n'ayant pas de contrainte de capacité. Nous exprimons ainsi la contrainte de capacité d'une place en utilisant le formalisme des graphes d'événements généralisé.

Nous rappelons dans un premier temps le résultat sur la notion de contrainte de précédence induite par une place marquée définie au chapitre 3. Soit une place $p = (t_i, t_j) \in P$ et un couple d'entiers $(\nu_i, \nu_j) \in \mathbb{N}^\star \times \mathbb{N}^\star$. Il existe une contrainte de précédence entre les occurrences de franchissements $< t_i, \nu_i >$ et $< t_j, \nu_j >$ si et seulement si :

Condition 1 $< t_j, \nu_j >$ peut être franchie avant $< t_i, \nu_i >$;

Condition 2 $< t_j, \nu_j - 1 >$ peut être franchie avant $< t_i, \nu_i >$ alors que $< t_j, \nu_j >$ ne peut pas (par manque de jetons sur la place).

Munier [Mun93] démontre alors le lemme suivant permettant de caractériser précisément la notion de contraintes de précédence entre les différents franchissements des transitions adjacentes d'une place :

Lemme 1 [Rappel]. *Soit une place $p = (t_i, t_j) \in P$ de marquage initial $M_0(p)$. La place p et son marquage initial $M_0(p)$ induisent une contrainte de précédence entre le ν_i-ième franchissement de la transition t_i et le ν_j-ième franchissement de la transition t_j si et seulement si :*

$$w(p) > M_0(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j \geq \max(w(p) - v(p), 0)$$

Nous pouvons transposer ce raisonnement pour décrire la notion de contrainte de précédence induite par la contrainte de capacité d'une place. Soit une place $p = (t_i, t_j) \in P$ ayant une contrainte de capacité. Il existe une contrainte de précédence entre les occurrences de transitions $< t_j, \nu_j >$ et $< t_i, \nu_i >$ due à la contrainte de capacité sur la place $p = (t_i, t_j)$ si et seulement si :

Condition 3 $< t_i, \nu_i >$ peut être franchie avant $< t_j, \nu_j >$;

Condition 4 $< t_i, \nu_i - 1 >$ peut être franchie avant $< t_j, \nu_j >$ alors que $< t_i, \nu_i >$ ne peut pas (par manque d'espace libre sur la place).

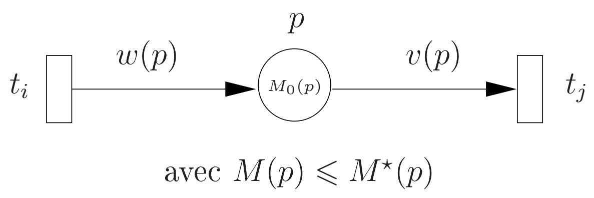


Fig. 5.2: Représentation graphique d'une place $p = (t_i, t_j)$ étant $M^\star$-bornée.

Lemme 8. *Soit une place $p = (t_i, t_j) \in P$ de marquage initial $M_0(p)$. La limitation par un entier $M^\star(p) \geq M_0(p)$ de la capacité de la place $p = (t_i, t_j) \in P$ induit une contrainte de précédence entre ν_j-ième franchissement de la transition t_j et le ν_i-ième franchissement de la transition t_i si et seulement si :*

$$v(p) > (M^\star(p) - M_0(p)) + v(p) \cdot \nu_j - w(p) \cdot \nu_i \geq \max(v(p) - w(p), 0)$$

Démonstration. Une place $p = (t_i, t_j) \in P$ de capacité bornée par $M^\star(p)$ et de marquage initial $M_0(p)$ modélise une contrainte de précédence entre $< t_j, \nu_j >$ et $< t_i, \nu_i >$ si et seulement si les conditions 3 et 4 sont vérifiées.

1. La condition 3 est équivalente à

$$M^\star(p) \geq M_0(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j$$

Nous obtenons alors :

$$(M^\star(p) - M_0(p)) + v(p) \cdot \nu_j - w(p) \cdot \nu_i \geq 0$$

2. De même, la condition 4 est équivalente à

$$M^\star(p) \geq M_0(p) + w(p) \cdot (\nu_i - 1) - v(p) \cdot (\nu_j - 1) > M^\star(p) - w(p)$$

soit :

$$M^\star(p) + w(p) - v(p) \geq M_0(p) + w(p) \cdot \nu_i - v(p) \cdot \nu_j > M^\star(p) - v(p)$$

d'où,

$$v(p) > (M^\star(p) - M_0(p)) + v(p) \cdot \nu_j - w(p) \cdot \nu_i \geq v(p) - w(p)$$

En combinant ces deux inégalités, nous aboutissons au lemme. □

Nous en déduisons immédiatement le théorème suivant :

Théorème 10. *Soit $p = (t_i, t_j)$ une place $M^\star$-bornée de marquage initial $M_0(p) \leq M^\star(p)$. Alors cette place peut être remplacée par deux places $p_1 = (t_i, t_j)$ et $p_2 = (t_j, t_i)$ telles que $w(p_1) = v(p_2) = w(p)$, $v(p_1) = w(p_2) = v(p)$ et de marquages initiaux $M_0^\star(p_1) = M_0(p)$ et $M_0^\star(p_2) = M^\star(p) - M_0(p)$ (voir figure 5.3 page ci-contre).*

Démonstration. D'après le lemme 1 page 35 rappelé précédemment, les contraintes de précédence entre les franchissements de t_i et t_j sont modélisées par la place p_1 avec un marquage initial $M_0^\star(p_1) = M_0(p)$. Selon le lemme 8, les contraintes de précédence induites par la capacité finie de la place p peuvent être modélisées par l'ajout d'une place $p_2 = (t_j, t_i)$ ayant un marquage initial $M_0^\star(p_2) = M^\star(p) - M_0(p)$. Le théorème est alors prouvé. □

Par la suite, les places p_1 et p_2 sont appelées les places associées de p et nous dirons que p_1 (*resp.* p_2) est la place retour de la place p_2 (*resp.* p_1).

Remarque 3. *Les places p_1 et p_2 forment un circuit unitaire : $W(p_1) \cdot W(p_2) = 1$.*

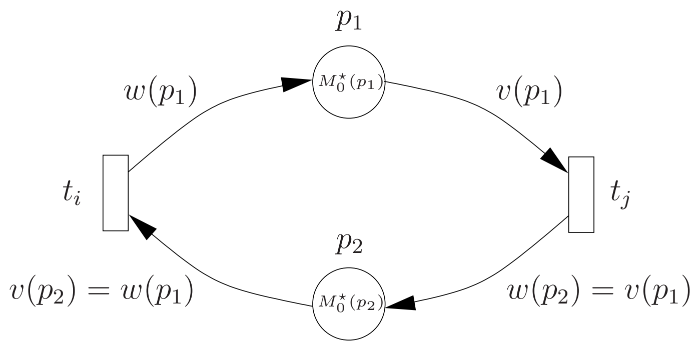


FIG. 5.3: Les places $p_1 = (t_i, t_j)$ et $p_2 = (t_j, t_i)$ associées à une place $M^\star$-bornée $p = (t_1, t_2)$ de marquage initial $M_0^\star(p)$.

5.2.3 Graphes d'événements généralisés à capacité

En utilisant la transformation suggérée dans le théorème 10, nous définissons dans cette sous-section le graphe d'événements généralisé à capacité associé. Nous en déduisons une propriété structurelle sur les graphes bornés.

Définition 12. *Soit $G = (T, P)$ un graphe borné. Le graphe à capacité associé $G_R = (T, P_R)$ est un graphe d'événements généralisé obtenu à partir de G en remplaçant chaque place $p \in P$ par deux places $(p_1, p_2) \in P_R \times P_R$ conformément au théorème 10.*

Nous désignons par P_R^1 (resp. P_R^2) l'ensemble des places $p = (t_i, t_j) \in P_R$ telles que la place correspondante dans P est définie depuis t_i vers t_j (resp. depuis t_j vers t_i). Nous avons alors $P_R^1 = P$ et $P_R^2 = P_R - P_R^1$.

Cette définition implique qu'il existe une correspondance entre tout marquage initial $M_0^\star$ de G_R et le marquage initial M_0 couplé aux capacités des places $M^\star$ dans G. De plus, d'après le théorème 10, les contraintes de précédence induites par ces deux représentations sont équivalentes. Nous en déduisons la conséquence suivante :

Conséquence 1. *Soient G un graphe d'événements généralisé et $G_R = (T, P_R)$ son graphe à capacité associé. $G = (T, P)$ est un graphe borné si et seulement s'il existe un marquage initial $M_0^\star$ tel que le graphe marqué $G'_R = (T, P_R, M_0^\star)$ soit vivant.*

Chaque place originelle de G étant remplacée par un circuit à deux places, nous formulons la remarque suivante :

Remarque 4. *Le graphe à capacité associé G_R d'un graphe G est fortement connexe.*

Muni de cette représentation des graphes bornés en graphes à capacité associés, nous allons à présent dégager quelques propriétés structurelles sur les graphes bornés.

5.2.4 Caractérisations des graphes bornés

Dans [Mun93], Munier montre que le nombre total de jetons d'un graphe d'événements généralisé G fortement connexe vivant est borné si et seulement si G est unitaire. Nous allons alors établir une propriété structurelle des graphes bornés en considérant les graphes à capacité.

Théorème 11. *Soient $G = (T, P)$ un graphe d'événements généralisé et $G_R = (T, P_R)$ son graphe à capacité associé. Si G est un graphe borné alors G_R est unitaire.*

Démonstration. La preuve est obtenue par un raisonnement par contradiction. Supposons que G soit un graphe borné et que G_R ne soit pas unitaire. D'après la conséquence 1, il existe un marquage initial $M_0^\star$ tel que $G'_R = (T, P, M_0^\star)$ soit vivant. Selon [Mun93], il s'ensuit que tout circuit C de G_R vérifie $W(C) \geq 1$. Or, G_R étant supposé non-unitaire, il existe un circuit C de G_R tel que $W(C) > 1$. L'ensemble de ses places retours forme également un circuit noté C' dans G_R. Par la remarque 3, nous en déduisons que $W(C) \cdot W(C') = 1$. Nous obtenons alors que $W(C') < 1$ et donc par conséquent G'_R n'est pas vivant, soit une contradiction. □

Ce résultat structurel peut se reformuler ainsi : G est un graphe borné si tout les chemins potentiels reliant un couple de transitions t_i et t_j sont de même poids. Intuitivement, pour qu'un graphe d'événements généralisé soit borné il faut que sa structure respecte un certain équilibre entre les transitions.

Selon ce dernier théorème et d'après la remarque 4, si G est un graphe borné alors son graphe associé à capacité G_R est normalisable. Nous en déduisons le corollaire suivant :

Corollaire 2. *Tout graphe borné G est normalisable.*

Dans la suite de ce chapitre, nous supposerons que tous les graphes bornés considérés seront normalisés. Nous pouvons donner une dernière caractérisation des graphes bornés.

Théorème 12. *Soit $G = (T, P)$ un graphe d'événements généralisé et $G_R = (T, P_R)$ son graphe à capacité associé. G est borné si et seulement si $G_R = (T, P_R)$ est unitaire.*

Démonstration. De par le théorème 11, si G est borné alors G_R est unitaire.

Réciproquement, supposons que G_R soit unitaire. Alors G_R peut être normalisé. Pour toute place $p \in P_R$, on pose $M_0^\star(p) = v(p)$ et $G'_R = (T, P_R, M_0^\star)$. Tout circuit C de G'_R satisfait à la condition suffisante établie par le théorème 7 page 63. Ceci implique que G'_R est vivant et par la conséquence 1, nous en déduisons que G est borné. □

5.2.5 Borne inférieure de la capacité d'une place

Nous exposons dans cette sous-section un borne inférieure de la capacité d'une place dans un graphe d'événements généralisé borné G. Nous démontrons formellement la borne inférieure de la capacité d'une place.

Soit $G = (T, P)$ un graphe d'événements borné. Pour toute place $p \in P$, on pose $M^\star_{min}(p) = w(p) + v(p) - pgcd_p$. Nous démontrons que $M^\star_{min}(p)$ est une borne inférieure de la capacité de la place p :

Théorème 13. *Pour toute place p $M^\star$-bornée,* $M^\star(p) \geq M^\star_{min}(p)$

Démonstration. D'après le théorème 10, la place p peut être remplacée par un circuit à deux places p_1 et p_2 avec des marquages initiaux respectivement égaux à $M_0^\star(p_1)$ et $M_0^\star(p_2)$. De plus, selon le théorème 8 établissant la condition nécessaire et suffisante de vivacité pour un circuit à deux transitions, nous obtenons l'inégalité suivante :

$$M_0^\star(p_1) + M_0^\star(p_2) > v(p_1) + v(p_2) - 2 \cdot pgcd_{p_1}$$

Par définition de p_1 et p_2, nous avons :

$$v(p_1) = w(p_2),\ \ w(p_1) = v(p_2)$$
$$\text{et} \quad pgcd_{p_1} = pgcd_{p_2}$$

Alors conformément à la notion de jetons utiles d'une place (*voir* lemme 3 page 50), les nombres de jetons présents dans les places p_1 et p_2 sont tous deux multiples de $pgcd_{p_1}$. Nous pouvons alors améliorer l'inégalité précédente comme suit :

$$M_0^\star(p_1) + M_0^\star(p_2) \geq v(p_1) + v(p_2) - pgcd_{p_1}$$

Par définition de p_1 et p_2, nous avons aussi $M^\star(p) = M_0^\star(p_1) + M_0^\star(p_2)$ et $v(p_1) = v(p)$, $v(p_2) = w(p)$ et $pgcd_{p_1} = pgcd_{p_2} = pgcd_p$. Nous en déduisons alors :

$$M^\star(p) \geq w(p) + v(p) - pgcd_p$$

En posant $M^\star_{min}(p) = w(p) + v(p) - pgcd_p$, nous obtenons le théorème. □

On notera que ce résultat a été obtenu également par Adé [Adé96] et Murthy [Mur96]. Cependant, l'importance des jetons utiles est mise en lumière dans notre démonstration. De plus, la suppression des contraintes de capacité des places par l'ajout de couples de places retours sans capacité permet d'aborder de façon simple le problème de marquage. L'analyse de la vivacité de ces systèmes est aussi grandement simplifiée par l'introduction de la normalisation.

5.3 Restriction et simplification du problème

Dans cette section, nous caractérisons des valeurs pertinentes pour les marquages initiaux des places des graphes à capacité. Cette caractérisation aboutit à une restriction importante sur l'ensemble des solutions du problème initial.

5.3.1 Graphe à capacité minimum

Le théorème 13 nous permet de déterminer pour chaque place $p \in P$ la capacité minimum $M^\star(p)$ à allouer. Nous allons montrer que le problème de marquage admet une solution lorsque l'on impose pour toutes places $p \in P$ que $M^\star(p) = M^\star_{min}(p)$. On définit dans un premier temps le graphe à capacité minimum.

Définition 13. *Soient $G = (T, P)$ un graphe d'événements généralisé borné, $G_R = (T, P_R)$ son graphe à capacité associé et $M_0^\star$ un marquage initial de G_R. $G_R^{min} = (T, P_R, M_0^\star)$ est un graphe à capacité minimum si pour tout couple de places $(p_1, p_2) \in P_R \times P_R$ associé à la place $p \in P$, $M_0^\star(p_1) + M_0^\star(p_2) = M^\star_{min}(p)$.*

On notera qu'un graphe à capacité minimum est étroitement lié à son marquage initial.

5.3.2 Elimination des places parallèles

Afin de simplifier davantage le problème de marquage, nous allons procéder à l'élimination des places parallèles des graphes bornés sur lesquels nous allons travailler.

Définition 14. *Soit G un graphe d'événements généralisé. Deux places $p = (t_i, t_j)$ et $q = (t_i, t_j)$ sont dites parallèles. Un graphe d'événements généralisé est dit générique s'il ne comporte aucune place parallèle.*

Nous démontrons que les places parallèles peuvent être retirées sans conséquence pour l'étude des graphes à capacités minimum.

Lemme 9. *Soient $G = (T, P)$ un graphe borné et $G_R^{min} = (T, P_R, M_0^\star)$ un graphe à capacité minimum. Si G_R^{min} est vivant, alors pour tout couple $(p, q) \in P_R \times P_R$ de places parallèles, $M_0^\star(p) = M_0^\star(q)$.*

Démonstration. Cette preuve est obtenue par un raisonnement par contradiction. Supposons que G_R^{min} soit vivant et qu'il existe deux places parallèles $p = (t_i, t_j)$ et $q = (t_i, t_j)$ dans G_R^{min} telles que $M_0^\star(p) > M_0^\star(q)$. Il s'ensuit que G_R^{min} comporte une place retour p' (*resp.* q') associée à la place p (*resp.* q) avec $M_0^\star(p') = M^\star_{min}(p) - M_0^\star(p)$ (*resp.* $M_0^\star(q') = M^\star_{min}(q) - M_0^\star(q)$). Le graphe à capacité minimum G_R^{min} étant normalisé, nous avons :

$$w(p) = w(q) = v(p') = v(q') \quad \text{et} \quad v(p) = v(q) = w(p') = w(q')$$

$$\text{d'où} \quad pgcd_p = pgcd_q = pgcd_{q'} = pgcd_{p'}$$

On en déduit que $M^\star_{min}(p) = M^\star_{min}(q)$. La figure 5.4 page ci-contre décrit la situation.

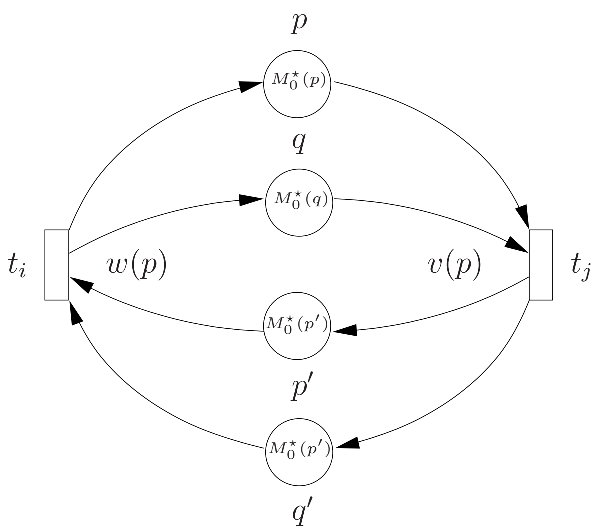


FIG. 5.4: Exemple de places parallèles $p = (t_i, t_j)$ et $q = (t_i, t_j)$.

Considérons alors le marquage initial du circuit $C = (t_i, q, t_j, p', t_i)$:

$$\begin{array}{rcl} M_0^\star(q) + M_0^\star(p') & = & M_0^\star(q) + M_{min}^\star(p) - M_0^\star(p) \\ & < & M_0^\star(q) + M_{min}^\star(p) - M_0^\star(q) \\ & < & M_{min}^\star(p) \\ & < & v(q) + v(p') - pgcd_q \end{array}$$

D'après la condition nécessaire et suffisante de vivacité pour les circuits à deux transitions, le circuit C n'est pas vivant, soit une contradiction. □

Selon les lemmes 1 page 35 et 9 page ci-contre, les places parallèles ayant le même marquage initial modélisent les mêmes contraintes de précédence entre les occurrences des transitions adjacentes. Nous pouvons alors restreindre notre analyse aux graphes d'événements généralisés génériques dans la suite de ce chapitre.

5.3.3 Une condition suffisante de vivacité

Le résultat suivant est une application de la condition suffisante du théorème 7 page 63 pour les graphes à capacité minimum :

Application 1. *Soient $G = (T, P)$ un graphe borné et $G_R^{min} = (T, P_R, M_0^\star)$ un graphe à capacité minimum (normalisé). Supposons que pour tout circuit C de G_R^{min}, l'inégalité*

$$\sum_{p \in C \cap P_R} v(p) > \sum_{p \in C \cap P_R} M_0^\star(p) > \sum_{p \in C \cap P_R} (v(p) - pgcd_p)$$

soit satisfaite, alors G_R^{min} est vivant.

Démonstration. Soit C un circuit de G_R^{min}. Nous devons alors considérer deux cas :

- Supposons que le circuit C soit constitué de deux transitions notées t_i et t_j et qu'il existe une seule place p entre ces deux transitions dans G. Nous en déduisons d'après les règles de construction de G_R^{min} et du théorème 8 page 64 que C est vivant.
- Sinon, conformément aux règles de construction de G_R^{min}, il existe un circuit C' dans G_R^{min} tel que pour toute place $p = (t_i, t_j)$ de C correspond une place retour $p' = (t_j, t_i)$ dans C'. Le nombre de jetons $N_{C \cup C'}$ dans les places de $C \cup C'$ vérifie :

$$\begin{aligned} N_{C \cup C'} &= \sum_{p \in C \cap P_R} M^\star_{min}(p) \\ &= \sum_{p \in C \cap P_R} (w(p) + v(p) - pgcd_p) \\ &= \sum_{p \in C \cap P_R} M^\star_0(p) + \sum_{p \in C' \cap P_R} M^\star_0(p) \end{aligned}$$

Le graphe G_R^{min} étant normalisé, de par le théorème 7 page 63, C est vivant si

$$\sum_{p \in C \cap P_R} M^\star_0(p) > \sum_{p \in C \cap P_R} (v(p) - pgcd_p)$$

De même, C' est vivant si

$$\sum_{p \in C' \cap P_R} M^\star_0(p) > \sum_{p \in C' \cap P_R} (v(p) - pgcd_p)$$

Comme

$$\sum_{p \in C' \cap P_R} (v(p) - pgcd_p) = \sum_{p \in C \cap P_R} (w(p) - pgcd_p)$$

$$\text{et} \sum_{p \in C' \cap P_R} M^\star_0(p) = \sum_{p \in C \cap P_R} (w(p) + v(p) - pgcd_p) - \sum_{p \in C \cap P_R} M^\star_0(p)$$

la condition devient alors :

$$\sum_{p \in C \cap P_R} v(p) > \sum_{p \in C \cap P_R} M^\star_0(p)$$

□

Les marquages des graphes à capacité minimum pour lesquels nous allons pouvoir certifier la vivacité par la condition suffisante doivent également vérifier la contrainte décrite précédemment. La somme des marquages d'un circuit doit être en quelque sorte ni trop importante ni trop faible afin que l'on puisse certifier que le marquage de son circuit retour soit également vivant.

5.3.4 Restriction du problème

Nous proposons à présent une limitation de l'ensemble des marquages possibles afin de simplifier la recherche d'un marquage vivant. Pour toute place $p \in P_R$, on considère que le marquage initial $M_0^\star(p)$ est à valeur dans $\{v(p), v(p) - pgcd_p\}$. Nous montrons qu'il existe un algorithme polynomial permettant de résoudre le problème de marquage. Cet algorithme est une preuve constructive du théorème suivant :

Théorème 14. *Soient $G = (T, P)$ un graphe d'événements généralisé borné et $G_R = (T, P_R)$ son graphe à capacité associé. Un graphe à capacité minimum vivant $G_R^{min} = (T, P_R, M_0^\star)$ tel que pour toute place $p \in P_R$, $M_0^\star(p) \in \{v(p), v(p) - pgcd_p\}$ peut être construit en temps polynomial.*

Les sections suivantes sont consacrées au développement de cet algorithme.

5.4 Vers un algorithme polynomial pour la construction d'un marquage vivant

Nous supposons dans cette section que le graphe $G = (T, P)$ est borné et fortement connexe. Le cas général sera traité à la fin de ce chapitre.

Nous présentons rigoureusement chaque étape nécessaire à notre algorithme. Dans un premier temps, le graphe $G_R = (T, P_R)$ est transformé en un graphe orienté associé $\mathcal{G}$ afin de simplifier la présentation de l'algorithme. La condition suffisante de vivacité de G_R^{min} est ensuite transcrite en une propriété de la valuation des arcs du graphe associé $\mathcal{G}$. La terminaison de l'algorithme découle alors de cette dernière propriété. Nous détaillons enfin les phases d'initialisation et d'itération de notre algorithme.

5.4.1 Définition du graphe associé

Nous supposons ici que $G = (T, P)$ est un graphe borné fortement connexe. Nous rappelons que d'après le corollaire 2 page 78, G est normalisable. Le graphe $G_R = (T, P_R)$ désigne le graphe à capacité associé à G.

Afin de simplifier les preuves, nous introduisons le graphe orienté associé $\mathcal{G} = (T, U)$ défini par :

1. les sommets de $\mathcal{G}$ sont les transitions de G ;
2. chaque place $p = (t_i, t_j)$ de P_R est associée à un arc u_p de t_i vers t_j. En outre, pour distinguer les places appartenant à P_R^1 de celles de P_R^2, nous représenterons sur les figures suivantes des arcs à traits continus (*resp.* en pointillés) pour décrire les arcs de U associés aux places de P_R^1 (*resp.* P_R^2).

D'après nos hypothèses, le marquage initial de chaque place $p \in P_R$ vérifie $M_0^\star(p) \in \{v(p), v(p) - pgcd_p\}$. On peut donc associer à tout graphe à capacité minimum $G_R^{min} = (T, P_R, M_0^\star)$ une valuation l des arcs de $\mathcal{G}$ définie de la manière suivante :

$$\begin{cases} l(u_p) = 0 & \text{si} \quad M_0^\star(p) = v(p) - pgcd_p \\ l(u_p) = 1 & \text{si} \quad M_0^\star(p) = v(p) \end{cases}$$

Inversement, à toute valuation du graphe orienté $\mathcal{G}$ correspond un marquage du graphe à capacité minimum $G_R^{min} = (T, P_R, M_0^\star)$.

Propriété 4. *Si $G_R^{min} = (T, P_R, M_0^\star)$ est un graphe à capacité minimum, alors pour tout couple de places $(p_1, p_2) \in P_R^1 \times P_R^2$ associé à une place p dans G :*

$$l(u_{p_1}) + l(u_{p_2}) = 1$$

Démonstration. Par construction de G_R^{min}, nous avons :

$$M_0^\star(p_1) + M_0^\star(p_2) = M_{min}^\star(p) = w(p) + v(p) - pgcd_p$$

Sachant que $M_0^\star(p_1) \in \{v(p), v(p) - pgcd_p\}$ et que $M_0^\star(p_2) \in \{w(p), w(p) - pgcd_p\}$, nous obtenons la propriété. □

Le lemme suivant établit un pont entre la condition suffisante de vivacité évoquée dans l'application 1 page 81 et la valuation associée à un marquage initial de G_R^{min}.

Lemme 10. *Soit l une valuation de $\mathcal{G}$ qui satisfait à la propriété 4. Si chaque circuit $\mathcal{C}$ de $\mathcal{G}$ possède au moins un arc x et un arc y tels que $l(x) = 0$ et $l(y) = 1$, alors le graphe G_R^{min} à capacité minimum associé à cette valuation est vivant.*

Démonstration. Considérons un circuit $\mathcal{C}$ de $\mathcal{G}$ ainsi que son circuit C correspondant dans le graphe G_R^{min}. Si la valuation l vérifie la condition du lemme, alors nous avons

$$|\mathcal{C}| > \sum_{p \in C \cap P_R} l(u_p) > 0$$

où $|\mathcal{C}|$ désigne le nombre d'arcs du circuit $\mathcal{C}$. Nous obtenons alors :

$$\sum_{p \in C \cap P_R} v(p) > \sum_{p \in C \cap P_R} M_0^\star(p) > \sum_{p \in C \cap P_R} (v(p) - pgcd_p)$$

D'après l'application 1, nous concluons que $G_R^{min} = (T, P_R, M_0^\star)$ est vivant. □

On remarque alors que ce problème semble étroitement lié au problème NP-Complet Feedback Arc Set [Kar72]. En effet, le problème Feedback Arc Set consiste à déterminer sur un graphe orienté $G = (V, U)$ un ensemble d'arcs U' de taille minimum tel que le sous graphe $G' = (V, U - U')$ soit acyclique. En revanche, dans notre problème l'instance est un graphe symétrique et il convient de sélectionner un ensemble d'arcs (*via* la valuation des arcs) qui vérifie uniquement la propriété structurelle du problème Feedback Arc Set.

5.4.2 Terminaison de l'algorithme

Soient U_0 l'ensemble des arcs de valuation nulle de $\mathcal{G}$ et $\mathcal{G}_0 = (T, U_0)$ le graphe partiel induit par ces arcs.

Théorème 15. *Si $\mathcal{G}_0$ est un graphe acyclique alors G_R^{min} est vivant.*

Démonstration. Supposons que G_R^{min} ne soit pas vivant. La valuation l vérifiant la propriété 4, d'après le lemme 10, il existe un circuit $\mathcal{C}$ de $\mathcal{G}$ tel que $l(x) = 0$ pour tout arc x de $\mathcal{C}$. Le circuit $\mathcal{C}$ appartient alors à $\mathcal{G}_0$ qui n'est donc pas acyclique. □

5.4.3 Phase d'initialisation de la valuation de $\mathcal{G}_0$

Considérons une arborescence couvrante notée $\mathcal{T}$ de G. L'arborescence $\mathcal{T}$ est bien définie car G est fortement connexe. On construit alors une valuation initiale de $\mathcal{G}$ comme suit :

- Si u_p est un arc à trait continu, alors $l(u_p) = 1$ si et seulement si la place correspondante $p \in P$ dans G appartient à $\mathcal{T}$.
- Si u_p est un arc à trait discontinu, alors $l(u_p) = 1$ si et seulement si l'arc retour $u_{p'}$ a une valuation $l(u_{p'}) = 0$.

Propriété 5. *Le graphe $\mathcal{G}_0$ ne contient pas de circuit à traits discontinus.*

Démonstration. Supposons que $\mathcal{G}_0$ contienne un circuit à traits discontinus $\mathcal{C}$. Selon les règles de construction des valuations, la propriété 4 s'applique. Il s'ensuit que le circuit retour $\mathcal{C}'$ de $\mathcal{C}$ est composé exclusivement d'arcs à traits continus et par conséquent $\mathcal{C}'$ est inclut dans $\mathcal{T}$, soit une contradiction. □

Conséquence 2. *Initialement, tous les circuits de $\mathcal{G}_0$ sont soit des circuits à traits continus soit des circuits à traits continus et discontinus.*

5.4.4 Elimination des circuits de $\mathcal{G}_0$

L'idée sous-jacente de notre algorithme est de construire une valuation de $\mathcal{G}$ telle que $\mathcal{G}_0$ soit dépourvu de circuit. D'après le théorème 15, le graphe à capacité minimum G_R^{min} correspondant à cette valuation est alors vivant.
A cette étape de l'algorithme, on peut imaginer qu'il subsiste des circuits dans le graphe $\mathcal{G}_0$. Nous devons alors proposer une méthode pour éliminer les circuits de $\mathcal{G}_0$.

Théorème 16. *Si $\mathcal{G}_0$ est dépourvu de circuit à traits discontinus et contient cependant un circuit $\mathcal{C}$, alors il existe au moins un arc à trait continu $\mathcal{C}$ pour lequel il est possible de modifier la valuation et de l'établir à 1 sans toutefois créer un circuit à traits discontinus dans $\mathcal{G}_0$.*

Démonstration. Cette preuve est basée sur un raisonnement par contradiction. Supposons que $\mathcal{G}_0$ soit dépourvu de circuit à traits discontinus. Soit $\mathcal{C}$ un circuit de $\mathcal{G}_0$ tel qu'il n'existe aucun arc à trait continu de $\mathcal{C}$ pouvant être valué par 1 sans créer de circuit à traits discontinus en ajoutant au graphe $\mathcal{G}_0$ l'arc retour à traits discontinus. On remarque que le retrait d'un arc à trait continu de $\mathcal{G}_0$ ne peut pas engendrer de circuit à traits continus dans $\mathcal{G}_0$. Dans ce cas, il s'ensuit qu'il existait déjà un circuit à traits discontinus dans $\mathcal{G}_0$ (*voir* Figure 5.5), soit une contradiction. □

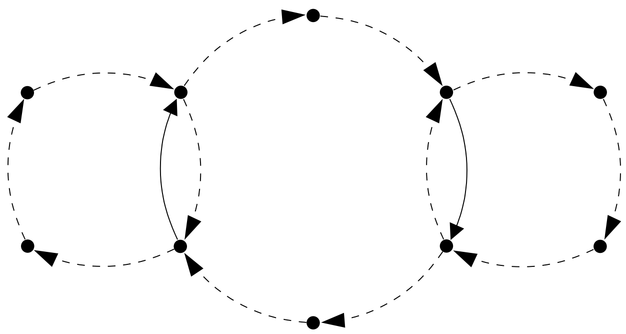


Fig. 5.5: Un circuit $\mathcal{C}$ est représenté au centre de la figure. Si aucune des valuations des arcs à traits continus de $\mathcal{C}$ ne peut être modifiée sans engendrer un nouveau circuit à traits discontinus, alors il existait au préalable un tel circuit dans $\mathcal{G}_0$(*cf.* le circuit pourtour de la figure).

Suivant le théorème 16, l'algorithme va donc progressivement supprimer des arcs de $\mathcal{G}_0$. De plus, nous en déduisons que le nombre d'arcs à traits continus du graphe $\mathcal{G}_0$ décroît strictement à chacune de ces étapes. L'algorithme s'arrête lorsque le graphe $\mathcal{G}_0$ est acyclique.

5.5 Algorithmes

Nous présentons dans un premier temps un algorithme pour les graphes d'événements généralisés fortement connexes et nous caractérisons sa complexité algorithmique. Nous illustrons ensuite cet algorithme sur un exemple. Enfin, nous proposons une généralisation de cet algorithme pour les graphes bornés.

5.5.1 Algorithme pour le cas fortement connexe

Nous remarquons d'abord que les étapes de l'algorithme font appel uniquement à la structure du graphe et non aux paramètres numériques de l'instance (*i.e.* les valeurs des fonctions de marquages). L'algorithme calcule pour toute place p un marquage initial $M_0^\star(p)$ à valeur dans $\{v(p) - pgcd_p, v(p)\}$. La normalisation étant une fonction linéaire croissante des attributs numériques d'une place (*i.e.* $w(p)$,$v(p)$, $M_0^\star(p)$ et

$pgcd(k \cdot w(p), k \cdot v(p)) = k \cdot pgcd_p$), il s'ensuit que l'étape de normalisation n'est pas nécessaire.

Nous notons alors U_0^{Δ} le sous-ensemble d'arcs à traits discontinus de U_0 et $\mathcal{G}_0^{\Delta} = (T, U_0^{\Delta})$.

1. Construire une arborescence couvrante $\mathcal{T}$ du graphe initial G (en utilisant par exemple l'algorithme de DIJKSTRA). Pour tout arc de $\mathcal{T}$, valuer l'arc à trait continu de $\mathcal{G}$ par 1. Pour tout arc de $G - \mathcal{T}$, valuer l'arc à trait continu de $\mathcal{G}$ par 0. Les arcs à trait discontinu de $\mathcal{G}$ sont valués suivant la propriété 4.
2. Extraire de $\mathcal{G}$ le sous-graphe $\mathcal{G}_0$.
3. Vérifier avec l'algorithme de RECHERCHE EN PROFONDEUR que le graphe $\mathcal{G}_0$ est acyclique :
   (a) Si $\mathcal{G}_0$ est acyclique alors le graphe à capacité minimum associé G_R^{min} est vivant (*voir* théorème 15). Aller à l'ÉTAPE 4.
   (b) Si $\mathcal{G}_0$ contient un circuit $\mathcal{C}$ alors soit S l'ensemble des arcs à traits continus de $\mathcal{C}$.
      i. Sélectionner un arc $u_s \in S$. Soit u_d son arc retour à traits discontinus.
      ii. Si le sous-graphe à traits discontinus $\mathcal{G}_0^{\Delta'} = (T, U_0^{\Delta} \cup \{u_d\})$ est acyclique, alors u_s peut être valué par 1 : poser $U_0 = U_0 \cup \{u_d\} - \{u_s\}$ et retourner à l'ÉTAPE 3.
      iii. Sinon, poser $S = S - \{u_s\}$ et aller à l'ÉTAPE 3(b)i.
4. Le marquage initial $M_0^{\star}$ de G_R^{min} est construit comme suit :
   Pour tout arc u_p de $\mathcal{G}$, poser $M_0^{\star}(p) = v(p) - pgcd_p$ si $l(u_p) = 0$ sinon $M_0^{\star}(p) = v(p)$.

Lemme 11. *Le temps de calcul de l'algorithme précédent est proportionnel à $\mathcal{O}(m^2)$.*

Démonstration. Les temps de calcul des algorithmes de DIJKSTRA et de RECHERCHE EN PROFONDEUR sont en $\mathcal{O}(m)$ [CLR90]. A chaque étape de l'algorithme, un arc à trait continu est sélectionné et nous devons vérifier que cet arc peut être remplacé par son arc retour à trait discontinu sans générer de circuit. Cette vérification est faite en utilisant l'algorithme de RECHERCHE EN PROFONDEUR. Le nombre d'itérations de l'ÉTAPE 3 étant en $\mathcal{O}(m)$, nous en déduisons que le nombre maximum d'opérations élémentaires de cet algorithme est en $\mathcal{O}(m^2)$. □

5.5.2 Exemple

Nous présentons le déroulement de l'algorithme précédent sur un exemple. On considère pour cela le graphe d'événements généralisé $G = (T, P)$ de la figure 5.6 a). La figure 5.6 b) représente le graphe à capacité associé noté $G_R = (T, P_R)$.

Lors de l'ÉTAPE 1, l'algorithme construit une arborescence couvrante $\mathcal{T}$ dont la racine est la transition t_5. Les arcs de $\mathcal{T}$ sont alors valués par 1. Le graphe résultant est alors décrit à la figure 5.7 c). La figure 5.7 d) montre le graphe valué associé $\mathcal{G}$.

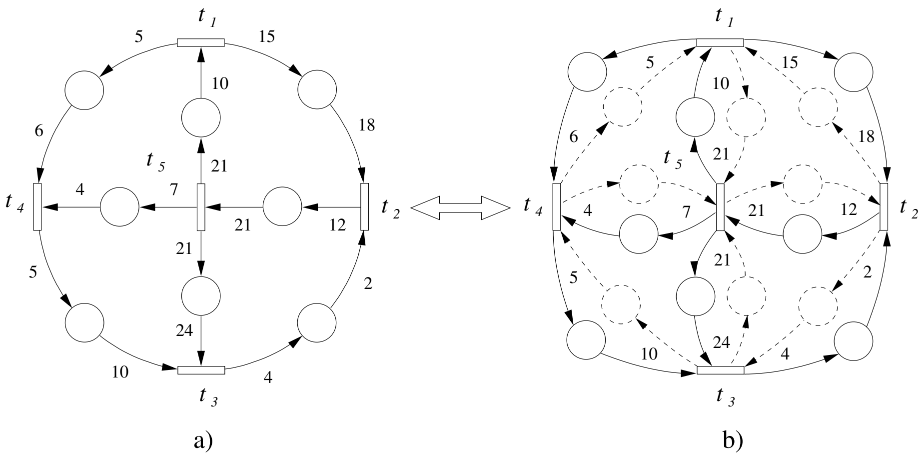


FIG. 5.6: Un graphe d'événements généralisé $G = (T, P)$ est décrit sur la gauche de la figure. Le graphe à capacité $G_R = (T, P_R)$ correspondant est présenté sur la droite.

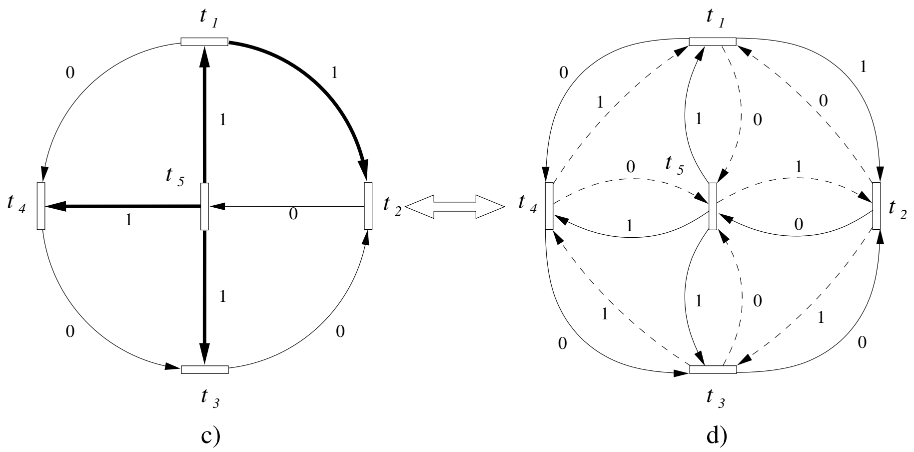


FIG. 5.7: L'arborescence couvrante $\mathcal{T}$ de G est présentée sur la figure c) par des traits en gras. La figure d) dépeint le graphe valué associé $\mathcal{G}$ correspondant.

Le sous-graphe $\mathcal{G}_0$ de $\mathcal{G}$ ainsi qu'un circuit $\mathcal{C}$ de $\mathcal{G}_0$ (représenté en gras) sont représentés sur la figure 5.8 e). Le circuit $\mathcal{C}$ est brisé en renversant l'arc (t_1, t_4). La figure 5.8 f) montre le graphe $\mathcal{G}_0$ résultant de cette opération. Ce graphe est acyclique.

La figure 5.9 g) décrit la valuation finale du graphe $\mathcal{G}$. Le marquage minimum vivant G_R^{min} correspondant à cette valuation est représenté sur la figure 5.9 h).

5.5.3 Généralisation

Nous généralisons dans cette sous-section l'algorithme au cas des graphes bornés.

Soit $G = (T, P)$ un graphe borné composé de k composantes fortement connexes

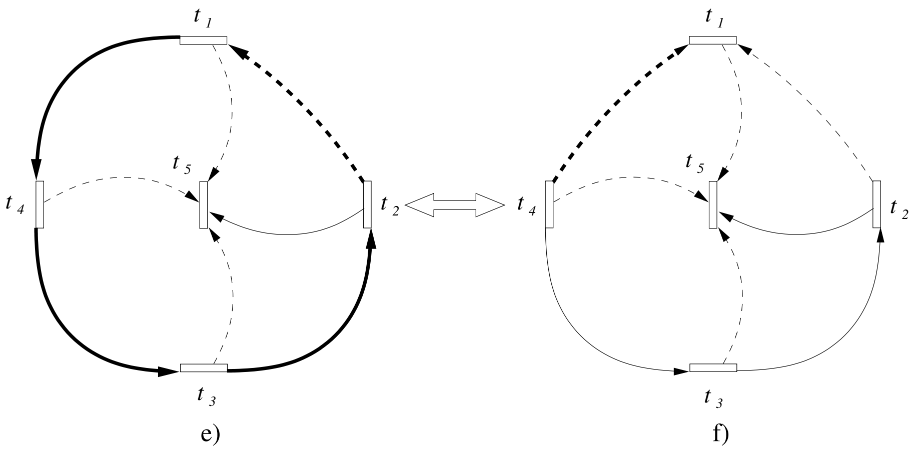


FIG. 5.8: Sur la gauche, le graphe $\mathcal{G}_0$ et un circuit $\mathcal{C}$ avec des arcs en gras sont présentés sur la gauche. Sur le côté droit de la figure, nous présentons le graphe $\mathcal{G}_0$ obtenu par le renversement de l'arc (t_1, t_4).

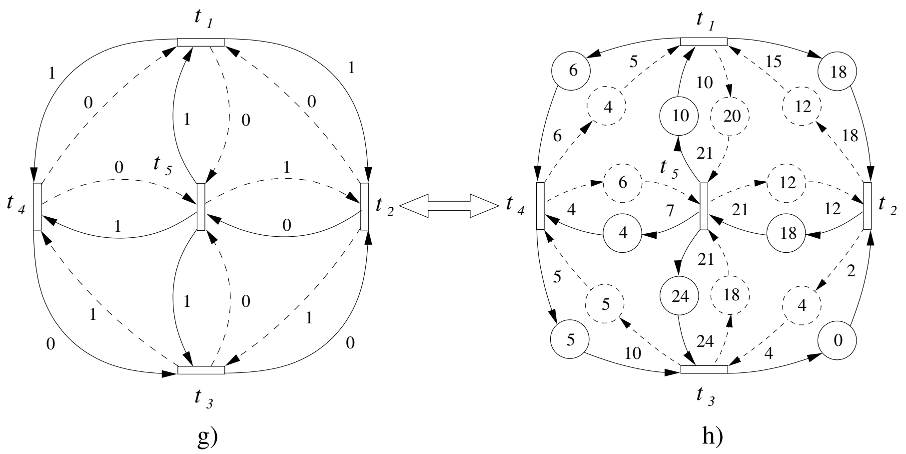


FIG. 5.9: La figure g) montre le graphe associé $\mathcal{G}$. Le graphe à capacité minimum G_R^{min} associé à cette valuation est présenté sur la figure h).

notées $\mathcal{FC}_1, \ldots, \mathcal{FC}_k$. On remarque alors que chaque composante $\mathcal{FC}_j$, $j \in \{1, \ldots, k\}$ est un graphe d'événements généralisé borné fortement connexe de sorte que l'algorithme précédent peut s'appliquer sur chacune d'elles. Le problème consiste alors à déterminer un marquage des places situées à cheval entre deux composantes fortement connexes distinctes. Ces places sont appelées places indépendantes et sont définies comme suit :

Définition 15. *Une place $p = (t_x, t_y) \in P$ est dite indépendante si*

$$(t_x, t_y) \in \mathcal{FC}_i \times \mathcal{FC}_j \ , (i, j) \in \{1, \ldots, k\}^2, i \neq j$$

L'ensemble des places indépendantes de G est noté par $P_I \subseteq P$. De même, $P_I^a \subseteq P_R$ représente l'ensemble des places de P_R associées à une place indépendante.

Nous généralisons alors l'algorithme précédent pour les graphes bornés :

1. Déterminer les composantes fortement connexes $\mathcal{FC}_1, \ldots, \mathcal{FC}_k$ du graphe G avec l'algorithme de RECHERCHE EN PROFONDEUR. Déterminer l'ensemble P_I des places indépendantes du graphe G.
2. A l'aide de l'algorithme précédent, calculer un marquage initial $M_0^\star$ pour les graphes à capacité associés aux composantes fortement connexes de G.
3. Pour chaque place $p \in P_I^a$, si $p \in P_R^1$ faire $M_0^\star(p) = v(p)$ sinon $M_0^\star(p) = v(p) - pgcd_p$.

Théorème 17. *L'algorithme généralisé construit un marquage vivant pour G_R^{min}.*

Démonstration. Nous procédons par un raisonnement par contradiction. Supposons que $k > 1$ et que G_R^{min} ne soit pas vivant. De par l'application 1, il existe un circuit C dans G_R^{min} dont chaque place p est marquée par $M_0^\star(p) = v(p)$ jetons. Par ailleurs, pour chaque composante fortement connexe de G, l'algorithme calcule un marquage initial vivant. Nous en déduisons que le circuit C contient des places appartenant à au moins deux composantes distinctes $\mathcal{FC}_i$ et $\mathcal{FC}_j$. Le marquage $M_0^\star(p)$ étant égal à $v(p)$ jetons pour toute place de C, nous en déduisons que $C \cap P_I^a \subset P_R^1$. Il existe alors un circuit dans G passant par les composantes fortement connexes $\mathcal{FC}_i$ et $\mathcal{FC}_j$, soit une contradiction. $\square$

On détermine alors la complexité de cet algorithme.

Théorème 18. *Le temps de calcul de cet algorithme est proportionnel à $\mathcal{O}(n \cdot m^2)$*

Démonstration. Le temps de calcul de cet algorithme dépend principalement du nombre d'appels à l'algorithme de RECHERCHE EN PROFONDEUR ainsi qu'à la complexité de l'algorithme du cas fortement connexe. La première et la troisième étape de cet algorithme sont toutes deux en $\mathcal{O}(m)$. Or $|T| = n$, le nombre de composantes fortement connexes du graphe G est en $\mathcal{O}(n)$. De plus, selon le lemme 11 page 87, la complexité de l'algorithme pour le cas fortement connexe est en $\mathcal{O}(m^2)$. Il s'ensuit que le nombre de calculs effectués lors de la deuxième étape de cet algorithme est en $\mathcal{O}(n \cdot m^2)$. $\square$

Conclusion

Dans ce chapitre, nous avons présenté et analysé le problème de marquage. Ce problème avait été abordé par Adé [Adé96] dans sa thèse. Sa méthode reposait sur l'analyse de motifs particuliers des graphes *synchronous dataflow*. Son heuristique s'appuie ensuite sur une décomposition du graphe de l'application en ces motifs particuliers. Nous avons proposé ici un algorithme polynomial pour résoudre le problème de marquage de façon exacte.

Pour traiter ce problème, nous avons procédé à trois simplifications importantes afin de contourner le problème de l'évaluation de la vivacité. Nous avons dans un premier temps montré que les contraintes de capacité associées aux places pouvaient être exprimées par le formalisme des réseaux de Petri en ayant recours à un couple de places dépourvues de contraintes de capacité. Cette modélisation de la capacité permet donc de simplifier le problème de marquage et de nous concentrer sur le problème de la vivacité. En outre, nous avons caractérisé la notion de graphe borné qui décrit les applications susceptibles de fonctionner avec des tailles de mémoires préalablement bornées par le concepteur.

La notion de jetons utiles présentée au chapitre précédent a permis de limiter l'ensemble des solutions de ce problème. Pour contourner le problème lié à l'évaluation de la vivacité d'un graphe d'événements généralisé, nous avons judicieusement limité les marquages initiaux que notre algorithme considère. En adaptant la condition suffisante de vivacité pour les graphes à capacité minimum, nous avons exprimé le problème initial comme un problème de valuation d'arcs sur un graphe orienté associé à l'instance initiale. Nous avons alors montré que l'on pouvait construire une valuation solution de ce nouveau problème et par conséquent un marquage initial solution du problème de marquage.

L'algorithme final permettant une résolution efficace du problème de marquage est fondé sur l'ensemble de ces concepts. Cependant, nous verrons au chapitre 6 qu'il est possible d'améliorer sensiblement la complexité de notre méthode.

Deuxième partie

Etude de problèmes d'optimisation bi-critère

Chapitre 6

Complexité de problèmes d'optimisation bi-critère

Sommaire

Ce chapitre est consacré à l'étude de la complexité de problèmes d'optimisation bi-critère pour les graphes d'événements généralisés. Après avoir rappelé la définition du débit des transitions d'un graphe d'événements généralisé, nous proposons une définition du débit global d'un tel système. Ce débit associé à un marquage initial souligne le lien entre les différents débits des transitions.

Nous évoquons ensuite un problème d'optimisation bi-critère pour lequel on dispose d'un marquage initial. Ce problème consiste à déterminer la taille minimum des mémoires à allouer à un système embarqué dont les conditions initiales des mémoires (*i.e.* présence de données initiales dans les mémoires) sont fixées et ce tout en garantissant l'absence de blocage. Nous rappelons le résultat obtenu par Murthy pour ce problème [Mur96].

Nous nous intéressons par la suite à la complexité théorique de problèmes d'optimisation bi-critère non-marqué (*i.e.* instances pour lesquelles, le concepteur peut lui même définir l'état initial des mémoires). Nous établissons un lien fort entre ces problèmes d'optimisation bi-critère et le problème de coloration des sommets d'un graphe. Pour cela, nous introduisons le problème d'orientation d'arêtes défini par Minty [Min62]. Cette relation nous permet également de concevoir un nouvel algorithme plus performant pour le problème de marquage évoqué au chapitre 5. Enfin, nous nous intéressons à deux problèmes d'optimisation bi-critère non-marqué faisant l'objet de fréquentes recherches dans la communauté réseau de Pétri. Ces deux problèmes, bien qu'étant largement étudiés depuis plus de 20 ans (*cf.* [HP89, Gau90, LPX92, DFMS97, GPS02, Sau03]), demeuraient de complexité théorique inconnue. Nous montrons qu'ils sont tous deux NP-complets.

6.1 Présentation des problèmes d'optimisation bi-critère

Nous présentons dans cette section les concepts de base. Nous rappelons ensuite un résultat obtenu par Murthy dans le cas marqué, puis nous évoquons le cas non-marqué.

6.1.1 Rappel des concepts de base

Dans cette section, nous traitons des graphes d'événements temporisés. On définit dans un premier temps la notion de marquage instantané $M(\tau, p)$ d'une place $p = (t_i, t_j)$ à la date τ. Pour toute valeur $\tau \in \mathbb{R}^{+\star}$, on note $E(\tau, t_i)$ le nombre de franchissements de t_i terminés à la date τ, soit :

$$E(\tau, t_i) = max\{q \in \mathbb{N},\ s(t_i, q) + l(t_i) \leqslant \tau\}$$

De même, $B(\tau, t_j)$ désigne le nombre de franchissements de la transition t_j entamés à la date τ, soit :

$$B(\tau, t_j) = max\{q \in \mathbb{N},\ s(t_j, q) \leqslant \tau\}$$

Nous pouvons à présent exprimer le marquage instantané de la place p à la date τ en utilisant ces deux dernières fonctions :

$$M(\tau, p) = M(0, p) + w(p) \cdot E(\tau, t_i) - v(p) \cdot B(\tau, t_j)$$

Un ordonnancement est valide si $M(\tau, p) \geqslant 0$ pour tout couple $(\tau, p) \in \mathbb{R}^{+\star} \times P$.

Considérons maintenant un ordonnancement valide $S = \{s(t_i, q),\ \forall (t_i, q) \in T \times \mathbb{N}^\star\}$ des transitions d'un graphe d'événements généralisé temporisé quelconque. Le débit de franchissement d'une transition $t_i \in T$ est défini par :

$$\lambda(t_i) = \lim_{q \to \infty} \frac{q}{s(t_i, q)}$$

Dans le cas non-généralisé, nous avions rappelé au chapitre 3 le résultat de Chrétienne [Chr83] caractérisant le débit des transitions dans l'ordonnancement au plus tôt :

Théorème 3. [Rappel] *Soit G un graphe d'événements temporisé fortement connexe ayant un marquage initial vivant. Il existe un instant $\tau \in \mathbb{R}^+$ à partir duquel les dates de début de l'ordonnancement au plus tôt forment des suites K-périodiques de période $\alpha(G) \cdot K$ où K est le produit des hauteurs des circuits critiques de G. De plus, le débit du système de marquage $M(G)$ noté $\lambda(M(G))$ vaut :*

$$\lambda(M(G)) = \min_{C \in C_G} \frac{H(C)}{L(C)} = \frac{1}{\alpha(G)}$$

On dira qu'un circuit C est critique si $\frac{H(C)}{L(C)} = \lambda$.

En se basant sur le lemme de Koenig [BBC92], Chrétienne montre également le théorème suivant :

Théorème 19 ([Chr83]). *Soient G un graphe d'événements temporisé et $M(G)$ un marquage initial vivant. Tout circuit critique de G peut être décomposé en circuits élémentaires critiques.*

Dans le cas non-généralisé, le calcul des fréquences de franchissement des transitions peut être réalisé efficacement par un algorithme de calcul de circuit de poids moyen minimum ([DIG99]).

Exemple

Nous illustrons par un exemple la notion de débit d'un graphe d'événements temporisé. On considère le graphe d'événements temporisé G de la figure 6.1 page suivante. Ce graphe

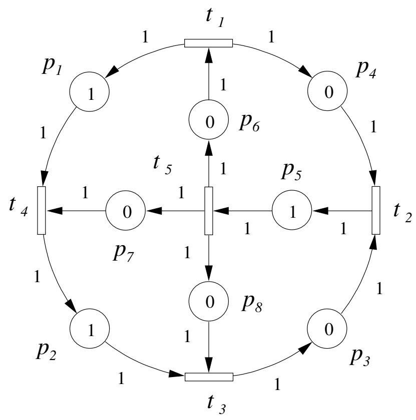


FIG. 6.1: Graphe d'événements temporisé de débit $\lambda(M(G)) = \frac{1}{7}$.

possède cinq transitions telles que $l(t_1) = l(t_2) = 1, l(t_3) = 4$ et $l(t_4) = l(t_5) = 2$ et son marquage initial vaut $M_0(p_1) = M_0(p_2) = M_0(p_5) = 1$ et $M_0(p_3) = M_0(p_4) = M_0(p_6) = M_0(p_7) = M_0(p_8) = 0$.

Nous calculons le rapport $\frac{H(C)}{L(C)}$ pour chaque circuit élémentaire $C \in C_G$ du graphe G.

- Pour le circuit $C_1 = (t_1, t_4, t_3, t_2, t_5, t_1)$, nous avons $\frac{H(C_1)}{L(C_1)} = \frac{3}{10}$.
- Pour le circuit $C_2 = (t_4, t_3, t_2, t_5, t_4)$, nous avons $\frac{H(C_2)}{L(C_2)} = \frac{2}{9}$.
- Pour le circuit $C_3 = (t_3, t_2, t_5, t_3)$, nous avons $\frac{H(C_3)}{L(C_3)} = \frac{1}{7}$.
- Pour le circuit $C_4 = (t_2, t_5, t_1, t_2)$, nous avons $\frac{H(C_3)}{L(C_3)} = \frac{1}{4}$.

Nous en déduisons que le débit des transitions associé à ce marquage vaut $\frac{1}{7}$ et que C_3 est le circuit critique.

Contrairement au cas non-généralisé, le calcul des débits des transitions avec une politique d'ordonnancement au plus tôt reste un problème de complexité inconnue pour le cas généralisé. De même que pour la vivacité, les méthodes exactes pour le calcul de ces débits reposent sur l'analyse des circuits du graphe expansé (*cf.* chapitre 3). En conséquence, la complexité du calcul des débits des transitions par le biais de l'expansion est pseudo-polynomiale.

Dans le cas généralisé, Munier [Mun93] montre que les débits des transitions $t_i \in T$ de l'ordonnancement au plus tôt sont tous liés. En effet, si on considère ${}^{\intercal}N = (N_1, \ldots, N_n)$ un T-semiflot d'un graphe unitaire fortement connexe, alors pour tout couple de transitions (t_i, t_j), nous avons :

$$\frac{\lambda(t_i)}{N_i} = \frac{\lambda(t_j)}{N_j}$$

Cependant, la normalisation présentée au chapitre 4 permet d'associer à un marquage initial d'un graphe d'événements généralisé temporisé une valeur unique pour décrire son débit. Le résultat suivant définit le débit global d'un graphe associé à un marquage initial :

Théorème 20. *Soient G un graphe d'événements généralisé temporisé et $M(G)$ un marquage initial vivant. Alors, l'ordonnancement au plus tôt est K-périodique et pour tout couple de transitions $(t_i, t_j) \in T^2$, nous avons :*

$$\lambda(t_i) \cdot Z_i = \lambda(t_j) \cdot Z_j \stackrel{def}{=} \lambda(M(G))$$

La valeur $\lambda(M(G))$ est appelé le débit de G associé au marquage initial $M(G)$.

Démonstration. La validité du théorème repose sur les résultats établis par Munier [Mun93]. En effet, pour tout couple de transitions $(t_i, t_j) \in T^2$, nous avons :

$$\frac{\lambda(t_i)}{N_i} = \frac{\lambda(t_j)}{N_j}$$

Le corollaire 1 page 61 sur la normalisation montre qu'il existe une constante K telle que :

$$N_i \cdot Z_i = K \quad \forall t_i \in T$$

En multipliant par K toutes les égalités sur les débits, nous obtenons le théorème. □

Supposons à présent que le marquage initial $M(G)$ d'un graphe d'événements généralisé temporisé G soit tel que son ordonnancement au plus tôt comporte au moins une transition qui est franchie sans interruption. On note $t_{i^\star}$ une telle transition. Le débit de cette transition vaut alors $\lambda(t_{i^\star}) = \frac{1}{l(t_{i^\star})}$. Alors, d'après la définition de $\lambda(M(G))$ et comme pour toute transition $t_i \in T$ nous avons $\lambda(t_i) \leqslant \frac{1}{l(t_i)}$, nous obtenons :

$$\lambda(M(G)) = \frac{Z_{i^\star}}{l(t_{i^\star})} \leqslant \frac{Z_i}{l(t_i)}, \forall t_i \in T$$

On remarque qu'il est toujours possible d'atteindre une telle configuration du débit (*i.e.* une transition qui est effectuée sans interruption) en disposant d'un marquage initial conséquent. Au delà d'un certain seuil, il est inutile d'ajouter des jetons pour espérer augmenter le débit du système. Le système a alors atteint sa capacité de fonctionnement maximale. Nous en déduisons une définition du débit maximum intrinsèque d'un graphe d'événements généralisé temporisé.

Définition 16. *Le débit maximum intrinsèque d'un graphe d'événements généralisé temporisé unitaire G vaut $\min_{t_i \in T} \left(\frac{Z_i}{l(t_i)}\right)$. Cette valeur de débit correspond au débit maximum atteignable par un marquage initial $M(G)$.*

La définition précédente implique que dans le cas non-généralisé, les Z_i étant tous égaux à 1, le débit maximum intrinsèque d'un graphe d'événements temporisé vaut :

$$\min_{t_i \in T}\left(\frac{Z_i}{l(t_i)}\right) = \frac{1}{\max\limits_{t_i \in T}(l(t_i))}$$

Dans ce cas, c'est la transition de durée maximale qui limite la vitesse de fonctionnement du système. En reprenant l'exemple de la figure 6.1 page 98, nous déduisons que le débit maximum intrinsèque du graphe G vaut $\frac{1}{4}$. La transition t_3 et sa contrainte de non-réentrance sont responsables de cette limite du débit des transitions.

6.1.2 Le cas marqué

Nous présentons dans cette section un résultat de complexité obtenu par Murthy [Mur96]. Les travaux de Murthy ont été effectués dans le cadre des graphes Synchronous Dataflow (*voir* chapitre 2) au sein du *Ptolemy Project* à l'université de Berkeley. Au cours du chapitre 3, nous avons vu que ce modèle était équivalent au modèle des graphes d'événements généralisés à quelques détails près. Pour obtenir ce résultat de complexité, Murthy se restreint aux graphes d'événements temporisés (*i.e.* Homogenous Dataflow). Murthy considère le problème qui consiste à déterminer, pour un graphe d'événements temporisé ayant un marquage initial, la taille minimale des capacités à allouer aux places de sorte qu'il existe un ordonnancement valide qui minimise la somme des capacités. Murthy définit dans un premier temps pour chaque mémoire la quantité maximum de données qu'elle stocke au cours de l'ordonnancement. En reformulant le problème dans le formalisme des graphes d'événements temporisé, cette quantité pour chaque place p_i vaut alors $\max\limits_{\tau \in R^{+\star}}(M(\tau, p_i))$.

Nous rappelons dans un premier temps la définition du problème de décision associé :

GET MARQUÉ - SURFACE MIN :

Instance : Soient $G = (T, P, l)$ un graphe d'événements temporisé de marquage initial $M(G)$ et un entier $K > 0$.

Question : Existe-t-il un ordonnancement de G tel que $\sum\limits_{p_i \in P} \max\limits_{\tau \in R^{+\star}}(M(\tau, p_i)) \leqslant |P| + K$?

Murthy montre que ce problème est NP-complet. La démonstration propose une réduction polynomiale avec le problème FEEDBACK ARC SET [Kar72]. Le problème sans critère de débit étant NP-complet, nous en déduisons que le problème de décision pour lequel on adjoint une borne de débit à la question du problème de décision est également NP-complet.

Nous notons également que dans le cas non-généralisé, un certificat du problème de décision est un ordonnancement valide décrit par une séquence de franchissements de longueur $|T|$. Dans le cas généralisé, un tel ordonnancement est de longueur $\sum\limits_{t_i \in T} N_i$ où les

N_i sont les composantes d'un T-semiflot du graphe. La longueur de cet ordonnancement n'étant pas polynomiale en la taille de l'instance, il ne constitue pas un bon certificat au regard de la théorie de la complexité. De ce fait, nous ignorons si ce problème appartient à la classe NP. Par conséquent, le problème général (*i.e.* avec instances du cas généralisé) est NP-difficile. A la difficulté théorique du problème s'ajoute alors la difficulté de mettre au point des heuristiques efficaces pour le cas généralisé.

6.1.3 Le cas non-marqué

Dans ce chapitre, nous analysons la complexité de problèmes d'optimisation bi-critère dans le cas non marqué. Au chapitre 5, nous avons étudié et résolu de façon satisfaisante le problème de marquage. Nous avons montré que ce problème pouvait être traité par un algorithme polynomial. Nous nous sommes naturellement posés la question de savoir s'il était également possible de déterminer efficacement un marquage minimum vivant vérifiant un certain débit. Le résultat principal de la section suivante s'avère répondre négativement à la question.

Les instances des problèmes d'optimisation bi-critère non-marqués sont des graphes d'événements temporisés dépourvus de marquages initiaux. Le concepteur peut alors placer librement les jetons dans les places. Il définit ainsi les conditions initiales de ce système. L'objectif de ces problèmes d'optimisation bi-critère est de définir un marquage initial qui confère au graphe un certain débit tout en utilisant le moins possible de jetons.

A l'instar de la vivacité, les méthodes dont nous disposons pour l'évaluation du débit d'un graphe d'événements généralisé temporisé sont toutes pseudo-polynomiales. Afin de montrer la difficulté de ces problèmes d'optimisation bi-critère non-marqués, nous allons nous restreindre au cas non-généralisé.

On rappelle d'abord qu'un graphe d'événements temporisé est symétrique si pour toute place $p = (t_i, t_j) \in P$ il existe une place $p' = (t_j, t_i) \in P$. La place p' (*resp.* la place p) est appelée place retour de la place p (*resp.* p').

Définition 17. *Un graphe d'événements temporisé symétrique $G = (T, P, l)$ est minimalement borné si pour tout couple de places retours (p, p') on a :*

$$M(0,p) + M(0,p') = M_{min}(p) = 1$$

Nous pouvons ainsi définir le problème de décision Débit Max - Surface Min :

Débit Max - Surface Min :

Instance : Soient $G = (T, P, l)$ un graphe d'événements temporisé symétrique tel que $l(t) = 1, \forall t \in T$ et un entier $K > 1$.

Question : Existe-t-il un marquage initial vivant $M(G)$ minimalement borné tel que $\lambda(M(G)) \geqslant \frac{1}{K}$?

Ce problème bi-critère non-marqué joue un rôle central pour établir la complexité d'autres problèmes d'optimisation bi-critère non-marqués.

6.2 Complexité théorique du problème Débit Max - Surface Min

Nous étudions dans cette section la complexité théorique du problème Débit Max - Surface Min. Les sous-sections suivantes sont consacrées à la présentation du problème de Flot ratio ainsi qu'à la réduction polynomiale avec le problème Débit Max - Surface Min. Nous verrons enfin que la réduction que nous allons mettre à jour permet également d'élaborer un algorithme pour le problème de marquage (*voir* chapitre 5) d'une complexité pire-cas moindre.

6.2.1 Présentation et définition du problème Flot ratio

L'objectif de cette sous-section est de proposer une réduction polynomiale entre le problème Débit Max - Surface Min et le problème d'orientation des arêtes Flot ratio.

Nous présentons dans un premier temps le problème Flot ratio définit par Minty dans [Min62].

Soit $\mathcal{G} = (V, E)$ un graphe simple. Une orientation des arêtes est une fonction $o : E \to V \times V$ telle que, pour toute arête $e = \{x, y\} \in E$, $o(e) \in \{(x, y), (y, x)\}$. Le flot ratio $\rho(o(\mathcal{G}))$ associé à une orientation o est alors défini ainsi : pour tout cycle c de $\mathcal{G}$, nous associons deux entiers $n_c(o)$ et $m_c(o)$ correspondant respectivement aux nombres d'arêtes de c orientées dans un sens donné et dans le sens opposé. Alors, le flot ratio d'une orientation des arêtes de $\mathcal{G}$ est défini par :

$$\rho(o(\mathcal{G})) = \max_{c \in \mathcal{C}_\mathcal{G}} \left(\frac{n_c(o)}{m_c(o)}, \frac{m_c(o)}{n_c(o)} \right)$$

où $\mathcal{C}_\mathcal{G}$ désigne l'ensemble des circuits de $\mathcal{G}$.

La figure 6.2 page ci-contre décrit un exemple de graphe (à gauche) admettant une orientation (représentée sur le graphe de droite) dont le flot ratio résultant est 3 (*i.e.* $\max\left(\frac{1}{3}, \frac{3}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{2}, \frac{1}{2}\right)$).

Dans [Min62], Minty exhibe la relation forte existante entre le problème Flot ratio et le problème de coloration des sommets d'un graphe. Le lemme suivant expose cette relation :

Lemme 12 ([Min62]). *Soit $\mathcal{G}$ un graphe simple. $\mathcal{G}$ est K-colorable si et seulement s'il existe une orientation des arêtes du graphe dont le flot ratio est inférieur à $K - 1$.*

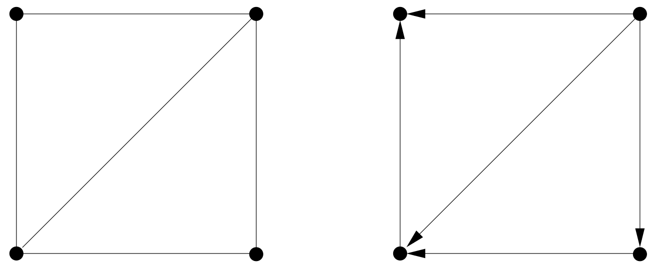


FIG. 6.2: Exemple de graphe admettant une orientation de flot ratio de 3.

Afin de caractériser la complexité de ce problème, nous définissons le problème de décision FLOT RATIO comme suit :

FLOT RATIO :

Instance : Soient $\mathcal{G} = (V, E)$ un graphe simple et $L > 0$ un entier positif.

Question : Existe-t-il une orientation o des arêtes de $\mathcal{G}$ telle que $\rho(o(\mathcal{G})) \leq L$?

Nous proposons de définir une application f transformant toute instance du problème FLOT RATIO en une instance de DÉBIT MAX - SURFACE MIN. Soit I une instance du problème FLOT RATIO composée d'un graphe $\mathcal{G} = (V, E)$ et d'un entier L. L'instance correspondante $f(I)$ de DÉBIT MAX - SURFACE MIN est définie par :

1. A chaque sommet $x \in V$ du graphe initial est associé une transition $t_x \in T$ de durée de franchissement $l(t_x) = 1$;
2. A chaque arête $e = (x, y) \in E$ correspond deux places $p_1 = (t_x, t_y)$ et $p_2 = (t_y, t_x)$;
3. La valeur cible du problème DÉBIT MAX - SURFACE MIN est $K = L + 1$.

A présent, si o est une orientation des arêtes de $\mathcal{G}$, nous pouvons associer un marquage initial minimalement borné pour le graphe d'événements temporisé G en posant :

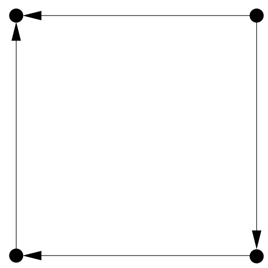
1. $M(0, p) \in \{0, 1\}$ pour toute place $p \in P$;
2. Pour chaque place $p = (t_x, t_y)$, $M(0, p) = 1$ si et seulement si $o(\{x, y\}) = (x, y)$.

On notera que la fonction f est une fonction bijective. De plus, chaque graphe d'événements temporisé G de marquage minimalement borné peut être associé avec exactement une seule orientation des arêtes de $\mathcal{G}$. Nous pouvons à présent établir une relation forte entre le flot ratio $\rho(o(\mathcal{G}))$ et le débit $\lambda(M(G))$.

Lemme 13. *Soient o une orientation des arêtes de $\mathcal{G}$ et M le marquage initial associé pour le graphe d'événements temporisés G. Nous avons alors :*

$$\lambda(M(G)) = \frac{1}{\rho(o(\mathcal{G})) + 1}$$

Démonstration. Soit c un cycle de $\mathcal{G}$. Alors, c est associé à deux circuits dans G ayant des directions opposés notés C_1 et C_2 (*voir* figure 6.3 page suivante).



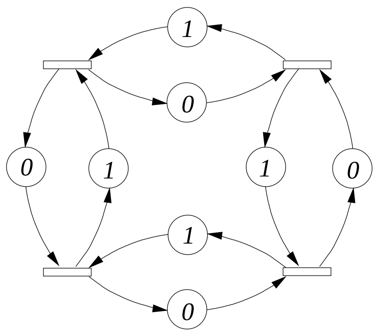


FIG. 6.3: Un cycle c est représenté sur le côté gauche de la figure. Le graphe d'événements symétrique marqué associé est décrit sur la droite de la figure.

Nous pouvons alors supposer sans perte de généralité qu'une direction des arcs de C_1 correspond aux $n_c(o)$ arcs de c. Nous posons $H(C_1) = n_c(o)$ et $H(C_2) = m_c(o)$. En outre, $L(C_1) = L(C_2) = n_c(o) + m_c(o)$.
Le débit λ_C du graphe d'événements symétrique composé uniquement de C_1 et C_2 vaut :

$$\begin{aligned}
\lambda_C &= \min\left(\frac{n_c(o)}{n_c(o)+m_c(o)}, \frac{m_c(o)}{n_c(o)+m_c(o)}\right) \\
&= \frac{1}{\max\left(\frac{n_c(o)+m_c(o)}{n_c(o)}, \frac{n_c(o)+m_c(o)}{n_c(o)}\right)} \\
&= \frac{1}{\max\left(\frac{m_c(o)}{n_c(o)}, \frac{n_c(o)}{m_c(o)}\right)+1}
\end{aligned}$$

Il s'ensuit que si ρ_c désigne le flot ratio du cycle c, nous avons $\lambda_C = \frac{1}{\rho_c+1}$. A présent, si c est un circuit de flot ratio maximum de $\mathcal{G}$, un des circuits associés C_1 ou C_2 possède le débit minimum du graphe d'événements G. Ce faisant, l'égalité établie dans ce lemme est vraie. □

Lemme 14. *f définit une transformation polynomiale des instances de* FLOT RATIO *vers les instances de* DÉBIT MAX - SURFACE MIN.

Démonstration. En effet, f peut être calculée en temps polynomial. La validité de la transformation découle du lemme 13. □

6.2.2 Complexité des problèmes FLOT RATIO et DÉBIT MAX - SURFACE MIN

Le résultat proposé par Minty dans [Min62] ayant été obtenu avant l'apparition de la théorie de la NP-complétude, nous nous proposons de montrer que le problème FLOT RATIO est effectivement NP-COMPLET.

Théorème 21. *Le problème* FLOT RATIO *est* NP-COMPLET.

Démonstration. Flot ratio appartient à la classe NP. En effet, le flot ratio d'une orientation des arêtes peut se calculer facilement à partir du débit du graphe d'événements marqué symétrique associé (en utilisant l'algorithme de Karp [Kar78] par exemple). De là, on vérifie si la contrainte sur le flot ratio de cette orientation est vérifiée en calculant le flot ratio par la formule du lemme 13.

Par ailleurs, le lemme de Minty [Min62] peut être vue comme une réduction polynomiale du problème de K-Coloration avec Flot ratio. Le problème K-coloration étant NP-complet (*voir* [GJ79]), nous obtenons le théorème. □

Nous en déduisons la complexité de notre problème :

Théorème 22. Débit Max - Surface Min *est* NP-complet.

Démonstration. Débit Max - Surface Min appartient à la classe NP. En effet, le débit associé à un marquage peut être calculé en temps polynomial par le biais de l'algorithme de Karp [Kar78] par exemple. Il suffit alors de vérifier que les contraintes du problème sont toutes vérifiées. Cette vérification s'opère en temps polynomial.

La fonction f étant bijective, le couple de fonctions (f, f^{-1}) caractérise une réduction polynomiale entre le problème Flot ratio et Débit Max - Surface Min. Le résultat de complexité précédent permet alors de compléter la preuve. Ces deux problèmes sont en quelque sorte équivalents. □

6.2.3 Un algorithme simple pour le problème de marquage

Etonnamment, la réduction que nous venons de proposer suggère un algorithme alternatif simple pour le problème de marquage que nous avons défini au chapitre 5. En effet, nous avons vu que la construction du graphe associé permet de simplifier la recherche d'un marquage vivant. Le problème de marquage devient équivalent à déterminer une valuation particulière des arcs du graphe associé. Nous avons alors montré qu'il était possible de construire un marquage vivant à partir d'une valuation du graphe associé qui satisfait aux conditions du lemme 10 page 84.

Nous pouvons voir le problème de la construction d'une valuation du graphe associé comme une problème d'orientation d'arêtes dans un graphe simple. Pour cela, il suffit d'observer qu'une valuation admissible (*cf.* propriété 4 page 84) confère à tout circuit de taille 2 une valuation totale de ses arcs valant 1. Le graphe associé étant symétrique, cette contrainte peut être vue comme un choix de marquage pour chaque couple de places retours. De plus, le théorème 15 page 85 permet de caractériser la structure d'une valuation solution : le sous graphe des arcs de valuation nulle doit être acyclique. Nous observons alors que ce qui importe dans cette caractérisation est davantage la structure du sous graphe que sa valuation en elle même. Nous allons montrer à présent que pour obtenir une telle structure (et par conséquent une valuation et donc un marquage), il n'est pas nécessaire de travailler sur le graphe symétrique associé au graphe à capacité minimum. L'algorithme que nous proposons ici construit directement un marquage vivant minimalement borné sans construire au préalable une valuation du graphe associé.

Nous nous appuyons ici sur l'application 1 page 81 qui décrit avec précision la condition suffisante sur les graphes à capacités minimum :

Application 1. [Rappel] *Soient $G = (T, P)$ un graphe borné et $G_R^{min} = (T, P_R, M_0^\star)$ un graphe à capacité minimum (normalisé). Supposons que pour tout circuit C de G_R^{min}, l'inégalité*

$$\sum_{p \in C \cap P_R} v(p) > \sum_{p \in C \cap P_R} M_0^\star(p) > \sum_{p \in C \cap P_R} (v(p) - pgcd_p)$$

soit satisfaite, alors G_R^{min} est vivant.

Nous allons montrer que l'on peut obtenir un graphe à capacité minimum d'une autre manière. Pour cela, considérons une instance du problème de marquage. Soit $G = (T, P)$ un graphe d'événements généralisé (on supposera sans perte de généralité que le graphe d'événements généralisé G est générique, *voir* définition 14 page 80). On construit alors un graphe non orienté $\mathcal{G} = (T, E)$ comme suit :

1. Les sommets de $\mathcal{G}$ sont les transitions de G.
2. Chaque place $p = (t_i, t_j)$ donne lieu à une arête (t_i, t_j) dans $\mathcal{G}$. On notera que le graphe $\mathcal{G}$ n'est pas forcément un graphe simple du fait de l'existence potentielle de couples de places retours dans G.

Nous construisons à présent une orientation des arêtes de $\mathcal{G}$. Nous allons déduire de cette orientation un graphe à capacité minimum.

1. Les sommets de $\mathcal{G}$ sont numérotés de 1 à $|T|$. On notera $n(t_i)$ le nombre attribué au sommet t_i. On définit une orientation des arêtes telle qu'une arête (t_i, t_j) est orientée de t_i vers t_j si et seulement si $n(t_j) > n(t_i)$.
2. Pour chaque arc (t_i, t_j) on marque la place $p = (t_i, t_j)$ du graphe à capacité de G par $v(p) - pgcd_p$ jetons. On définit ensuite le marquage initial de sa place retour $p' = (t_j, t_i)$ en posant $M_0(p') = v(p')$ jetons.

Nous en déduisons le théorème suivant :

Théorème 23. *L'algorithme précédent résout en $\Theta(m)$ le problème de marquage.*

Démonstration. La validité de l'algorithme repose sur la définition de l'orientation des arcs. Nous démontrons cet algorithme par un raisonnement par contradiction. Remarquons dans un premier temps que l'algorithme marque chaque place p du graphe à capacité par $v(p)$ ou $v(p) - pgcd_p$ jetons. De plus, comme nous l'avions remarqué à la sous-section 5.5.1 page 86 du chapitre 5, la normalisation n'est pas une étape nécessaire ici. En effet, la normalisation consistant à multiplier les attributs d'une place par un entier k, les marquages construits par l'algorithme pour une place p du graphe normalisé sont eux dans $\{k \cdot v(p) - pgcd(k \cdot w(p), k \cdot v(p)), k \cdot v(p)\}$ soit $\{k \cdot (v(p) - pgcd_p), k \cdot v(p)\}$.

Supposons que l'algorithme construit un marquage qui ne soit pas vivant. Compte tenu de la remarque précédente, on en déduit qu'il existe un circuit C dans le graphe à capacité de G pour lequel chaque place p est marquée par $v(p) - pgcd_p$ jetons. Cela signifie que l'orientation que nous avons défini pour le graphe contenait un circuit. Or la numérotation des sommets dont est issue cette orientation correspond également à un tri topologique des sommets du graphe orienté $\mathcal{G}$, nous obtenons une contradiction.

De même, pour tout couple de places retours (p, p'), l'algorithme engendre un circuit dont le marquage initial vérifie :

$$M_0(p) + M_0(p') = w(p) + v(p) - pgcd_p$$

Le marquage ainsi défini est donc minimalement borné. Nous avons donc bien construit un graphe à capacité minimum.

D'autre part, l'algorithme définit une numérotation en $\Theta(n)$ étapes élémentaires. L'algorithme construit ensuite une orientation et en même temps un marquage initial en $\Theta(m)$ étapes élémentaires. □

6.2.4 Exemple

Nous illustrons sur un exemple notre nouvel algorithme pour le problème de marquage. Reprenons l'exemple traité dans ce manuscrit. Initialement, nous construisons le graphe $\mathcal{G}$ qui donne en quelque sorte l'ossature du graphe d'événements généralisé G. Nous numérotons ensuite les sommets de 1 à $|T| = 5$. Ces deux étapes sont exposées sur la figure 6.4 page suivante.

Nous construisons ensuite les arcs de $\mathcal{G}$ en orientant chaque arête du sommet de plus petit indice vers le sommet d'indice supérieur (*voir* figure 6.5 c) page suivante). De là, nous attribuons le marquage initial du graphe à capacité de G conformément aux instructions de l'algorithme (*voir* figure 6.5 d)). Nous obtenons alors un graphe à capacité minimum.

6.3 Résultats nouveaux pour le cas non-marqué

Le problème DÉBIT MAX - SURFACE MIN est un problème clé pour la démonstration des problèmes d'optimisation bi-critère sur les graphes d'événements (généralisés ou non). Ce problème va nous servir dans toutes les réductions auxquelles nous allons procéder. Nous établissons dans un premier temps la complexité théorique d'un problème d'optimisation voisin. Nous déterminons ensuite la complexité du problème OPTIMISATION DE MARQUAGE et enfin celle du problème DÉBIT MAXIMUM INTRINSÈQUE.

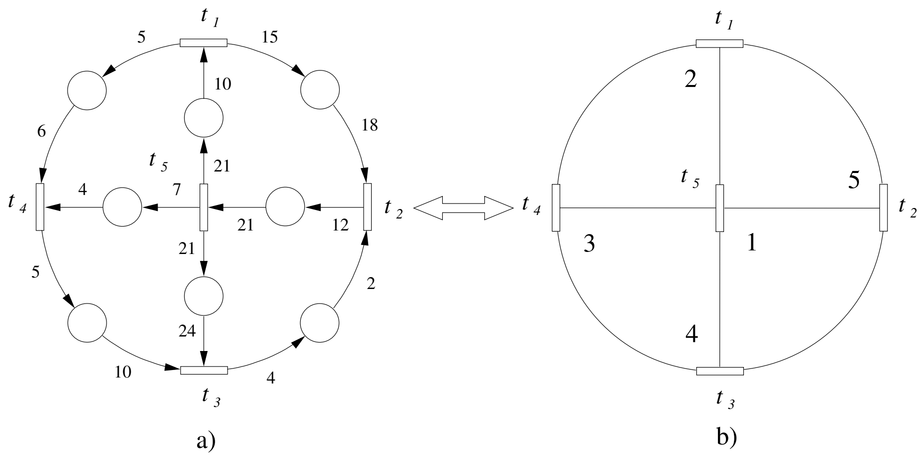


FIG. 6.4: A gauche le graphe d'événements généralisé G. Sur la droite, nous représentons le graphe $\mathcal{G}$ ainsi qu'une numérotation des sommets.

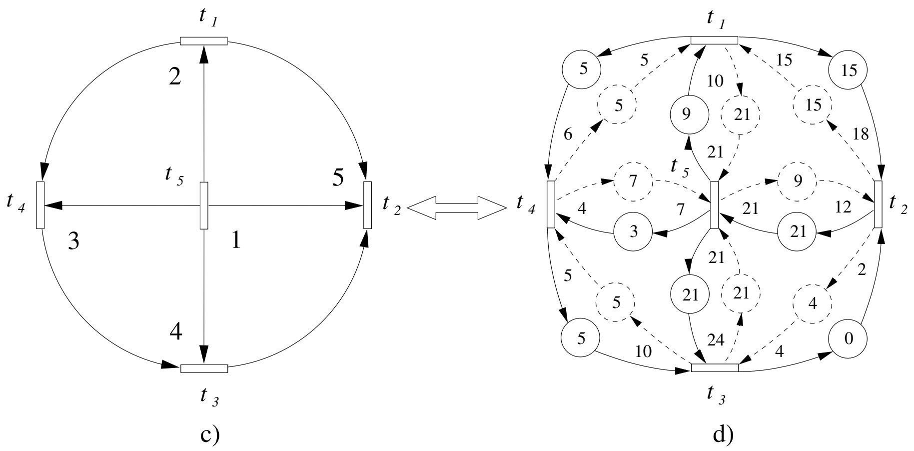


FIG. 6.5: A gauche le graphe $\mathcal{G}$ muni de son orientation. Le graphe à capacité minimum est décrit sur la droite.

6.3.1 Complexité du problème Conception de marquages

Nous définissons le problème bi-critère Conception de marquages comme suit :

Conception de marquages :

Instance : Soient $G = (T, P)$ un graphe d'événements temporisé, un réel $Q \in [0, 1]$ et un entier K tel que $K \geqslant |P|$.

Question : Existe-t-il un marquage initial $M(G)$ tel que :

$$\sum_{p \in P} M(0,p) \leqslant K \qquad \text{et} \qquad \lambda(M(G)) \geqslant Q$$

Le problème Conception de marquages est une généralisation du problème Débit

Max - Surface Min. Les instances de ce problème ne sont plus aussi contraintes.

Théorème 24. Conception de marquages *est* NP-complet.

Démonstration. Le problème Conception de marquages appartient à la classe NP. En effet, étant donné une solution proposée, il est possible de vérifier que le débit du système satisfait à la contrainte du problème par un algorithme de calcul du poids moyen minimum d'un graphe (*cf.* [DIG99]). On vérifie aisément la contrainte sur la taille de ce marquage. De plus, il existe une réduction polynomiale immédiate entre Débit Max - Surface Min et ce problème. D'où le résultat. □

Il est facile d'observer que la réduction précédente demeure toujours valable même lorsque le graphe d'événements temporisé est symétrique.

6.3.2 Complexité du problème Optimisation de marquage

Le problème Optimisation de marquage a été introduit par Laftit et Proth [LPX92].

Optimisation de marquage :

Instance : Soient $G = (T, P, l)$ un graphe d'événements temporisé , Y est un P-semiflot de G, $N \in \mathbb{N}^\star$ et $Q \in [0, 1]$.

Question : Existe-t-il un marquage initial $M(G)$ tel que :

$$^{T}Y \cdot M(0,p) \leqslant N \qquad \text{avec} \qquad \lambda(M(G)) \geqslant Q$$

Ce problème est important pour l'analyse et la conception des systèmes manufacturiers. La pondération du P-semiflot sur le marquage initial reflète un coût sur les encours circulant entre les différents ateliers. Il convient alors de minimiser le coût total tout en ayant une cadence de production maximale.

Nous pouvons déduire du théorème 24 le résultat suivant :

Théorème 25. *Le problème* Optimisation de marquage *est* NP-complet.

Démonstration. Optimisation de marquage appartient à la classe NP car étant donné un solution proposée, on peut calculer en temps polynomial le débit du système et vérifier les contraintes sur le marquage.

On prouve que le problème Débit Max - Surface Min est un sous-problème de Optimisation de marquage. Une instance I du problème Débit Max - Surface Min est définie par un graphe d'événements temporisé G et un entier K. D'autre part, G étant symétrique, nous avons :

$$|\mathcal{P}^+(t)| = |\mathcal{P}^-(t)|, \ \forall t \in T$$

Nous en déduisons que chaque colonne de la matrice d'incidence du graphe a exactement autant de valeurs positives que de valeurs négatives. Le graphe étant un graphe d'événements, on a également :

$$w(p) = v(p) = 1 \ , \quad \forall p \in P$$

Par conséquent le vecteur unitaire $1^{|P|}$ est un P-semiflot de la matrice d'incidence de G. En outre, en posant $N = \frac{|P|}{2}$, la contrainte ${}^{\intercal}Y \cdot M(0,p) \leqslant N$ est équivalente à imposer que M doit être un marquage minimalement borné. Enfin, en posant $Q = \frac{1}{K}$ nous déduisons que I est une instance de OPTIMISATION DE MARQUAGE. □

6.3.3 Complexité du problème DÉBIT MAXIMUM INTRINSÈQUE

Nous étudions à présent le problème de la minimisation du nombre total de jetons d'un marquage initial sous contrainte que ce marquage puissent atteindre un débit maximum intrinsèque *i.e.* il existe une transition qui est effectuée sans discontinuer. Dans ce cas, le circuit critique est une place boucle sur une transition t_i ayant une durée de franchissement maximum. Nous définissons d'abord le problème de décision DÉBIT MAXIMUM INTRINSÈQUE.

DÉBIT MAXIMUM INTRINSÈQUE :

Instance : Soient $G = (T, P, l)$ un graphe d'événements temporisé symétrique et $Z \in \mathbb{N}^{\star}$.

Question : Existe-t-il un marquage initial vivant $M(G)$ tel que :

$$\sum_{p \in P} M(0,p) \leqslant Z \qquad \text{et} \qquad \lambda(M(G)) \geqslant \frac{1}{\max\limits_{t_j \in T}(l(t_j))}$$

Ce problème a été abordé pour des instances du problème général *i.e.* où la structure du graphe d'événements temporisé n'est pas forcément symétrique [HP89, Gau90, DFMS97, GPS02]. Comme précédemment, le résultat suivant de NP-complétude reste vrai lorsqu'aucune hypothèse n'est faite sur la structure du graphe d'événements temporisé.

Nous montrons que le problème DÉBIT MAXIMUM INTRINSÈQUE est NP-COMPLET.

Théorème 26. *Le problème* DÉBIT MAXIMUM INTRINSÈQUE *est* NP-COMPLET.

Démonstration. Le problème appartient à la classe NP. En effet, on vérifie facilement que le nombre de jetons d'un marquage donné vérifie la contrainte associée. De même, en utilisant un algorithme de poids moyen minimum, on détermine le débit du système et on vérifie ensuite la contrainte associée au débit. Nous exhibons une réduction de DÉBIT MAX - SURFACE MIN vers DÉBIT MAXIMUM INTRINSÈQUE. Une instance de DÉBIT MAX - SURFACE MIN est définie par un graphe d'événements temporisé symétrique $G = (T, P, l)$ et une valeur cible K. Nous construisons alors une instance de DÉBIT MAXIMUM INTRINSÈQUE associée à cette instance de la manière suivante :

1. Nous construisons un autre graphe d'événements temporisé symétrique en ajoutant une transition $t^\star$ avec $l(t^\star) = K$ et un couple de places retours définies entre $(t, t^\star)$ et $(t^\star, t)$ pour une transition $t \in T$.
2. On pose ensuite $Z^\star = \frac{|P|}{2} + 2$. On remarque alors que le débit maximum intrinsèque de ce nouveau graphe d'événements temporisé vaut $\frac{1}{K}$.

- Si le marquage M est une solution pour l'instance de Débit Max - Surface Min, nous pouvons construire une solution $M^\star$ pour l'instance associée de Débit maximum intrinsèque en ajoutant 1 jeton sur chaque place nouvellement crée. Le nombre total des jetons initiaux vaut alors $Z^\star = \frac{|P|}{2} + 2$. A présent, si $c^\star$ est un circuit critique de $\mathcal{G}^\star$, alors d'après le théorème 19, nous pouvons considérer que $c^\star$ est un circuit élémentaire. Nous distinguons à présent deux cas :
  (a) Si le circuit $c^\star$ est inclus dans $\mathcal{G}$, alors $\lambda(M^\star(\mathcal{G}^\star)) \geq \frac{H(c^\star)}{L(c^\star)} \geq \frac{1}{K}$;
  (b) Sinon, le circuit $c^\star$ étant élémentaire, nous déduisons qu'il est composé des transitions t et $t^\star$ et par conséquent :
  $$\frac{H(c^\star)}{L(c^\star)} = \frac{2}{K+1} > \frac{1}{K}$$
  Il en résulte que le marquage $M^\star$ est bien une solution de l'instance correspondante Débit maximum intrinsèque.
- Supposons à présent qu'on dispose d'un marquage $M^\star$ solution de l'instance de Débit maximum intrinsèque proposée. En posant $M(0,p) = M^\star(0,p)$ pour toute les places $p \in P$, nous prouvons que M est également une solution de l'instance correspondante du problème Débit Max - Surface Min.

  Par définition de M et $M^\star$, $\lambda(M(\mathcal{G})) \geq \lambda(M^\star(\mathcal{G}^\star)) \geq \frac{1}{K}$.
  Supposons que c soit un circuit de $\mathcal{G}^\star$ composé des transitions t et $t^\star$. Le marquage $M^\star$ étant un marquage vivant, $H(c) \geq 1$. Il s'ensuit que :
  $$\frac{H(c)}{L(c)} = \frac{H(c)}{K+1} \geq \frac{1}{K}$$
  Nous avons alors :
  $$H(c) \geq 1 + \frac{1}{K}$$
  Par conséquent, le circuit c doit avoir au moins deux jetons sur l'ensemble de ces places.
  Le nombre total de jetons présents dans les places de $\mathcal{G}$ est donc majoré par $\frac{|P|}{2}$. Le marquage $M^\star$ étant vivant, nous en déduisons que M est vivant et par conséquent pour tout couple de places retours (p, p') nous avons :
  $$M(0,p) + M(0,p') \geq 1$$

Il y a alors exactement un jeton par couple de places retours. Le marquage M est donc minimalement borné et constitue bien une solution au problème Débit Max - Surface Min.

□

De nouveau, le problème pour le cas généralisé n'étant pas dans la classe NP, nous en déduisons que le problème Débit maximum intrinsèque pour des graphes d'événements généralisés est NP-difficile. Néanmoins, nous proposons au chapitre suivant un algorithme 2-approché pour ce problème.

Conclusion

Nous avons présenté dans ce chapitre des résultats de complexité nouveaux pour des problèmes d'optimisations bi-critères. Pour cela, nous avons relié ces problèmes à un problème d'orientation des arêtes d'un graphe non orienté. La réduction polynomiale que nous avons établie avec notre problème central de complexité (*i.e.* le problème Débit Max - Surface Min) met en lumière le lien avec le problème de coloration des sommets d'un graphe. Cette relation forte pourra être exploitée pour mettre au point des heuristiques de résolution du problème Débit Max - Surface Min.

D'autre part, l'analyse du problème d'orientation d'arêtes de Minty nous a suggéré un algorithme performant pour le problème de marquage. La complexité de l'algorithme que nous avons développé est sensiblement moins importante que celle de la méthode du chapitre 5.

De plus, nous avons également établi la complexité théorique de deux problèmes d'optimisation importants :

1. le problème Optimisation de marquage ;
2. le problème Débit maximum intrinsèque.

Ces deux derniers sont des problèmes classiques en automatique. Cependant les complexités théoriques de ces problèmes restaient encore inconnues.

Chapitre 7

Borne supérieure

Sommaire

Dans ce chapitre, nous nous intéressons au problème d'optimisation bi-critère Débit maximum intrinsèque. Au cours du chapitre précédent, nous avons démontré la NP-complétude de ce problème. Cependant, une analyse approfondie de ce problème s'avère utile. En effet, le comportement optimal d'un système à événements discrets (*i.e.* le débit) est un critère important pour les concepteurs de systèmes embarqués ou de chaînes d'assemblage. De plus, dans un contexte d'optimisation des ressources, une bonne caractérisation d'une solution à ce problème peut servir de base pour la mise au point d'heuristiques pour le problème Optimisation de marquage.

Après avoir rappelé quelques définitions et notations, nous présentons le problème traité dans ce chapitre. Nous étudions ensuite le problème Débit maximum intrinsèque dans le cas des graphes d'événements temporisés symétriques. Nous montrons qu'il existe un algorithme polynomial simple permettant de résoudre des instances particulières puis nous en déduisons que ce même algorithme est 2-approché pour des instances quelconques. Dans une troisième partie, nous établissons une série de résultats intermédiaires afin d'étendre les deux résultats de la section précédente aux problèmes avec instances généralisées. Enfin, nous proposons à la section 4 une généralisation de ces deux résultats.

7.1 Présentation du problème et notations

Dans cette section, nous rappelons quelques propriétés importantes exposées au cours des chapitres précédents puis nous présentons le problème traité dans ce chapitre.

7.1.1 Notations du chapitre

Nous rappelons dans un premier temps la définition du marquage instantané. Le marquage instantané d'une place $p = (t_i, t_j)$ à la date $\tau \in \mathbb{R}^+$ correspond au nombre de jetons contenus dans la place p à la date τ. Le marquage instantané d'une place $p = (t_i, t_j)$ est alors noté $M(\tau, p)$. Si on considère à présent un ordonnancement $S = \{s(t_i, q), \forall (t_i, q) \in T \times \mathbb{N}^\star\}$, on peut définir le marquage instantané d'une place p à une date τ en fonction des nombres de franchissements entamés et accomplis à cette date (respectivement notés $B(\tau, p)$ et $E(\tau, p)$). Soit $E(\tau, t_i)$ le nombre de franchissements de t_i terminés à la date τ :

$$E(\tau, t_i) = max\{q \in \mathbb{N},\, s(t_i, q) + l(t_i) \leqslant \tau\}$$

De même, $B(\tau, t_j)$ désigne le nombre de franchissements de la transition t_j entamés à la date τ, soit :

$$B(\tau, t_j) = max\{q \in \mathbb{N},\, s(t_j, q) \leqslant \tau\}$$

Le marquage instantané de la place p à la date τ vaut alors :

$$M(\tau, p) = M(0, p) + w(p) \cdot E(\tau, t_i) - v(p) \cdot B(\tau, t_j) \quad \forall p = (t_i, t_j) \in P$$

Nous dirons qu'un ordonnancement est valide si $M(\tau, p) \geqslant 0$ pour tout couple $(\tau, p) \in \mathbb{R}^{+\star} \times P$.

Par le biais de la normalisation, nous avons proposé une extension de la notion de débit définie par Chrétienne [Chr83] dans le cas non-généralisé pour des instances généralisées :

Théorème 20. [Rappel] *Soient G un graphe d'événements généralisé temporisé et $M(G)$ un marquage initial vivant. Alors, l'ordonnancement au plus tôt est K-périodique et pour tout couple de transitions $(t_i, t_j) \in T^2$, nous avons :*

$$\lambda(t_i) \cdot Z_i = \lambda(t_j) \cdot Z_j \stackrel{def}{=} \lambda(M(G))$$

La valeur $\lambda(M(G))$ est appelée le débit de G associé au marquage initial $M(G)$.

De plus, nous avons introduit la notion de débit maximum intrinsèque qui caractérise la limite supérieure de fonctionnement d'un graphe d'événements généralisé temporisé :

Définition 16. [Rappel] *Le débit maximum intrinsèque d'un graphe d'événements généralisé temporisé unitaire G vaut $\min_{t_i \in T} \left(\frac{Z_i}{l(t_i)} \right)$. Cette valeur de débit correspond au débit maximum atteignable par un marquage initial $M(G)$.*

7.1.2 Présentation du problème

Au cours du chapitre précédent, nous avons établi la complexité du problème de décision Débit maximum intrinsèque. Naturellement, ce problème reste également intraitable lorsque nous ajoutons à l'instance une fonction de coût croissante sur les places. Dans ce chapitre, nous nous intéressons à la résolution du problème dans le cas généralisé. Ce problème est alors appelé Débit maximum intrinsèque - Généralisé et se formule ainsi :

Débit maximum intrinsèque - Généralisé :

Instance : Soient $G = (T, P, l)$ un graphe d'événements généralisé temporisé symétrique normalisé, un vecteur positif $x = (x_1, \ldots, x_m)$ tel que $x_p = x_{p'}$ pour tout couple de places retours (p, p') et $Z \in \mathbb{N}^\star$.

Question : Existe-t-il un marquage initial vivant $M(G)$ tel que :

$$\sum_{p \in P} x_p \cdot M(0, p) \leqslant Z \qquad \text{et} \qquad \lambda(M(G)) \geqslant \min_{t_i \in T} \left(\frac{Z_i}{l(t_i)} \right)$$

Une des grandes difficultés associées à ce problème de décision est l'existence d'une méthode polynomiale pour évaluer le débit d'une solution proposée. En effet, pour le moment, nous ne connaissons pas d'algorithme polynomial d'évaluation du débit d'un système marqué. Les méthodes connues à ce jour sont toutes de complexité pseudo-polynomiale (*voir* [Mun93]). Dans un contexte pratique d'optimisation, cet aspect théorique du problème représente un sérieux obstacle.

Au cours des sections suivantes, nous mettons progressivement au point un algorithme polynomial 2-approché pour ce problème. Pour cela, nous nous concentrons dans un premier temps sur la résolution des instances du problème pour lesquelles nous disposons d'un graphe d'événements non-généralisé. Par la suite, nous généraliserons ces résultats.

7.2 Le cas simple

Dans cette section, nous nous concentrons sur les instances non-généralisées. Nous énonçons d'abord un résultat d'optimalité pour des instances encore plus restreintes puis nous décrivons un algorithme 2-approché pour le problème de cette section.

7.2.1 Résultat d'optimalité

Le résultat suivant montre qu'il existe un algorithme polynomial pour des instances particulières du problème Débit maximum intrinsèque - Généralisé dans le cas non-généralisé.

Théorème 27. *Le problème* Débit maximum intrinsèque - Généralisé *est polynomial pour un graphe d'événements temporisé non-généralisé symétrique ayant des durées de franchissement de transitions égales.*

Démonstration. Nous montrons que le marquage initial $M(0,p) = 1, \forall p \in P$ est une solution au problème Débit maximum intrinsèque - Généralisé. En effet, en supposant que la durée de franchissement des transitions est définie par $l(t_i) = R, \quad \forall t_i \in T$, nous avons :

$$\frac{1}{\max\limits_{t_j \in T}(l(t_j))} = \frac{1}{R} \quad \text{ainsi que} \quad \lambda(M(G)) \geq \frac{1}{R}$$

Le graphe d'événements temporisé $\mathcal{G}$ étant symétrique, chaque couple de places retours doit avoir au moins deux jetons répartis sur chacune de ses places pour atteindre le débit maximum intrinsèque de $\mathcal{G}$. Nous en déduisons que $|P|$ est une borne inférieure du nombre total de jetons contenus dans le marquage initial d'une solution du problème.

De même, $\sum\limits_{p \in P} x_p$ est une borne inférieure de la valeur de la fonction objectif pour toute solution. D'où le résultat. $\square$

7.2.2 Résultat d'approximation

Nous pouvons nous demander ce qu'il advient de la solution provenant de cet algorithme lorsque l'hypothèse restrictive sur les durées des transitions n'est plus vérifiée. L'exemple présenté sur la figure 7.1 décrit une instance pour laquelle il n'est pas nécessaire de mettre un jeton sur chaque place pour atteindre le débit maximum intrinsèque.

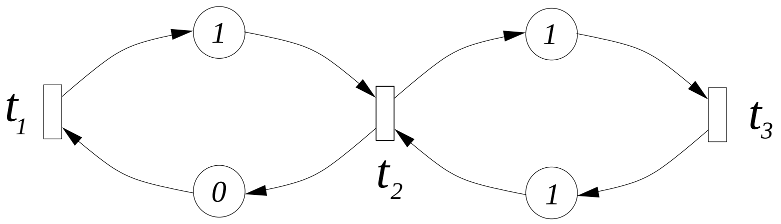


FIG. 7.1: Le graphe d'événements temporisé symétrique possède trois transitions ayant des durées de franchissement $l(t_1) = 1$, $l(t_2) = 2$ et $l(t_3) = 3$. Le marquage initial représenté atteint bien le débit maximum intrinsèque du système, soit ici $\frac{1}{3}$.

L'algorithme permet cependant de définir une borne supérieure intéressante du nombre minimum de jetons qu'il est nécessaire de mettre pour atteindre le débit maximum. Nous déduisons le corollaire suivant :

Corollaire 3. *En posant $M(0,p) = 1$ pour toute place p, nous obtenons une solution 2-approchée pour le problème* DÉBIT MAXIMUM INTRINSÈQUE - GÉNÉRALISÉ *pour un graphe d'événements temporisé non-généralisé symétrique.*

Démonstration. On remarque que tout circuit C contient exactement $|C|$ jetons (avec $|C|$ désignant le nombre de places contenues dans le circuit C). Nous en déduisons que le système est vivant. D'autre part, pour tout circuit C, nous avons :

$$\frac{\sum\limits_{p \in C} M(0,p)}{\sum\limits_{t_i \in C} l(t_i)} = \frac{|C|}{\sum\limits_{t_i \in C} l(t_i)} \geqslant \frac{|C|}{|C| \cdot \max\limits_{t_i \in T}(l(t_i))}$$

$$\text{soit} \qquad \frac{H(C)}{L(C)} \geqslant \frac{1}{\max\limits_{t_i \in T}(l(t_i))}$$

Nous en déduisons que le débit $\lambda(M(G))$ vaut $\frac{1}{\max_{t_i \in T}(l(t_i))}$ et par conséquent le marquage proposé satisfait bien à la contrainte du débit maximum intrinsèque. Par ailleurs, le graphe étant symétrique, il convient de mettre pour chaque couple de places retours $(p, p') \in P^2$ au moins un jeton sur l'une de ces places pour garantir la vivacité de chaque circuit à deux transitions. Il en résulte qu'une borne inférieure à toute solution est $\frac{|P|}{2}$. Le marquage défini par l'algorithme utilise $|P|$ jetons. De plus, comme pour tout couple de places retours (p, p') nous avons $x_p = x_{p'}$, nous en déduisons le ratio d'approximation. Ceci complète la preuve. $\square$

Nous notons que le marquage défini pour une place p par l'algorithme vaut $M(0,p) = 1$, soit également $M_{\min}(p)$. Le débit maximum intrinsèque d'un graphe d'événements temporisé est donc fortement lié au marquage minimum vivant de ce graphe. Dans la section suivante, nous montrons que ces deux résultats peuvent être étendus dans le cas d'instances généralisées.

7.3 Généralisation

Dans cette section, nous nous proposons de généraliser le résultat d'optimalité exposé à la section précédente. Nous considérons pour cela les instances DÉBIT MAXIMUM INTRINSÈQUE - GÉNÉRALISÉ pour lesquelles le graphe d'événements généralisé temporisé symétrique normalisé (GEGTS en abrégé) a des durées de franchissement telles que :

$$\frac{Z_i}{l(t_i)} = \rho \qquad \forall t_i \in T$$

Nous montrons qu'en posant $M(0,p) = M_{\min}(p)$ pour toute place p, nous obtenons une solution optimale au problème.

Nous prouvons d'abord que cette solution est bien réalisable. Nous établissons par la suite l'optimalité de cette solution.

7.3.1 Admissibilité de la solution

Nous démontrons que la solution $M(0,p) = M_{\min}(p)$ pour toute place p est effectivement une solution du problème.

Lemme 15. *Soit $G = (T, P, l)$ un GEGTS tel que :*

$$\forall t_i \in T, \quad \frac{l(t_i)}{Z_i} = \rho \qquad et \qquad M(0,p) = M_{\min}(p)\,,\ \forall p \in P$$

L'ordonnancement périodique $S^\star$ défini par :

$$S^\star = \{s^\star(t_i,k) = (k-1)\cdot l(t_i), \quad \forall (t_i,k) \in T \times \mathbb{N}^\star\}$$

est un ordonnancement valide et son débit est maximum intrinsèque.

On remarque que le marquage défini ne dépend que des fonctions d'écriture et de lecture des places. Comme pour les instances non-généralisées, la structure du graphe n'a pas d'incidence sur le problème.

Démonstration. Nous proposons une preuve de la validité de l'ordonnancement périodique $S^\star$. Soit $p = (t_i, t_j)$ une place du GEGTS G (*voir* figure 7.2 page suivante). On considère le marquage instantané de cette place à la date $\tau \in \mathbb{R}^+$. Le graphe étant normalisé, nous

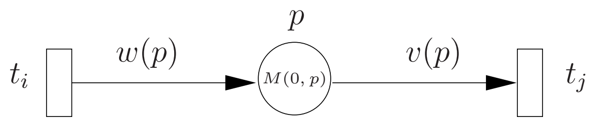


FIG. 7.2: Une place $p = (t_i, t_j)$ de G.

pouvons supposé sans perte de généralité que $Z_i = w(p)$ et $Z_j = v(p)$. L'ordonnancement $S^\star$ étant périodique et de par l'hypothèse sur les durées de franchissement (*i.e.* $l(t_k) = \rho Z_k$ avec $k \in \{1, 2\ \}$), nous en déduisons que le nombre de franchissements accomplis de la transition t_i à la date τ vaut :

$$E(\tau, t_i) = \left\lfloor \frac{\tau}{l(t_i)} \right\rfloor = \left\lfloor \frac{\tau}{\rho \cdot w(p)} \right\rfloor$$

De même, le nombre de franchissements entamés par la transition t_j à la date τ vaut :

$$B(\tau, t_j) = \left\lceil \frac{\tau}{l(t_j)} \right\rceil = \left\lceil \frac{\tau}{\rho \cdot v(p)} \right\rceil$$

Nous pouvons alors exprimer le marquage instantané d'une place $p = (t_i, t_j)$ par les équations suivantes :

$$\begin{cases} M(0,p) & = & M_{\min}(p) & = & w(p) + v(p) - pgcd_p \\ M(\tau,p) & = & M(0,p) + \left\lfloor \frac{\tau}{\rho \cdot w(p)} \right\rfloor \cdot w(p) - \left\lceil \frac{\tau}{\rho \cdot v(p)} \right\rceil \cdot v(p) & & \forall \tau > 0 \end{cases}$$

Montrons à présent que la fonction $M(\tau, p)$ caractérisant le marquage instantané de cet ordonnancement est positive. Nous adoptons ici un raisonnement par contradiction. Supposons qu'il existe un couple $(\tau, p) \in \mathbb{R}^{+\star} \times P$ tel que $M(\tau, p) < 0$.

Comme la fonction $M(\tau, p)$ est une combinaison linéaire des fonctions d'écriture et de lecture $w(p)$ et $v(p)$ nous avons :

$$M(\tau, p) \equiv 0 \bmod pgcd_p$$

Il résulte de cette observation que l'hypothèse que nous avons faite précédemment se reformule ainsi :

$$M(\tau, p) < 0 \Longleftrightarrow M(\tau, p) \leqslant -pgcd_p$$

En remplaçant $M(\tau, p)$ par son expression analytique, nous obtenons :

$$M(0,p) + \left\lfloor \frac{\tau}{\rho \cdot w(p)} \right\rfloor \cdot w(p) - \left\lceil \frac{\tau}{\rho \cdot v(p)} \right\rceil \cdot v(p) \leqslant -pgcd_p$$

D'après les hypothèses du théorème, nous avons aussi $M(0, p) = M_{\min}(p) = w(p) + v(p) - pgcd_p$. On a donc également :

$$w(p) + v(p) + \left\lfloor \frac{\tau}{\rho \cdot w(p)} \right\rfloor \cdot w(p) \leqslant \left\lceil \frac{\tau}{\rho \cdot v(p)} \right\rceil \cdot v(p)$$

Or, pour tout réel x, nous avons les inégalités suivantes :

$$x - 1 < \lfloor x \rfloor \leqslant x \leqslant \lceil x \rceil < x + 1$$

Nous obtenons alors :

$$w(p) + v(p) + \frac{\tau}{\rho} - w(p) < \frac{\tau}{\rho} + v(p)$$

soit,

$$0 < 0$$

D'où, une contradiction. L'ordonnancement $S^\star$ défini bien un ordonnancement valide pour ce type de GEGTS. D'autre part, la structure de l'ordonnancement et les durées des transitions sont telles qu'il n'y a pas de temps d'oisiveté entre deux franchissements consécutifs d'une transition. Les exécutions des transitions sont donc contiguës et par conséquent :

$$\lambda(t) = \frac{1}{l(t)}, \ \forall t \in T$$

Le débit du système est alors maximum intrinsèque. □

7.3.2 Résultats intermédiaires

Afin de démontrer que la solution proposée au lemme 15 page 118 est bien optimale, nous allons établir quelques résultats intermédiaires importants sur les instances à deux transitions.

Soit G_{p_1,p_2} un GEGTS normalisé à deux transitions t_i et t_j tel que $\frac{Z_i}{l(t_i)} = \frac{Z_j}{l(t_j)}$. Supposons à présent que l'on dispose d'un marquage initial suffisamment important qui permette d'atteindre le débit maximum intrinsèque du graphe G_{p_1,p_2}.

Nous démontrons dans un premier temps que pour tout ordonnancement valide qui atteint ce débit, il existe une date $\tau^\star$ à partir de laquelle les dates de franchissements des transitions sont périodiques.

Lemme 16. *Etant donné un marquage initial qui permet de définir un ordonnancement valide S de débit maximum intrinsèque, il existe une date $\tau^\star$ telle que quelle que soit la transition $t \in \{t_i, t_j\}$, il existe un entier k_t tel que $s(t,k) = \tau^\star + (k - k_t) \cdot l(t)$*

Démonstration. D'après le théorème 20 rappelé à la page 115, l'ordonnancement au plus tôt d'un tel système est K-périodique. Ayant supposé que le système avait un débit maximum intrinsèque, et compte tenu des durées de franchissement des transitions t_i et t_j, il n'existe aucun temps d'oisiveté entre deux franchissements successifs d'une même transition. Les dates de début de chaque transition deviennent périodiques.

Nous procédons par un raisonnement par contradiction afin de démontrer qu'il existe une date $\tau^\star$ à partir de laquelle les dates de début des transitions sont définies par $s(t,k) =$

$\tau^\star + (k - k_t) \cdot l(t), \quad \forall t \in (t_i, t_j)$.
Supposons donc qu'il n'existe aucune date $\tau^\star$. On considère alors pour chaque transition, la date à partir de laquelle les dates de début deviennent périodiques. Notons $\tau_{\max}$ la plus grande de ces deux dates. Soit ν l'occurrence de la transition franchie à la date $\tau_{\max}$. Nous en déduisons qu'il existe un trou dans l'ordonnancement entre la date de fin de la $(\nu - 1)$-ième occurrence de cette transition et la date de franchissement de l'occurrence suivante. Il s'ensuit que la ν-ième occurrence n'a pas été ordonnancée au plus tôt (*voir* figure 7.3). D'où une contradiction.

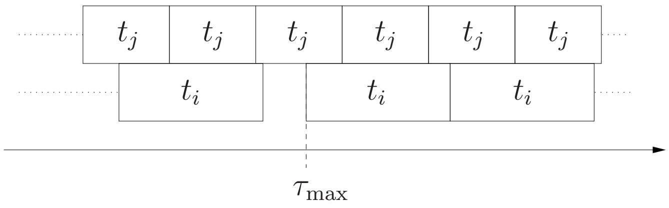


FIG. 7.3: Cas d'un ordonnancement où la date $\tau^\star$ n'existe pas.

□

Nous allons nous intéresser ici aux marquages des places du graphe G_{p_1,p_2} lorsque les deux transitions sont en cours de franchissement. Pour cela, nous montrons qu'il est possible d'analyser les différents marquages des places au cours l'évolution du graphe en considérant une période de temps limitée.

On considère le vecteur ${}^\intercal N = (N_i, N_j)$ comme étant le plus petit T-semiflot de G_{p_1,p_2} *i.e.* N_i et N_j sont les plus petits entiers tels que $N_i \cdot w(p_1) = N_j \cdot v(p_1)$. D'autre part, d'après les hypothèses, le graphe G_{p_1,p_2} étant normalisé, nous avons $w(p_1) = Z_i$ et $v(p_1) = Z_j$. Nous en déduisons :

$$N_i \cdot Z_i = N_j \cdot Z_j \qquad \text{et aussi} \qquad \frac{l(t_i)}{l(t_j)} = \frac{Z_i}{Z_j} = \frac{N_j}{N_i}$$

On appelle $H = l(t_i) \cdot N_i = l(t_j) \cdot N_j$ l'hyper-période associée à l'ordonnancement au plus tôt des transitions de G_{p_1,p_2}. Nous en déduisons d'après le lemme 16 page ci-contre que pour toute date $\tau \geqslant \tau^\star$ et pour toute place $p \in \{p_1, p_2\}$, le marquage instantané $M(\tau, p)$ est défini par :

$$M(\tau, p) = M(\tau^\star, p) + \left\lfloor \frac{\tau - \tau^\star}{l(t_i)} \right\rfloor \cdot w(p) - \left\lceil \frac{\tau - \tau^\star}{l(t_j)} \right\rceil \cdot v(p)$$

En particulier, nous avons pour toute date $\tau \geqslant \tau^\star$:

$$\begin{aligned}
M(\tau + H, p) &= M(\tau^\star, p) + \left\lfloor \tfrac{\tau + H - \tau^\star}{l(t_i)} \right\rfloor \cdot w(p) - \left\lceil \tfrac{\tau + H - \tau^\star}{l(t_j)} \right\rceil \cdot v(p) \\
&= M(\tau^\star, p) + \left(\left\lfloor \tfrac{\tau - \tau^\star}{l(t_i)} \right\rfloor + N_i \right) \cdot w(p) - \left(\left\lceil \tfrac{\tau - \tau^\star}{l(t_j)} \right\rceil + N_j \right) \cdot v(p) \\
&= M(\tau^\star, p) + \left\lfloor \tfrac{\tau - \tau^\star}{l(t_i)} \right\rfloor \cdot w(p) - \left\lceil \tfrac{\tau - \tau^\star}{l(t_j)} \right\rceil \cdot v(p) \\
&= M(\tau, p)
\end{aligned}$$

Il est donc possible de se restreindre à l'étude des marquages durant une hyper-période H.

Nous allons à présent étudier les marquages des places de $G_{p1,p2}$ durant une hyper-période.

Durant l'intervalle $[\tau^\star, \tau^\star + H[$, il y a exactement N_i franchissements de t_i et N_j franchissements de t_j. On remarque également que :

- les transitions t_i et t_j sont franchies simultanément à la date $\tau^\star$,
- le T-semiflot ${}^\intercal N = (N_i, N_j)$ étant minimal, l'intervalle de temps $]\tau^\star, \tau^\star + H[$ contient exactement $N_i - 1$ initiations de la transition t_i et $N_j - 1$ initiations de la transition t_j s'effectuant toutes à des moments distincts (autrement N_i et N_j ne seraient pas minimum). Il y a donc au total $N_i + N_j - 2$ initiations de franchissements à des dates distinctes.

Nous allons alors subdiviser cette hyper-période en *classe de temps* de la manière suivante :

- On trie par ordre croissant les dates d'initiation des transitions que nous indiçons comme suit :
$$s_1 < s_2 < \ldots < s_{N_i+N_j-3} < s_{N_i+N_j-2}$$
- On crée alors $N_i + N_j - 1$ classes de temps notées C_i avec $i \in \{1, \ldots, N_i + N_j - 1\}$ telles que :
$$\begin{array}{rcl} C_1 & = &]\tau^\star, s_1[\\ C_2 & = &]s_1, s_2[\\ C_3 & = &]s_2, s_3[\\ & \vdots & \\ C_{N_i+N_j-2} & = &]s_{N_i+N_j-3}, s_{N_i+N_j-2}[\\ C_{N_i+N_j-1} & = &]s_{N_i+N_j-2}, \tau^\star + H[\end{array}$$

Le diagramme de Gantt (*voir* figure 7.5 page suivante) associé au circuit à deux transitions de la figure 7.4 page ci-contre permet de visualiser cette construction des classes de temps. Nous avons pour cet exemple $s_1 = \tau^\star + 3$, $s_2 = \tau^\star + 4$, $s_3 = \tau^\star + 6$, $s_4 = \tau^\star + 8$, $s_5 = \tau^\star + 9$ et $H = 12$.

Nous construisons alors 6 classes de temps comme suit :

$$\begin{array}{rcl} C_1 & = &]\tau^\star, s_1[\\ C_2 & = &]s_1, s_2[\\ C_3 & = &]s_2, s_3[\\ C_4 & = &]s_3, s_4[\\ C_5 & = &]s_4, s_5[\\ C_6 & = &]s_5, \tau^\star + 12[\end{array}$$

Nous remarquons que pour toute date $\tau \in C_i$ avec $i \in \{1, 2, \ldots, N_i + N_j - 1\}$, les transitions t_i et t_j sont toutes deux en cours de franchissement. De plus, le graphe étant

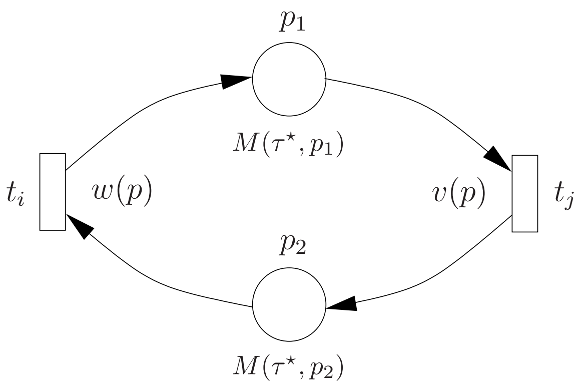


FIG. 7.4: Un circuit normalisé à deux transitions t_i et t_j tel que $v(p) = l(t_j) = 4$ et $w(p) = l(t_i) = 3$.

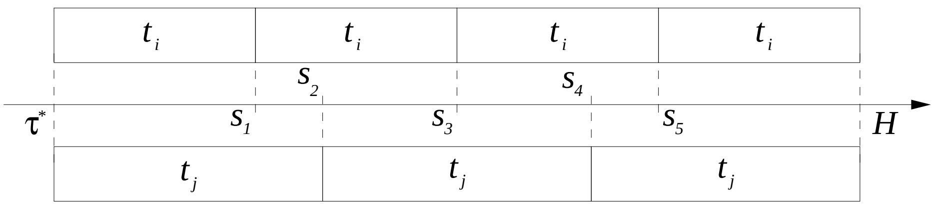


FIG. 7.5: Ordonnancement au plus tôt entre $\tau^\star$ et $\tau^\star{+}H$ associé au système de la figure 7.4.

un circuit à deux transitions normalisées, nous en déduisons :

$$M(\tau, p) + M(\tau, p') = M(0, p) + M(0, p') - w(p) - v(p)$$

En outre, le marquage des places du circuit reste constant sur toute la durée d'une classe de temps, soit :

$$\forall(\tau_1, \tau_2) \in C_i \times C_i \text{ on a } \left\{ \begin{array}{rcl} M(\tau_1, p) & = & M(\tau_2, p) \\ M(\tau_1, p') & = & M(\tau_2, p') \end{array} \right.$$

Le marquage du circuit durant une classe de temps est appelé par la suite état intermédiaire. Il y a alors au plus autant d'états intermédiaires distincts que de classes de temps, soit $N_i + N_j - 1$.

Nous considérons à présent le graphe d'événements généralisé non temporisé G'_{p_1,p_2} défini comme suit :

1. Les places et les transitions de G'_{p_1,p_2} sont les mêmes que celles de G_{p_1,p_2}.
2. Le marquage initial de G'_{p_1,p_2} est défini par $M_0(p) = M(\tau^\star, p) - v(p)$ pour toute place $p \in \{p_1, p_2\}$.

On montre à présent que la séquence des états intermédiaires d'une hyper-période est identique à la séquence des marquages générée par G'_{p_1,p_2}. En effet, nous pouvons observer

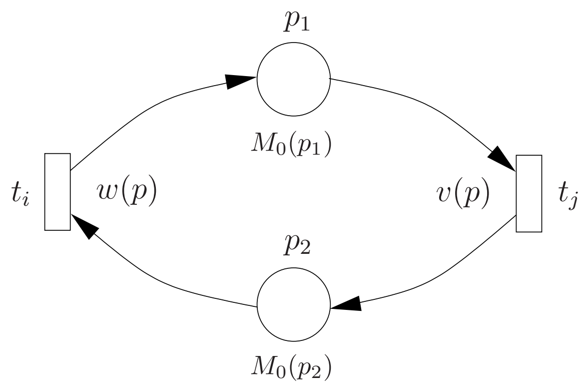


FIG. 7.6: Graphe d'événements généralisé G'_{p_1,p_2} associé à G_{p_1,p_2}.

que pour toute date $\tau_1 \in C_i$, $\forall i \in \{1, 2, \ldots, N_i + N_j - 2\}$ nous avons les marquages d'un état intermédiaire du graphe G_{p_1,p_2} qui vérifient :

$$\begin{cases} M(\tau_1, p) &= \alpha \geqslant 0 \\ M(\tau_1, p') &= \beta \geqslant 0 \\ \text{tels que} \quad \alpha + \beta &= M(0, p) + M(0, p') - w(p) - v(p) \end{cases}$$

De plus, si la date s_i correspond à une initiation de la transition t_i, nous avons alors :

$$\begin{cases} M(\tau_1, p) = \alpha \geqslant 0 \xrightarrow{t_i} M(\tau_2, p) = \alpha + w(p) \geqslant 0 \\ M(\tau_1, p') = \beta \geqslant 0 \xrightarrow{t_i} M(\tau_2, p') = \beta - w(p) \geqslant 0 \end{cases}$$

De même, si la date s_i correspond à un franchissement de la transition t_j, alors :

$$\begin{cases} M(\tau_1, p) = \alpha \geqslant 0 \xrightarrow{t_j} M(\tau_2, p) = \alpha - v(p) \geqslant 0 \\ M(\tau_1, p') = \beta \geqslant 0 \xrightarrow{t_j} M(\tau_2, p') = \beta + v(p) \geqslant 0 \end{cases}$$

De telles situations sont possibles si :

- Dans le premier cas, on a $\beta \geqslant w(p)$.
- Dans le second cas, on a $\alpha \geqslant v(p)$.

Ces contraintes sur les marquages intermédiaires peuvent être interprétées comme étant les règles sémantiques du graphe G'_{p_1,p_2}. Nous démontrons à présent de façon rigoureuse cette analogie.

Soit u_l la transition franchie à la date s_l, $\forall l \in \{1, \ldots, N_i + N_j - 2\}$. Nous notons $u = u_1 \cdots u_{N_i+N_j-2}$ la séquence de franchissements du graphe G_{p_1,p_2} pendant l'intervalle de temps $]\tau^\star, \tau^\star + H[$. Pour notre exemple, la séquence u est constituée de $u_1 = t_i$, $u_2 = t_j$, $u_3 = t_i$, $u_4 = t_j$, $u_5 = t_i$.

Le résultat suivant montre la relation forte entre les graphes G_{p_1,p_2} et G'_{p_1,p_2}.

Lemme 17. *$u = u_1 \cdots u_{N_i+N_j-2}$ est une séquence de franchissements valide pour G'_{p_1,p_2}.*

Démonstration. Pour tout $l \in \{1, \ldots, N_i + N_j - 2\}$, nous montrons par récurrence que le marquage des places de G'_{p_1,p_2} à l'issue du franchissement de la séquence $u = u_1 \cdots u_{l-1}$ vaut exactement $M(\tau, p)$ avec $p \in \{p_1, p_2\}$ et $\tau \in C_l$.

• *Base de la récurrence.* On démontre la propriété au rang $l = 1$. Il s'ensuit que la séquence $u = u_1 \cdots u_{l-1}$ est vide. Or, pour toute date $\tau \in C_1$, nous avons $M(\tau, p_1) = M(\tau^\star, p_1) - v(p_1)$ et $M(\tau, p_2) = M(\tau^\star, p_2) - v(p_2)$, soit par définition le marquage initial de G'_{p_1,p_2}. La base de la récurrence est donc vérifiée.

• *Hypothèse de récurrence.* Supposons que la propriété soit vraie pour tout $l \in \{1 \leqslant l \leqslant N_i + N_j - 1\}$.

• *Etape de la récurrence.* Considérons un couple de dates $(\tau_1, \tau_2) \in C_l \times C_{l+1}$. Nous avons alors deux cas à traiter :

(a) Si $u_l = t_i$, alors le marquage de G_{p_1,p_2} après le franchissement de la transition t_i est défini par $M(\tau_2, p_1) = M(\tau_1, p_1) + w(p_1)$ et $M(\tau_2, p_2) = M(\tau_1, p_2) - w(p_1)$. De plus, d'après l'hypothèse de récurrence les marquages des places p_1 et p_2 dans G'_{p_1,p_2} sont respectivement égaux à $M'(p_1) = M(\tau_1, p_1)$ et $M'(p_2) = M(\tau_1, p_2)$ à l'issue de la séquence de franchissements $u = u_1 \cdot u_l$. Nous en déduisons qu'après l'initiation de la transition t_i dans le graphe G'_{p_1,p_2}, les marquages des places p_1 et p_2 dans G'_{p_1,p_2} sont respectivement égaux à $M(\tau_2, p_1)$ et $M(\tau_2, p_2)$.

(b) Si $u_l = t_j$, alors le marquage de G_{p_1,p_2} après le franchissement de la transition t_j est défini par $M(\tau_2, p_1) = M(\tau_1, p_1) - w(p_2)$ et $M(\tau_2, p_2) = M(\tau_1, p_2) + w(p_2)$. En outre, d'après l'hypothèse de récurrence les marquages des places p_1 et p_2 dans G'_{p_1,p_2} valent respectivement $M'(p_1) = M(\tau_1, p_1)$ et $M'(p_2) = M(\tau_1, p_2)$ à l'issue de la séquence de franchissements $u = u_1 \cdot u_l$. Nous en déduisons qu'après l'initiation de la transition t_j dans le graphe G'_{p_1,p_2}, les marquages des places p_1 et p_2 dans G'_{p_1,p_2} sont respectivement égaux à $M(\tau_2, p_1)$ et $M(\tau_2, p_2)$.

Ceci complète notre raisonnement par récurrence, le lemme est alors démontré. □

Nous allons à présent considérer le nombre de couples de marquages distincts obtenus par le système G'_{p_1,p_2} à partir d'un certain marquage initial. Nous proposons d'abord un résultat important sur ces marquages :

Lemme 18. *Si $M'_0(p_1) + M'_0(p_2) < v(p_1) + w(p_2) - 2 \cdot pgcd_p$ alors tous les couples de marquages obtenus par le système G'_{p_1,p_2} sont distincts.*

Démonstration. D'après la condition nécessaire et suffisante de vivacité du théorème 8 page 64, le graphe d'événements généralisé G'_{p_1,p_2} est vivant si et seulement si $M'_0(p_1) + M'_0(p_2) > v(p_1) + w(p_2) - 2 \cdot pgcd_p$. Il en résulte que tous les couples de marquages atteints par G'_{p_1,p_2} sont distincts (autrement, le système serait vivant). □

Nous en déduisons le lemme suivant :

Lemme 19. *Si le marquage initial de G'_{p_1,p_2} satisfait à la contrainte $M'_0(p_1) + M'_0(p_2) < v(p_1) + w(p_2) - 2 \cdot pgcd_p$ alors le nombre de couples de marquages obtenus par G'_{p_1,p_2} vaut au plus $N_i + N_j - 2$.*

Démonstration. De par le lemme 3 page 50 sur la notion de jetons utiles, nous pouvons nous restreindre aux couples de marquages qui sont multiples de $pgcd_p$. De plus, d'après le lemme 7 page 63, le nombre total de jetons présents dans les places de G'_{p_1,p_2} demeure constant. Il en résulte que la somme des marquages des places de G'_{p_1,p_2} vaut initialement au plus $w(p_1) + v(p_1) - 3 \cdot pgcd_{p_1}$. On pose alors $S = w(p_1) + v(p_1) - 3 \cdot pgcd_{p_1}$.

Enumérons l'ensemble des couples de marquages $\begin{pmatrix} a \\ b \end{pmatrix}$ que l'on peut construire tout en vérifiant les conditions suivantes :

$$\begin{cases} a & \equiv & 0 \bmod (pgcd_p) \\ b & \equiv & 0 \bmod (pgcd_p) \\ a+b & = & S \end{cases}$$

Les couples (a, b) possibles sont énumérés comme suit :

$$\underbrace{\begin{array}{c|c|c|c|c|c} 0 & pgcd_p & 2 \cdot pgcd_p & \dots & S - pgcd_p & S \\ & & & & & \\ S & S - pgcd_p & S - 2 \cdot pgcd_p & \dots & pgcd_p & 0 \end{array}}_{\text{Couples } (a,b) \text{ tels que } a+b=S}$$

Soit au total $\frac{S}{pgcd_p}+1$ couples exactement. Par définition de N_i et N_j, nous avons $N_i \cdot w(p) = N_j \cdot v(p) = ppcm_p$. Nous avons également $w(p) \cdot v(p) = ppcm_p \cdot pgcd_p$ ce qui induit alors :

$$\frac{w(p)}{pgcd_p} = N_j \quad \text{et} \quad \frac{v(p)}{pgcd_p} = N_i$$

Nous avons alors :

$$\begin{array}{rcl} \frac{S}{pgcd_p} + 1 & = & \frac{w(p_1)+v(p_1)-3 \cdot pgcd_p}{pgcd_p} + 1 \\ & = & N_j + N_i - 2 \end{array}$$

En conséquence, le nombre de couples de marquages générés par G'_{p_1,p_2} vaut au plus $N_i + N_j - 2$. Ceci prouve le lemme. □

7.3.3 Optimalité de la solution

Les résultats précédents permettent alors de montrer que la solution proposée (*i.e.* $M(0, p) = M_{\min}(p),\ \forall p \in P$) s'avère être la plus petite solution possible pour le graphe G_{p_1,p_2}.

Théorème 28. *Soit G_{p_1,p_2} un circuit normalisé à deux transitions telles que :*

$$\forall t_i \in T,\ l(t_i) = \rho \cdot Z_i\ ,\ \rho > 0$$

Le marquage $M(0, p) = M_{\min}(p),\ \forall p \in \{p_1, p_2\}$ confère un débit maximum intrinsèque à G_{p_1,p_2}. De plus, la somme $M(0, p_1) + M(0, p_2)$ est minimum.

Démonstration. Le lemme 15 page 118 indique que la solution $M(0,p) = M_{\min}(p)$, $\forall p \in P$ permet au système G_{p_1,p_2} d'atteindre son débit maximum intrinsèque avec une politique d'ordonnancement au plus tôt.

Nous montrons alors que le marquage total du graphe correspond au marquage total minimum nécessaire au graphe G_{p_1,p_2} pour atteindre son débit maximum intrinsèque ρ. Nous faisons ici un raisonnement par contradiction. Supposons qu'il existe un marquage initial de ce système tel que $M(0,p_1) + M(0,p_2) < 2 \cdot M_{\min}(p)$ pour lequel le graphe G_{p_1,p_2} atteint son débit maximum intrinsèque ρ. D'après le lemme 16 page 120, nous pouvons considérer que le marquage initial est tel que l'ordonnancement au plus tôt est périodique sans période transitoire. D'après le lemme 17 page 124, le graphe d'événements généralisé non-temporisé G'_{p_1,p_2} de marquage initial $M'_0(p_1) = M(\tau^\star, p_1) - v(p_1)$ et $M\prime_0(p_2) = M(\tau^\star, p_2) - v(p_2)$ modélise les états intermédiaires de G_{p_1,p_2}. D'autre part comme nous avons supposé que $M(0,p_1) + M(0,p_2) < 2 \cdot M_{\min}(p)$, il s'ensuit que $M'_0(p_1) + M'_0(p_2) < v(p_1) + w(p_1) - 2 \cdot pgcd_{p_1}$. Or, d'après le lemme 19 page ci-contre, le nombre d'états intermédiaires de G_{p_1,p_2} est au plus de $N_i + N_j - 2$ avec ce marquage initial. Cependant, d'après les lemmes 17 page 124 et 18 page 125, le nombre d'états intermédiaires de G_{p_1,p_2} doit être égal à $N_i + N_j - 1$. D'où une contradiction.

De plus, comme nous avons $x_{p_1} = x_{p_2}$, il s'ensuit que la valeur de la solution est minimale également. □

Nous présentons ici une généralisation de ce résultat pour les GEGTS.

Corollaire 4. *Soit G un GEGTS unitaire tel que :*

$$\forall t_i \in T, l(t_i) = \rho \cdot Z_i, \ \ \rho > 0$$

En posant $M(0,p) = M_{\min}(p)$ pour toute place $p \in P$, nous obtenons un système de débit maximum intrinsèque et tel que $\sum\limits_{p \in P} x_p \cdot M(0,p)$ est minimum.

Démonstration. De nouveau, le lemme 15 page 118 indique que la solution $M(0,p) = M_{\min}(p)$, $\forall p \in P$ permet au système G_{p_1,p_2} d'atteindre son débit maximum intrinsèque avec une politique d'ordonnancement au plus tôt.

D'autre part, le graphe étant symétrique, le théorème précédent assure la minimalité de la solution. □

7.4 Généralisation pour une temporisation quelconque des transitions

Le résultat précédent permet de résoudre certaines instances bien déterminées de DÉBIT MAXIMUM INTRINSÈQUE - GÉNÉRALISÉ. On remarque également que l'ordonnancement périodique $S^\star$, dans ce cas précis, est critique dans le sens où si l'une des transitions ne respecte pas sa date de début alors toutes les dates de début des occurrences suivantes des transitions du GEGT seront décalées. L'aspect critique de cet ordonnancement étant étroitement lié à l'hypothèse sur les durées des transitions, on peut se demander ce qu'il advient du résultat lorsque cette hypothèse n'est plus vérifiée.

Le résultat suivant montre que le marquage défini pour le cas critique confère également à tout système un débit maximum intrinsèque. Ce résultat se révèle être la généralisation du corollaire 3 page 117.

Théorème 29. *Soit $G = (T, P, l)$ un GEGTS normalisé. Si $\forall p \in P, \quad M(0, p) = M_{\min}(p)$ alors l'ordonnancement au plus tôt de ce système est de débit maximum intrinsèque. De plus, la valeur de la fonction objectif pour cette solution est au pire deux fois plus grande que la valeur d'une solution optimale.*

Démonstration. Soit $i^\star \in \{1, \ldots, n\}$ tel que $\frac{Z_{i^\star}}{l(t_{i^\star})} = \min_{t_i \in T} \left\{ \frac{Z_i}{l(t_i)} \right\}$. On pose $\rho = \frac{Z_{i^\star}}{l(t_{i^\star})}$.

Nous montrons que la solution $M(0, p) = M_{\min}(p)$ pour toute place $p \in P$ est valide et que le débit de ce marquage initial $M(G)$ est alors maximum intrinsèque. Nous prouvons ensuite que cette solution donne à la fonction objectif du problème une valeur qui est au pire deux fois plus grande que celle de la valeur optimale.

- On considère l'ordonnancement périodique suivant :

$$S_1 = \left\{ s(t, k) = (k - 1) \cdot \frac{Z_t}{\rho}, \ \forall (t, k) \in T \times \mathbb{N}^\star \right\}$$

  Le débit de cet ordonnancement vaut alors ρ. Afin de démontrer la validité de cet ordonnancement, nous associons à chaque transition t_j le nombre réel $l'(t_j) = \frac{Z_j}{\rho}$. Par définition $\rho = \min_{t_i \in T} \left\{ \frac{Z_i}{l(t_i)} \right\}$, ce qui entraîne alors $l'(t_j) \geqslant l(t_j)$ pour chaque transition $t_j \in T$. D'après le lemme 15 page 118, l'ordonnancement S est valable pour le marquage initial proposé et cette nouvelle temporisation. Il en résulte que S_1 est également un ordonnancement valide pour le système initial et qu'il est de débit maximum intrinsèque.

- D'après la condition nécessaire et suffisante de vivacité du théorème 8 page 64, tout marquage vivant doit alors vérifier pour tout couple de places retours (p, p') la condition suivante :

$$M(0, p) + M(0, p') \geqslant M_{\min}(p)$$

  Nous en déduisons que $\frac{1}{2} \sum_{p \in P} x_p \cdot M_{\min}(p)$ est une borne inférieure de la valeur de la solution optimale. D'où le ratio d'approximation.

$\square$

Conclusion

Nous avons proposé dans ce chapitre un algorithme polynomial permettant de résoudre efficacement le problème Débit maximum intrinsèque - Généralisé aussi bien dans sa version non-généralisée que généralisée. Pour cela, nous avons étudié un ensemble d'instances particulières pour lesquelles nous avons élaboré une méthode menant à une résolution optimale du problème. De là, nous en avons déduit un algorithme polynomial 2-approché pour toutes les instances de Débit maximum intrinsèque - Généralisé.

Ces résultats sont importants car ils fournissent un outil performant pour la résolution d'un problème bi-critère qui vraisemblablement n'appartient pas à la classe NP. En outre, la solution principale proposée (*i.e.* le Théorème 29 page précédente) dans ce chapitre permet de définir une borne intéressante pour la résolution du problème Optimisation de marquage. Celle-ci pourra constituer une base intéressante pour une méthode de type *Branch and Bound* pour une résolution exacte de ces problèmes d'optimisation bi-critère.

Chapitre 8

Conclusion

Le problème du dimensionnement des mémoires pour systèmes embarqués est un problème crucial dans le contexte industriel et économique actuel. Toutes les méthodes algorithmiques développées jusqu'ici pour ce problème sont limitées par l'explosion combinatoire liée à leur complexité et aux modèles utilisés.

La première partie de cette thèse est consacrée à l'étude de la vivacité des graphes d'événements généralisés. Pour vérifier la vivacité d'un graphe, les méthodes développées jusqu'à présent reposent sur des techniques algorithmiques de complexité pseudo-polynomiale (*cf.* technique de l'expansion de Munier [Mun93] ou la technique basée sur le plus petit T-semiflot de Teruel *et al.* [TCWCS92]). Ces méthodes s'avèrent donc inappropriées pour des instances de grande taille. Nous avons mis au point une transformation appelée normalisation qui permet de simplifier l'analyse de la vivacité. Nous avons alors remarqué que le nombre de jetons demeurait constant sur tout circuit d'un graphe d'événements généralisé normalisé. Cette observation nous a conduit à proposer une condition suffisante de vivacité portant sur le marquage total des places d'un circuit. Nous avons en outre montré que cette condition suffisante était également nécessaire dans le cas des circuits à deux transitions. Nous avons alors proposé un algorithme polynomial permettant de vérifier cette condition suffisante pour tout graphe d'événements généralisé.

Nous avons défini et résolu de façon satisfaisante au chapitre 5 le problème de marquage. Notre résolution du problème a montré qu'en dépit de la non-appartenance apparente à la classe NP du problème de la vivacité, il était possible de s'attaquer à des problèmes d'optimisation sur ce modèle. L'étude du problème de dimensionnement des mémoires a mis en exergue l'importance du problème de la vivacité. En effet, nous avons montré que l'expression de la contrainte de capacité sur les mémoires pouvait être modélisée par un couple de places non bornées avec un marquage initial particulier. Cette modélisation de la contrainte de capacité permet alors de nous focaliser sur le problème de la vivacité du système résultant. La condition suffisante de vivacité et la notion de marquages utiles que nous avons définis au chapitre 4 ont permis de résoudre efficacement le problème de marquage.

La normalisation et la condition suffisante de vivacité constituent des avancées importantes pour la compréhension du problème. La normalisation permet de donner une description mathématique plus simple de l'ensemble des problèmes associés au modèle des graphes d'événements généralisés. L'algorithme polynomial associé à la condition suffisante de vivacité permet également de traiter des instances plus importantes. Cependant, la complexité théorique de la vivacité reste un problème ouvert et difficile. L'étude de la vivacité par des méthodes approchées efficaces constitue un défi important et permettrait d'obtenir des outils plus adaptés que les techniques existantes.

La vérification du débit de l'application constitue également un problème important qui survient dans la phase de dimensionnement des mémoires pour systèmes embarqués. Contrairement au cas non-généralisé, il n'existe aucune méthode algorithmique polynomiale pour le calcul des débit des transitions d'un graphe d'événements généralisé temporisé. Les méthodes de calcul pour ce problème sont basées sur l'expansion du graphe d'événements généralisé définie par Munier [Mun93] et leur usage est donc limité par la taille des instances à traiter. Nous avons cependant abordé quatre problèmes d'optimisation bi-critère. Ces quatre problèmes proposent d'optimiser le débit d'une application étant donnée une taille totale de mémoire bornée. Il existe très peu de travaux pour prendre en compte l'optimisation de ces deux critères [SGB06]. Ces problèmes d'optimisation bi-critère sont difficiles à appréhender en raison de la non-appartenance supposée à la classe NP des principaux problèmes de décision sous-jacents.

Nous avons prouvé au chapitre 6 la NP-complétude de quatre problèmes d'ordonnancement cyclique bi-critère. Nous avons démontré entre autres la NP-complétude des problèmes Optimisation de marquage et Débit maximum intrinsèque qui était inconnue depuis plus de vingt ans. La réduction initiale que nous avons établie pour le problème Débit Max - Surface Min a permis également de mettre en rapport ces problèmes d'ordonnancement cyclique bi-critère au problème de la K-coloration *via* le lemme de Minty. Nous espérons que cette analogie entre les deux problèmes sera féconde pour la mise au point d'heuristiques nouvelles pour ces problèmes d'ordonnancement cyclique.

Nous avons également défini un algorithme polynomial 2-approché pour le problème Débit maximum intrinsèque dans sa version non-généralisée. Nous avons montré que ce résultat pouvait être étendu pour les graphes d'événements généralisés. De plus, nous avons identifié une partie de la structure des instances pour lesquels la solution proposée par notre algorithme est optimale. Cette caractéristique laisse espérer des améliorations du ratio d'approximation pour ce problème. L'étude de ce problème est importante car elle permet de garantir qu'une application donnée atteindra bien un certain débit. La minimisation de la somme totale des marquages initiaux correspond alors à la minimisation de la taille des mémoires du système embarqué. En outre, l'analyse de la structure des ordonnancements associés aux marquages construits par notre algorithme pourrait permettre de réaliser des méthodes d'évaluation approximatives efficaces du débit d'un graphe d'événements généralisé.

Une extension naturelle de cette thèse consisterait à réaliser des méthodes heuristiques efficaces pour la vérification de la vivacité d'un graphe d'événements généralisé. Une caractérisation théorique du problème de la vivacité pourrait servir de base pour la réalisation de ce type d'heuristiques. Il serait également intéressant d'étudier le calcul du débit des graphes d'événements généralisés. Une méthode d'évaluation approximative efficace s'avérerait particulièrement utile pour vérifier rapidement les performances spécifiées des systèmes embarqués. Enfin, une étude approfondie de la relation entre les problèmes d'ordonnancement cyclique bi-critère et le problème de la K-coloration permettrait de réaliser des heuristiques originales pour la résolution de ces problèmes dans le cas d'instances non-généralisées.

Bibliographie

[Adé96] M. Adé. *Data memory minimization for synchronous data flow graphs emulated on DSP-FPGA targets.* PhD thesis, Université Catholique de Louvain, 1996.

[ALP94] M. Adé, R. Lauwereins, et J. A. Peperstraete. Buffer memory requirements in dsp applications. In *IEEE 5th International Workshop on Rapid System Prototyping*, pages 108–123, Grenoble, France, 1994.

[ALP95] M. Adé, R. Lauwereins, et J. A. Peperstraete. Hardware-software codesign with grape. In *RSP '95 : Proceedings of the Sixth IEEE International Workshop on Rapid System Prototyping (RSP'95)*, page 40, Washington, DC, USA, 1995. IEEE Computer Society.

[ALP96] M. Adé, R. Lauwereins, et J. A. Peperstrate. Implementing dsp applications on heterogeneous targets using minimal size data buffers. In *RSP '96 : Proceedings of the 7th IEEE International Workshop on Rapid System Prototyping (RSP '96)*, page 166, Washington, DC, USA, 1996. IEEE Computer Society.

[BBC92] D. Beauquier, J. Berstel, et P. Chrétienne. *Eléments d'algorithmique.* Masson, 1992.

[BBHL95] S. Bhattacharyya, J. Buck, S. Ha, et E. Lee. A compiler scheduling framework for minimizing memory requirements of multirate dsp systems represented as dataflow graphs. *IEEE Transactions on circuits and systems*, 42(3), 1995.

[Bha94] S. S. Bhattacharyya. *Compiling dataflow programs for digital signal processing.* PhD thesis, Université de Californie, Berkeley, 1994.

[BL94] S. S. Bhattacharyya et E. A. Lee. Looped schedules for dataflow descriptions of multirate signal processing algorithms. *Formal Methods in System Design*, 5(3) :183–205, 1994.

[BML97] S. S. Bhattacharyya, P. K. Murthy, et E. A. Lee. Optimized software synthesis for synchronous dataflow. In *ASAP '97 : Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors*, page 250, Washington, DC, USA, 1997. IEEE Computer Society.

[BML99] S. S. Bhattacharyya, P. K. Murthy, et E. A. Lee. Synthesis of embedded software from synchronous dataflow specifications. *Journal of VLSI Signal Processing*, 21(2) :151–166, 1999.

[Buc93] J. T. Buck. *Scheduling Dynamic Dataflow Graphs with Bounded Memory Using the Token Flow Model.* PhD thesis, Université de Californie, Berkeley, 1993.

[CC88] J. Carlier et P. Chrétienne. *Problèmes d'ordonnancement : modélisation, complexité, algorithmes.* Masson, 1988.

[CDQV85] G. Cohen, D. Dubois, J. P. Quadrat, et M. Viot. A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing. In *IEEE Transactions on automatic control*, volume AC-30, pages 210–220, Mar 1985.

[CHEP71] F. Commoner, A. W. Holt, S. Even, et A. Pnueli. Marked directed graphs. *J. Comput. Syst. Sci.*, 5(5) :511–523, 1971.

[Chr83] P. Chrétienne. *Les réseaux de Petri temporisés.* PhD thesis, Thèses d'état, Université P. et M. Curie, 1983.

[CLR90] T. Cormen, C. Leiserson, et R. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *STOC '71 : Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

[ČP93] M. Čubrić et P. Panangaden. Minimal memory schedules for dataflow networks. In *CONCUR '93, 4th International Conference on Concurrency Theory, Lecture Notes in Computer Science*, volume 715, pages 368–383, 1993.

[CWR93] P. Chrzastowski-Wachtel et M. Raczunas. Liveness of weighted circuits and the diophantine problem of frobenius. In *Proceeding of FCT'93, Lecture Notes in Computer Science*, volume 710, pages 171–180. Springer, 1993.

[CZW93] D. T. Chao, M. Zhou, et D. T. Wang. Multiple-weighted marked graph. In *IFAC - 12th Triennial World Congress*, pages 371–374, Sidney, Australia, 1993.

[Den74] J. B. Dennis. First version of a data flow procedure language. In *Programming Symposium, Proceedings Colloque sur la Programmation*, pages 362–376. Springer-Verlag, 1974.

[DFMS97] A. Di Febbraro, R. Minciardi, et S. Sacone. Deterministic timed event graphs for performance optimization of cyclic manufacturing processes. *IEEE Transactions on robotics and automation*, 13(2), 1997.

[DGN92] V. Van Dongen, G. R. Gao, et Q. Ning. A polynomial time method for optimal software pipelining. In *CONPAR '92/ VAPP V : Proceedings of the Second Joint International Conference on Vector and Parallel Processing*, pages 613–624, London, UK, 1992. Springer-Verlag.

[DIG99] A. Dasdan, S. S. Irani, et R. K. Gupta. Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems. In *DAC '99 : Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 37–42, New York, NY, USA, 1999. ACM Press.

[DK82] A. L. Davis et R. M. Keller. Data flow program graphs. *IEEE Computer*, 15(2) :26–41, 1982.

[EBLP95] M. Engels, G. Bilsen, R. Lauwereins, et J. A. Peperstraete. Cyclo-static dataflow : Model and implementation. In *In Proc. 28th Asilomar Conf. on Signals, Systems, and Computers*, pages 503–507, Washington, DC, USA, 1995. IEEE Computer Society.

[Gau90] S. Gaubert. An algebraic method for optimizing resources in timed event graphs. In *Proceedings of the 9th conference on Analysis and Optimization of Systems, Lecture Notes in Computer Science*, volume 144, pages 957–966. Springer, 1990.

[GBS05] M. Geilen, T. Basten, et S. Stuijk. Minimising buffer requirements of synchronous dataflow graphs with model checking. In *DAC '05 : Proceedings of the 42nd annual conference on Design automation*, pages 819–824, New York, NY, USA, 2005. ACM Press.

[GG93] R. Govindarajan et G. R. Gao. A novel framework for multi-rate scheduling in dsp applications. In *Proc. Inter. Conf. on Application-Specific Array Processors*, pages 77–88. IEEE Press, 1993.

[GG95] R. Govindarajan et G.R. Gao. Rate-optimal shcedule for multi-rate dsp computations. *Journal of VLSI Signal Processing*, (9) :211–235, 1995.

[GGB+06] A. H. Ghamarian, M. C. W. Geilen, T. Basten, B. D. Theelen, M. R. Mousavi, et S. Stuijk. Liveness and boundedness of synchronous data flow graphs. Technical report, 2006. disponible à http ://www.es.ele.tue.nl/esreports/esr-2006-04.pdf.

[GGD94] R. Govindarajan, G. Gao, et P. Desai. Minimizing memory requirements in rate-optimal schedules. In *Proc. of the Inter. Conf. on Application Specific Array Processors, San Francisco*, 1994.

[GGD02] R. Govindarajan, G. Gao, et P. Desai. Minimizing memory requirements in rate-optimal schedule in regular dataflow networks. *Journal of VLSI Signal Processing*, 31(3), 2002.

[GGS+06] A. H. Ghamarian, M. C. W. Geilen, S. Stuijk, T. Basten, A. J. M. Moonen, M. J. G. Bekooij, B. D. Theelen, et M. R. Mousavi. Throughput analysis of synchronous data flow graphs. In *Sixth International Conference on Application of Concurrency to System Design ACSD*, pages 25–36, Washington, DC, USA, 2006. IEEE Computer Society.

[GJ79] M. R. Garey et D. S. Johnson. *Computers and Intractability : : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[GPS00] A. Giua, A. Piccaluga, et C. Seatzu. Incremental optimization of timed cyclic event graphs. In *Proc. 1999 IEEE Int. Conf. on Robotics and Automation (San Francisco, California)*, pages 2211–2216, April 2000. InternalNote : Submitted by : hr.

[GPS02] A. Giua, A. Piccaluga, et C. Seatzu. Firing rate optimization of cyclic timed event graphs by token allocations. *Automatica*, 38(1) :91–103, January 2002.

[HM94] C. Hanen et A. Munier. *Cyclic scheduling on parallel processors : an overview, dans Scheduling Theory and its Applications, P. Chrétienne, E. G. Coffman, J. K. Lenstra, and Z. Liu.* John Wiley & Sons Ltd., 1994.

[HP89] H. Hillion et J. M. Proth. Performance evaluation of a job-shop system using timed event graph. *IEEE transactions on automatic control*, 34(1) :3–9, 1989.

[IP95] K. Ito et K. K. Parhi. Determining the minimum iteration period of an algorithm. *J. VLSI Signal Processing*, 11(3) :229–44, 1995.

[Kah74] G. Kahn. The semantics of a simple language for parallel programming. In *Proceedings of the IFIP Congress 74*, North Holland, Amsterdam, 1974.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller et J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Kar78] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23 :309 – 311, 1978.

[KM66] R. M. Karp et R. E. Miller. Properties of a model for parallel computations : Determinacy, termination, queueing. *SIAM*, 14(63) :1390 – 1411, 1966.

[LEAP95] R. Lauwereins, M. Engels, M. Adé, et J. A. Peperstraete. Grape-II : A system-level prototyping environment for dsp applications. *Computer*, 28(2) :35–43, 1995.

[Lie76] Y. E. Lien. Termination properties of generalized petri nets. *SIAM J. Comput.*, 5(2) :251–265, 1976.

[LK98] E. Levner et V. Kats. A parametric critical path problem and an application for cyclic scheduling. *Discrete Applied Mathematics*, 87 :149–158, 1998.

[LM87a] E.A. Lee et D.G. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Transaction on Computers*, C-36(1) :24–35, 1987.

[LM87b] E.A. Lee et D.G. Messerschmitt. Synchronous data flow. *IEEE Proceedings of the IEEE*, 75(9), 1987.

[LPX92] S. Laftit, J.M. Proth, et X. Xie. Optimization of invariant criteria for event graphs. *IEEE Transactions on Automatic Control*, 37(5) :547–555, 1992.

[MB99] P. K. Murthy et S. S. Bhattacharyya. A buffer merging technique for reducing memory requirements of synchronous dataflow specifications. In *ISSS '99 : Proceedings of the 12th international symposium on System synthesis*, page 78, Washington, DC, USA, 1999. IEEE Computer Society.

[MBL97] P. K. Murthy, S. S. Bhattacharyya, et E. A. Lee. Joint minimization of code and data for synchonous dataflow programs. *Journal of Formal Methods in System Design*, 11(1), 1997.

[Min62] G. J. Minty. A theorem on n-coloring the points of a linear graph. *The American Mathematical Monthly*, 69(7) :623–624, 1962.

[Mun91] A. Munier. *Contribution à l'étude des ordonnancements cycliques.* PhD thesis, Université P. et M. Curie, 1991.

[Mun93] A. Munier. Régime asymptotique optimal d'un graphe d'événements temporisé : application à un problème d'assemblage. *RAIRO*, Vol. 27(5) :171–180, 1993.

[Mur96] P. K. Murthy. *Scheduling Techniques for Synchronous and Multidimensional Synchronous Dataflow.* PhD thesis, Université de Californie, Berkeley, 1996.

[NGL99] Walid A. Najjar, Guang R. Gao, et Edward A. Lee. Advances in the dataflow computationnal model. *Parallel Computing*, (25), 1999.

[Par89] K. K. Parhi. Algorithm transformation techniques for concurrent processors. In *Proceedings of the IEEE*, volume 77, pages 1879–1895, Dec. 1989.

[Pet62] C. A. Petri. *Kommunikation mit Automaten. Ph.D. dissertation.* PhD thesis, Institut fur Instrumentelle Mathematik, Bonn, 1962.

[PPL95] T. M. Parks, J. L. Pino, et E. A. Lee. A comparison of synchronous and cylco-static dataflow. Technical report, Nov 1995. disponible à http ://ptolemy.eecs.berkeley.edu/publications/papers/95/csdfVSsdf/.

[PSX97] J. M. Proth, N. Sauer, et X. L. Xie. Optimization of the number of transportation devices in a flexible manufacturing system using event graphs. *IEEE transactions on industrial electronics*, 44(3) :298–306, 1997.

[PX95] J. M. Proth et X. Xie. *Les réseaux de Petri pour la conception et la gestion des systèmes de production.* Masson, 1995.

[Rei68] R. Reiter. Scheduling parallel computations. *Journal of the Association for Computing Machinery*, 15(4) :590–599, 1968.

[RH80] C. V. Ramamoorthy et G. S. Ho. Performance evaluation of asynchronous concurrent systems using petri nets. *IEEE Transactions on software engineering*, (5), 1980.

[Sau03] N. Sauer. Marking optimization of weighted marked graphs. *Discrete Event Dynamic Systems*, 13(3) :245–262, 2003.

[SGB06] S. Stuijk, M. Geilen, et T. Basten. Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs. In *DAC '06 : Proceedings of the 43rd annual conference on Design automation*, pages 899–904, New York, NY, USA, 2006. ACM Press.

[TCWCS92] E. Teruel, P. Chrzastowski-Wachtel, J. M. Colom, et M. Silva. On weighted T-systems. In *Proocedings of the 13th Internationnal Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Science*, volume 616. Springer, 1992.

[TS04] L. Toursi et N. Sauer. Branch and bound approach for marking optimization problem of weighted marked graphs. In *In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 1777–1782, Washington, DC, USA, 2004. IEEE Computer Society.

[WELP95] P. Wauters, M. Engels, R. Lauwereins, et J. A. Peperstraete. Cyclo-static data flow. In *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3255–3528, Washington, DC, USA, 1995. IEEE Computer Society.

[WELP96] P. Wauters, M. Engels, R. Lauwereins, et J. A. Peperstraete. Cyclo-dynamic dataflow. In *PDP '96 : Proceedings of the 4th Euromicro Workshop on Parallel and Distributed Processing (PDP '96)*, page 319, Washington, DC, USA, 1996. IEEE Computer Society.

Publications personnelles

Conférence nationale avec comité de lecture et actes

Complexity results for bi-criteria cyclic scheduling problem.
Olivier Marchetti, Alix Munier-Kordon.
Conférence RenPar 17 à Perpignan (4-6 Oct 2006).

Conférence internationale avec comité de lecture et actes

A polynomial algorithm for a bi-criteria cyclic scheduling problem.
Olivier Marchetti, Alix Munier-Kordon.
The 25th Workshop of the UK PLANNING AND SCHEDULING Special Interest Group à Nottingham (14-15 Decembre 2006).

Article soumis à des revues internationales avec comité de lecture

A sufficient condition for the liveness of weighted event graphs.
Olivier Marchetti, Alix Munier-Kordon.
Soumis au journal EJOR en Octobre 2004. Actuellement en révision.

Minimizing Place Storage Capacities of a Weighet Event Graph.
Olivier Marchetti, Alix Munier-Kordon.
Soumis au journal Discrete Event Dynamic Systems en Mai 2005. Actuellement en révision.

Complexity results for bi-criteria cyclic scheduling problem.
Olivier Marchetti, Alix Munier-Kordon.
Soumis à Discrete Optimization fin septembre 2006.