Ordonnancement dans les ateliers hybrides en environnement incertain

Ghassen Cherif

THESE

Pour obtenir le diplôme de doctorat

Spécialité Automatique, Signal, Productique, Robotique

Préparée au sein de « Université du Havre Normandie »

Ordonnancement dans les ateliers hybrides en environnement incertain

Présentée et soutenue par

Ghassen CHERIF

Thèse soutenue publiquement le 25/03/2021 devant le jury composé de		
M. Ahmed NAIT-SIDI-MOH	Maitre de Conférences-HDR / LTI / Université de Picardie Jules Verne	Rapporteur
M. Moamar SAYED-MOUCHAWEH	Professeur des Universités / Institut Mines-Télécom Lille Douai	Rapporteur
Mme Isabel DEMONGODIN	Professeur des Universités / LSIS / Université de Marseille	Examinatrice
Mme Pascale MARANGE	Maitre de Conférences / CRAN / Université de Lorraine	Examinatrice
M. Édouard LECLERCQ	Maitre de Conférences / GREAH / Université le Havre Normandie	Encadrant de thèse
M. Dimitri LEFEBVRE	Professeur des Universités / GREAH / Université le Havre Normandie	Directeur de thèse

Thèse dirigée par Dimitri LEFEBVRE, laboratoire GREAH (EA 3220)

Dédicaces

A celle qui m'a élevé avec amour et tendresse
A celle qui a toujours cru en moi
A ma maman chérie.

A celui qui m'a toujours guidé vers la bonne voie
A celui qui s'est sacrifié pour ma réussite
A mon cher papa.

A celle qui m'a toujours soutenu avec tant d'amour et de tendresse.
A ma Femme chérie Ines

A mon frère Faiz et ma sœur Ghada qui m'ont toujours soutenu.

Aucune dédicace ne saurait exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être.

A toute ma famille et Belle Famille.
A tous mes amis.
A ceux que j'aime et qui m'aiment...

Remerciements

Cette thèse de doctorat était une aventure unique et passionnante. Une expérience enrichissante qui m'a beaucoup apporté sur tous les plans. Elle n'aurait cependant pas été vécue ainsi sans les nombreuses personnes que j'ai eu l'honneur et le plaisir de côtoyer et qui m'ont aidé, de près ou de loin, à atteindre mes objectifs.

Ainsi, je tiens à exprimer ma forte reconnaissance à mon directeur de thèse : M. Dimitri LEFEBVRE ainsi qu'à mon encadrant M. Édouard LECLERCQ, d'abord pour la confiance qu'ils m'ont accordée en m'acceptant dans cette aventure, ensuite pour leur disponibilité de tous les instants, leur remarquable générosité et leur précieux conseils prodigués durant toutes ces années de thèse. Au-delà de mon encadrement qu'ils ont mené avec grande énergie, ils ont été un soutien permanent sur le plan personnel et professionnel. Je veux avec ces quelques mots leur exprimer toute ma gratitude. Ce travail est aussi le vôtre!

Merci à Mme Pascale MARANGE et à M. Eric SANLAVILLE pour leur participation à mon comité de suivi de thèse et leurs suggestions toujours avisées.

Je tiens aussi à remercier M. Ahmed NAIT-SIDI-MOH et M. Moamar SAYED-MOUCHAWEH pour avoir accepté d'être les rapporteurs de ma thèse et pour le temps qu'ils ont consacré, malgré leurs charges et leurs responsabilités, à l'évaluation de mes travaux avec soin. Je remercie également Mme Isabel DEMONGODIN et Mme Pascale MARANGE pour avoir accepté d'examiner mon travail. Merci pour le temps et l'effort que vous m'avez consacrés et de m'avoir fait l'honneur de faire partie de mon Jury de thèse.

Cette thèse s'est déroulée au sein du Groupe de Recherche en Electrotechnique et Automatique du Havre (GREAH). Je remercie le directeur du GREAH : M. Georges BARAKAT de m'avoir accueilli au sein de son laboratoire. Je tiens à remercier tous mes amis, collègues, enseignants et personnels du laboratoire. Mes remerciements s'adressent également aux membres du département Génie Mécanique et Productique (GMP) de l'IUT du Havre qui m'ont chaleureusement accueilli au sein de l'équipe enseignante.

Je ne peux terminer sans adresser mes plus profonds et sincères remerciements aux êtres qui me sont le plus chers. Je pense bien évidement à mes très chers parents à qui je dois tout, à mon frère et ma sœur qui m'ont toujours soutenu. Je n'aurais jamais pu aller aussi loin sans vous. Je vous remercie infiniment et j'espère vous rendre fiers. A ma tendre femme qui m'a toujours soutenu. Ton amour,

ton regard et ton sourire donnent un sens à ma vie, cet aboutissement est aussi le tien. Je dédie également ce mémoire à mes grands-parents, à ma belle-famille, tous mes cousins et cousines, tantes et oncles ainsi qu'à tous mes amis.

Table des matières

Liste des figures

Liste des Tableaux

Abréviations

FMS	Système manufacturier flexible
SED	Système à événements discrets
PN	Réseau de Petri
TPN	Réseau de Petri temporisé
T-TPN	Réseau de Petri temporisé sur les transitions
P-TPN	Réseau de Petri temporisé sur les places
IGS	Recherche informée de graphes
S3PR	Systèmes de processus séquentiels simples avec ressources
S4R	Systèmes de systèmes séquentiels avec ressources partagées
S2OPR	Ensemble de processus "open" simples avec des ressources
BS	Recherche en faisceau
FBS	Recherche filtrée en faisceau
FBS-D	Recherche filtrée en faisceau sur la base de durée
FBS-DR	Recherche filtrée en faisceau sur la base de durée et du risque
HFBS	Recherche filtrée en faisceau hybride
IHFBS	Recherche filtrée itérée en faisceau hybride
BAS	Recherche en faisceau A*
GFBS	Recherche filtrée en faisceau par génération
GFBS-D	Recherche filtrée en faisceau par génération sur la base de durée
GFBS-DR	Recherche filtrée en faisceau par génération sur la base de durée et du risque
GDFBS	Recherche double filtrée en faisceau par génération
HPSO	Optimisation hybride par essaim particulaire
GREAH	Groupe de Recherche en Electrotechnique et Automatique du Havre

Chapitre 1 : Introduction générale

1.1 Contexte général

Les systèmes manufacturiers flexibles (FMS) se composent d'un ensemble fini d'opérations et de ressources et peuvent traiter différents types de produits en fonction d'une gamme opératoire. Dans un FMS, il existe une certaine flexibilité qui le rend capable de répondre à des conditions de production variées. Actuellement, le problème d'ordonnancement est un des thèmes les plus étudiés de la recherche opérationnelle. Un ordonnancement consiste à organiser des tâches dans le temps, tout en respectant un ensemble de contraintes temporelles ou liées à la disponibilité des ressources, tout en optimisant un ou plusieurs critères donnés [Rodammer & Preston White, 1999]. En effet, les problèmes d'ordonnancement occupent une place importante dans la littérature consacrée à l'optimisation combinatoire. Les problèmes d'ordonnancement sont présents dans tous les secteurs d'activité de l'économie tels que l'administration [Carlier et Chrétienne, 1988], l'informatique [Blazewicz et al., 1996], la construction [Carlier et Chrétienne, 1988], ou encore l'industrie manufacturière [Pinedo, 1955].

Le but de ce travail est de résoudre des problèmes d'ordonnancement pour une certaine classe d'ateliers flexibles présents dans l'industrie 4.0. De nos jours, la production avec de grandes séries de produits identiques n'est plus compétitive. Les consommateurs demandent des produits personnalisés. Par conséquent, le grand défi de l'Industrie 4.0 est de proposer ce type de produits, et malgré les faibles volumes de fabrication, de maintenir les gains. Les consommateurs réclament aussi davantage d'information durant les phases de production. Le concept d'industrie 4.0 ou d'industrie du futur correspond à ces nouvelles organisations des moyens de production appelée «production intelligente». La production intelligente nécessite, par exemple, de donner de la flexibilité à certaines opérations (et de maintenir des contraintes de précédence à d'autres) comme le montre l'exemple de la figure 1 qui représente un atelier avec deux gammes opératoires comportant chacune quatre moyens de production notés M_1, M_2, M_3, et M_4. Dans la première gamme opératoire, la production pourrait démarrer par M_1 puis M_2 ou par M_2 puis M_1. La production intelligente nécessite également d'utiliser les mêmes machines ou serveurs pour réaliser différentes opérations. Sur la figure 1, M_1 est utilisée pour une opération d'une durée de 60s dans la première gamme opératoire et pour une autre opération d'une durée de 100s dans la deuxième gamme opératoire.

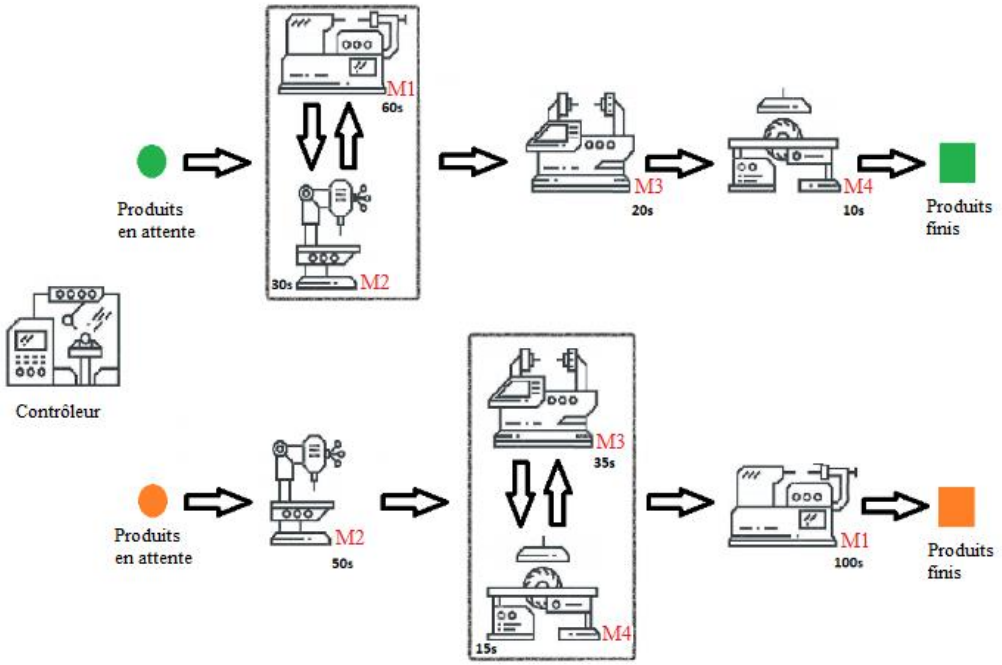


Figure 1 : Exemple d'une organisation d'un FMS de l'industrie 4.0

Dans ce contexte, une classe particulière de FMS appelée « ateliers hybrides » qui combine des opérations avec des contraintes de précédence totale et des opérations avec une flexibilité totale est proposée dans ce travail.

Afin de résoudre un problème d'ordonnancement, une description du système est nécessaire. En effet, la modélisation d'un problème d'ordonnancement se traduit en respectant les contraintes de disponibilité, les caractéristiques des opérations, les types des ressources et surtout l'organisation et la configuration de l'atelier [Yalaoui, 2006]. Celle-ci peut être modélisée de différentes manières selon le type et la complexité du problème.

Un problème d'ordonnancement peut être formulé mathématiquement [Koné et al., 2013 ; Koné et al., 2011] en représentant les différentes contraintes liées à ce problème sous forme d'équations ou d'inéquations. Cette modélisation mathématique a deux limites principales. D'un côté, à chaque modification des contraintes ou des spécifications du problème, il est nécessaire de réviser toute la formulation mathématique [Graham et al., 1979]. Cette limitation devra être surmontée car les FMS sont des systèmes modulaires et souvent flexibles. D'un autre côté, formaliser mathématiquement un problème d'ordonnancement ne permet souvent que de résoudre des problèmes de petite taille et ne permet pas de s'attaquer à des problèmes avec un espace de recherche de grande taille [Brucker, 2007].

L'essentiel pour résoudre un problème d'ordonnancement est de trouver un modèle qui décrit bien le problème et qui soit suffisamment simple pour être manipulé. Dans ce contexte, on s'intéressera, au cours de cette thèse, à la résolution des problèmes d'ordonnancement pour les systèmes à événements discrets (SED) [Cassandras & Lafortune, 2009 ; Ramadge & Wonham, 1989]. En effet, les ateliers flexibles peuvent être décrits par des modèles dynamiques événementiels. Les SED se distinguent par un espace d'état discret et un changement d'état déclenché par occurrence d'événements. Leur comportement est caractérisé principalement par le parallélisme, la synchronisation, la concurrence, le séquençage et le partage des ressources [Cassandras, 1993] dont la compréhension nécessite l'utilisation d'outils de modélisation qui préservent les propriétés fondamentales du comportement du système [Hervé Hillion, 1989]. Camus [Camus, 1997] présente une description et une comparaison des différents outils de modélisation utilisés pour évaluer les performances des SED [Camus, 1997]. On peut citer les systèmes de transitions, les algèbres de processus, les automates finis et les réseaux de Petri.

Parmi l'ensemble des modèles existants, les réseaux de Petri (PN) [Petri, 1962] ont un intérêt particulier puisqu'ils fournissent une modélisation plus compacte des systèmes basés sur un support graphique. Cela facilite l'expression ainsi que la compréhension des mécanismes de base des systèmes communicants. De plus, les PN ne sont pas liés à un langage particulier et assurent ainsi l'indépendance de la modélisation vis-à-vis des implémentations. Enfin, une troisième raison qui justifie le choix des PN est qu'ils incluent plusieurs outils d'analyse, comportementale (graphe des marquages), structurelle, ainsi qu'une analyse par algèbre linéaire. Plusieurs extensions de PN ont été introduites, plus compactes en termes de description ou plus puissantes en termes de pouvoir d'expression [Diaz, 2003]. Notons que les PN ont été largement utilisés par notre équipe de recherche pour l'étude des SED [Lefebvre el al., 2018 ; Rachidi et al., 2017 ; Ammour et al., 2015] au sein du laboratoire GREAH. Ceci constitue une motivation supplémentaire quant à l'exploitation de ce formalisme.

L'objectif de l'ordonnancement d'un FMS consiste à programmer l'exécution d'une réalisation en attribuant des ressources aux tâches et en fixant leurs dates d'exécution [Carlier & Chrétienne, 1988]. L'utilisation des PN se traduit par la recherche d'une séquence de tirs de durée minimale à partir d'un état initial pour converger vers un état de référence. Cela nécessite la construction du graphe des états accessibles [Karp & Miller, 1969] du modèle PN. Cependant, cette construction, nécessaire pour résoudre le problème d'ordonnancement d'un FMS complexe, n'est pas toujours possible à cause du problème d'explosion combinatoire du nombre d'états due à la complexité du système [Berthomieu & Vernadat, 2003]. En effet, le temps de calcul pour obtenir un ordonnancement optimal augmente

de façon exponentielle par rapport à la taille du problème. Le problème devient encore plus difficile dans le contexte de l'industrie 4.0, et une large littérature a été consacrée à la résolution des problèmes d'optimisation [Luo et al., 2015 ; Lee & DiCesare, 1994 ; Cassandras, 1993]. Pour les problèmes d'ordonnancement de petite taille, obtenir une solution optimale en un temps raisonnable est garanti en utilisant les méthodes exactes [Mazdeh & Rostami, 2014 ; Aggoune, 2002 ; Le Pape, 1995] qui explorent la totalité de l'espace d'état. Cependant, l'obtention d'un ordonnancement optimal, en un temps raisonnable, pour les problèmes de grande taille nécessite l'utilisation de méthodes approchées basées sur l'exploration sélective de l'espace d'état du modèle PN. Une de ces méthodes approchées, appelée recherche en faisceau [Mejía & Niño, 2017 ; Luo et al., 2015 ; Ow, & Morton, 1988], est utilisée dans cette thèse afin de résoudre le problème d'ordonnancement pour les ateliers hybrides.

Les approches d'ordonnancement traditionnelles se concentrent sur les modèles utilisés pour des environnements sûrs et invariants [Cherif, Leclercq, & Lefebvre, 2021a ; Cherif, Leclercq, & Lefebvre, 2019a ; Lefebvre, 2016 ; Baruwa, Piera & Guasch, 2015 ; Mejía, & Montoya, 2009], dans lesquels toutes les données du problème sont supposées être fixées et connues avec exactitude. En effet, ces méthodes utilisent des valeurs exactes comme le temps d'exécution attendu ou la vitesse de la machine afin de représenter l'état du système réel. Cependant, les valeurs exactes de ces paramètres ne sont pas toujours connues à l'avance, et des difficultés surviennent lorsque certains délais de traitement des opérations (qui étaient supposés connus à l'avance) varient en raison d'interruptions.

En réalité, il existe plusieurs types de risques qui peuvent perturber un plan de production et affecter le coût de l'ordonnancement. En effet, des paramètres tels que la disponibilité des matières premières, la fiabilité des machines et les exigences du marché sont souvent sujets à des écarts inattendus. Prenons, par exemple, le délai de livraison dans une usine de véhicules située en Finlande [Holmström & Aaviko, 1994] : la plupart des composants de la voiture arrivent par ferry, et si un camion avec des composants manque un ferry, il doit attendre au moins douze heures pour le prochain départ. Les camions sont fortement exposés à des incertitudes telles que la circulation, les pannes mécaniques, les conditions météorologiques ou les barrages routiers. Ainsi, l'ajout d'une marge de temps pour la livraison des matériaux augmentera la robustesse du calendrier et évitera des perturbations dans la livraison des composants. Par conséquent, il est important de développer des méthodes systématiques pour résoudre le problème de l'ordonnancement dans un environnement incertain, afin de créer des ordonnancements efficaces et fiables.

1.2 Contributions

Trois contributions sont présentées dans ce manuscrit.

La première contribution concerne le calcul incrémental des séquences de commande pour les SED. L'approche permet de modéliser systématiquement des systèmes flexibles hybrides, c'est-à-dire des systèmes où certaines opérations sont effectuées avec des contraintes de précédence totales et d'autres avec une flexibilité totale. L'idée est d'étendre les modèles job shops et open shops à une classe plus large d'ateliers flexibles. La modélisation du problème est réalisée avec des réseaux de Petri T-temporisés (T-TPN) et avec un formalisme systématique à plusieurs niveaux, basé sur la structuration hiérarchique des opérations.

La deuxième contribution concerne le calcul de l'ordonnancement pour les ateliers hybrides. Une nouvelle fonction coût, adaptée aux ateliers hybrides, est introduite pour estimer le temps résiduel jusqu'à la référence. Cette estimation est prouvée être une borne inférieure de la durée réelle. Ensuite, une stratégie de commande a été élaborée, afin de minimiser la nouvelle fonction coût, en allouant de manière optimale un nombre limité de ressources partagées. En effet, une nouvelle méthode appelée Generation Filtered Beam Search (GFBS), qui calcule de manière incrémentale les séquences de contrôle pour les SED complexes, a été présentée. L'algorithme GFBS explore sélectivement l'espace d'état du modèle et retourne une séquence valide de tirs de transition ainsi que le coût résultant.

La troisième contribution concerne le traitement des opérations dans un environnement incertain en raison de l'indisponibilité des ressources et de l'interruption des opérations. L'objectif est de trouver une séquence de commande permettant de rejoindre un état de référence à partir d'un état initial avec un compromis entre performance et risque. Une nouvelle fonction coût, adaptée aux ateliers hybrides, et prenant en compte le temps et le risque résiduel, est introduite. Enfin, un algorithme de recherche en faisceau (Generation Double Filtered Beam search) est proposé pour explorer sélectivement l'espace d'état du modèle.

1.3 Plan du document

Le reste du manuscrit est organisé en cinq chapitres.

Dans le deuxième chapitre, les problèmes d'ordonnancement sont introduits, formalisés et leurs spécificités sont présentées. Les principaux types d'organisation d'ateliers sont présentés. Les outils de formalisation des problèmes d'ordonnancement ainsi que les méthodes de résolution, exactes et approchées, sont présentées. La complexité de la résolution de ces problèmes est également discutée.

Dans le troisième chapitre, l'utilisation des PN pour la modélisation d'un SED ainsi que les organisations usuelles d'ateliers telles que les job shops et open shops sont détaillées. Une nouvelle méthode systématique est proposée afin de permettre la modélisation des ateliers dont les contraintes

de précédences sont partielles : la méthode proposée est basée sur les PN et utilise une technique permettant la modélisation systématique à partir de la description synthétique du système.

Dans le quatrième chapitre, sur la base de la modélisation proposée dans le troisième chapitre, une nouvelle fonction coût est proposée afin de prendre en compte la modélisation itérée du système. Nous prouvons que cette estimation est une borne inférieure de la durée réelle. De plus, une nouvelle variante de la méthode de recherche en faisceau, appelée GFBS (Generation Filteread Beam Search), est détaillée pour calculer des séquences de durée minimale ou quasi minimale en explorant sélectivement l'espace d'état du PN. Cette approche doit également éviter les blocages qui sont a priori inconnus pour le contrôleur. La méthode GFBS est appliquée sur un exemple de la littérature et les résultats obtenus sont comparés avec ceux de la littérature.

Dans le chapitre cinq, la modélisation en présence d'incertitudes est abordée. Une nouvelle fonction coût prenant en compte le risque est proposée. Pour prendre en compte le risque, une mise à jour de l'algorithme GFBS est présentée. Une nouvelle variante de recherche en faisceau appelée Generation Double Filtered Beam Search (GDFBS) est proposée dans le but d'améliorer la recherche d'ordonnancement. Cette nouvelle variante est appliquée sur des exemples de la littérature.

Le chapitre six du mémoire est consacré à la conclusion générale qui résume les résultats obtenus ainsi que les perspectives de poursuite de ces travaux de recherche.

Chapitre 2 : Ordonnancement, spécificités, ateliers et méthodes

2.1. Introduction

La théorie de l'ordonnancement est une branche de la recherche opérationnelle qui a pour objectif de répondre au mieux aux besoins des clients, au meilleur coût et dans les meilleurs délais, en tenant compte de différentes contraintes.

Les problèmes d'ordonnancement affectent tous les secteurs d'activité de l'économie tels que l'administration [Carlier et Chrétienne, 1988], l'informatique [Blazewicz et al., 1996], la construction [Carlier et Chrétienne, 1988], ou encore l'industrie manufacturière [Pinedo, 1955].

Résoudre un problème d'ordonnancement consiste à ordonnancer des tâches [Rodammer & White, 1988] : c'est-à-dire leurs attribuer les ressources nécessaires (matérielles et humaines) et déterminer leurs dates de démarrage, de telle sorte que les contraintes soient respectées pour optimiser un objectif préalablement défini [Lopez & Roubellat, 2001 ; Gotha, 1993].

La planification de l'utilisation de ressources afin de réaliser une grande variété de produits au meilleur coût et dans des délais impératifs, est un garant de la compétitivité. Pour ces raisons, les problèmes d'ordonnancement deviennent de plus en plus complexes. Le résultat d'un ordonnancement présente un calendrier précis de tâches à réaliser avec trois importantes caractéristiques : (1) l'allocation des ressources nécessaires pour l'exécution de chaque tâche, (2) l'ordre de passage de chaque tâche, et (3) les dates de début et de fin d'exécution des tâches.

Dans la première section de ce chapitre, une formalisation des problèmes d'ordonnancement est détaillée et les différentes spécificités sont présentées. La deuxième section présente les principaux types d'organisation d'ateliers. La troisième section s'intéresse aux outils de représentation des problèmes d'ordonnancement. Dans la dernière section, les principales méthodes d'optimisation des problèmes d'ordonnancement utilisées dans la littérature sont représentées.

2.2. Formalisation d'un problème d'ordonnancement

Ordonnancer le fonctionnement d'un système vise la bonne gestion de l'allocation des ressources (les moyens humains et techniques nécessaires pour apporter des modifications à un produit) au cours du temps, afin d'optimiser au mieux un ou plusieurs critères. Pour un problème d'ordonnancement, différentes données interviennent. Ces données, qui seront présentées dans cette partie du chapitre, sont les tâches, les ressources, les contraintes et les critères.

2.2.1 Les tâches

Une tâche ou encore une opération, est une entité élémentaire de travail localisée dans le temps, par une date de début et une date de fin comme le montre la figure 2.

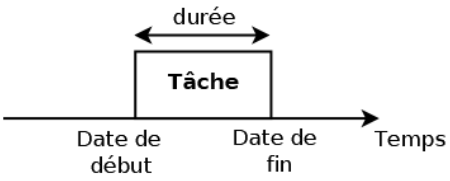


Figure 2 : Représentation d'une tâche

Les opérations sont traitées de façon à optimiser un objectif comme la durée totale de traitement ou le coût total de production. Il existe deux types de tâches :

- Les tâches préemptibles qui peuvent être exécutées en plusieurs fois, permettant à d'autres tâches de les interrompre. Si une tâche est en cours d'exécution et une autre tâche plus prioritaire arrive, le contrôleur peut interrompre la tâche la moins prioritaire au profit de la tâche plus prioritaire.
- Les tâches indivisibles qui doivent être exécutées en une seule fois et ne peuvent pas être interrompues. Le contrôleur ne peut pas arrêter l'exécution d'une tâche indivisible.

L'exécution d'une tâche nécessite généralement l'allocation d'une ou plusieurs ressources qui sont présentées dans la partie suivante.

2.2.2 Les ressources

Les ressources sont des moyens humains ou techniques utilisés pour la réalisation des tâches d'un système. Une tâche nécessite l'allocation d'une ou plusieurs ressources. Les ressources peuvent être classées selon leurs disponibilités au cours du temps en ressources renouvelables et ressources consommables :

- Les ressources renouvelables deviennent à nouveau disponibles après l'exécution de la tâche à laquelle elles ont été allouées.
- Les ressources consommables ne sont plus disponibles après avoir été allouées à une tâche.

D'un autre côté, les ressources renouvelables peuvent être classées selon leurs capacités en ressources disjonctives et ressources cumulatives. Les ressources disjonctives sont les ressources qui ne peuvent

être allouées qu'à une seule tâche à la fois alors que les ressources cumulatives sont les ressources qui peuvent être utilisées par plusieurs tâches simultanément.

Différentes contraintes peuvent s'appliquer et doivent être prisent en compte pour la réalisation d'un ordonnancement. Ces contraintes seront introduites dans la partie suivante.

2.2.3 Les contraintes

Les contraintes et les variables de décisions portent sur des décisions liées au temps et aux ressources. Principalement, la prise de décision dépend des variables temporelles et des variables d'affectation des ressources. Les contraintes expriment des restrictions sur la prise de décision. Deux types de contraintes peuvent être distingués : les contraintes suivant la disponibilité des ressources et les contraintes suivant l'évolution temporelle.

- Contraintes de ressources : plusieurs contraintes peuvent être induites par la nature des ressources. Par exemple, les ressources consommables ne sont plus disponibles après une première utilisation. De plus, la capacité limitée d'une ressource renouvelable impliquera la limitation du nombre de tâches, utilisant la même ressource, à réaliser en parallèle.
- Contraintes temporelles : d'autres contraintes peuvent être induites par des restrictions sur les valeurs que peuvent prendre certaines variables temporelles d'ordonnancement. Ces contraintes peuvent être par exemple des contraintes de précédence, une tâche doit précéder une autre tâche.

Les contraintes varient et dépendent principalement du type d'atelier. Ces contraintes agissent directement sur le résultat et sur la satisfaction des critères de l'ordonnancement.

2.2.4 Les critères

Les critères correspondent à des exigences qui doivent être satisfaites pour permettre d'évaluer la qualité d'un ordonnancement établi : le choix parmi des solutions candidates pour un problème d'ordonnancement se fait par rapport à un ou plusieurs critères. Les critères d'évaluation sont basés sur des indicateurs de performances qu'on cherche à minimiser ou maximiser. Pour une même application, plusieurs critères peuvent être retenus. On distingue deux types de critères, réguliers et irréguliers.

- Les critères réguliers constituent des fonctions décroissantes des dates d'achèvement des opérations telles que la minimisation des dates d'achèvement des tâches ou la minimisation des retards sur les dates d'achèvement des tâches.

- Les critères irréguliers ne sont pas des fonctions monotones des dates de fin d'exécution des opérations. Par exemple, la minimisation du coût de stockage des matières premières ou l'optimisation des changements d'outils.

Les critères peuvent être équivalents si une solution optimale existe pour tous ces critères [Carlier et al., 1988] mais la satisfaction de tous les critères à la fois est très difficile car l'ensemble des critères conduit souvent à des situations contradictoires [Roy & Bouysson, 1993].

De façon générale, un problème d'ordonnancement est composé d'un ensemble de tâches dont l'exécution nécessite l'allocation d'une ou plusieurs ressources pour chaque tâche. L'exécution des tâches est soumise à certaines contraintes afin d'optimiser un ou plusieurs critères.

Les tâches, les ressources, les contraintes et les critères qui ont été déjà présentés dans cette section, forment un atelier dont le type dépend de l'organisation des tâches. Les différents types d'ateliers sont introduits dans la partie suivante.

2.3. Les ateliers

Dans un problème d'atelier, une pièce doit être usinée par différentes opérations qui peuvent être liées exclusivement par des contraintes de précédence. Chaque opération utilise un certain nombre de ressources afin d'être traitée. Les opérations d'un atelier sont regroupées en des entités appelées gammes opératoires permettant d'apporter des modifications sur un produit. Plus précisément, un atelier est caractérisé par les opérations qu'il contient et par son type. Une classification des problèmes d'ordonnancement est donnée dans la figure 3.

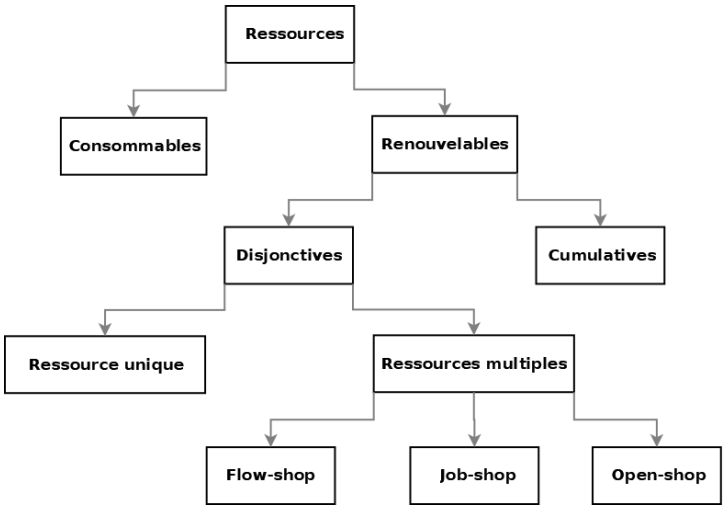


Figure 3 : Classification des problèmes d'ordonnancement

Un atelier est défini principalement par la manière dont les opérations sont liées dans les gammes opératoires qui le composent. Trois types d'atelier sont détaillés dans cette partie du chapitre.

2.3.1 Flow shop

Appelés également ateliers à cheminement unique, ce sont des unités manufacturières où toutes les gammes opératoires sont identiques comme le montre la figure 4. Dans un atelier Flow shop, une gamme opératoire est une ligne de fabrication constituée de plusieurs opérations en série où l'ordre de ces opérations est le même pour toutes les gammes opératoires.

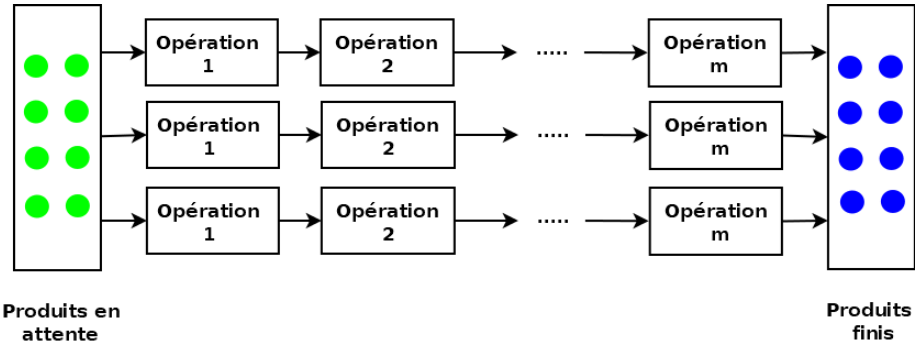


Figure 4 : Atelier Flow shop

2.3.2 Job shop

Appelés également ateliers à cheminement multiple, ce sont des unités manufacturières permettant de traiter une variété de produits suivant différentes gammes opératoires avec des contraintes de précédence totales. Chaque gamme opératoire est constituée par une séquence d'opérations effectuées selon un ordre bien défini comme le montre l'exemple de la figure 5.

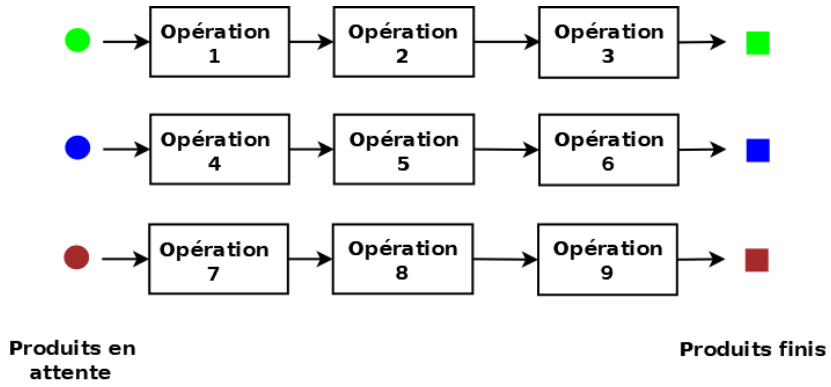


Figure 5 : Atelier Job shop

2.3.3 Open shop

Appelés également ateliers à cheminement libre, ce sont des unités manufacturières permettant de traiter une variété de produits suivant différentes gammes opératoires où aucune contrainte de précédence n'est appliquée. Chaque produit est traité suivant une gamme opératoire constituée par un ensemble d'opérations dont l'ordre n'est pas fixé a priori. Dans l'exemple de la figure 6, dans un atelier Open shop constitué d'une seule gamme opératoire avec deux opérations, le produit peut passer par l'opération 1 puis l'opération 2 ou inversement.

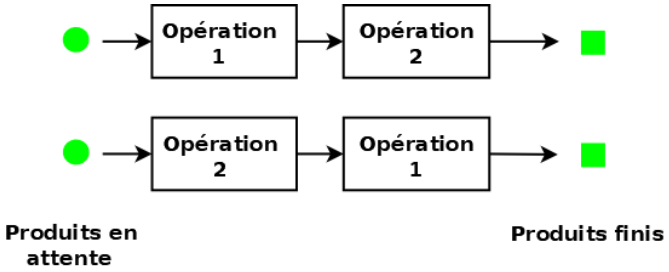


Figure 6 : Atelier Open shop

Ce type d'atelier offre une grande flexibilité pour la planification des tâches mais n'est pas souvent utilisé dans les entreprises car il décrit un modèle peu réaliste : en réalité, un atelier nécessite un minimum de contraintes de précédence entre les tâches. En effet, certaines opérations doivent être réalisées avant d'autres.

2.3.4 Atelier Complexe

Afin de surmenter les limites des ateliers classiques, une classe particulière d'ateliers, appelés *ateliers complexes*, qui satisfait certaines propriétés d'organisation spécifiques, est proposée dans ce travail [Cherif, Leclercq & Lefebvre, 2021a ; Cherif, Leclercq & Lefebvre, 2019a ; Cherif, Leclercq & Lefebvre, 2018]. Un atelier complexe est composé d'un ensemble de gammes opératoires avec des contraintes de précédence partielle qu'on appelle *gammes opératoires hybrides* : certaines opérations ou sous-ensembles d'opérations sont traitées avec des contraintes de précédence totale et d'autres opérations avec une flexibilité totale. Dans l'exemple de la figure 7, un atelier hybride constitué de trois gammes opératoires est présenté : une première gamme opératoire constituée par une séquence d'opérations o_1, o_2 et o_3 traitées selon un ordre bien défini, une deuxième gamme opératoire avec deux opérations o_4 et o_5 traitées avec flexibilité totale et une dernière gamme opératoire avec trois opérations traitées avec des contraintes de précédence partielles. La gamme opératoire hybride

comprend trois opérations o_6, o_7 et o_8 dont o_6 et o_7 sont traitées avec une flexibilité totale suivie du traitement de l'opération o_8.

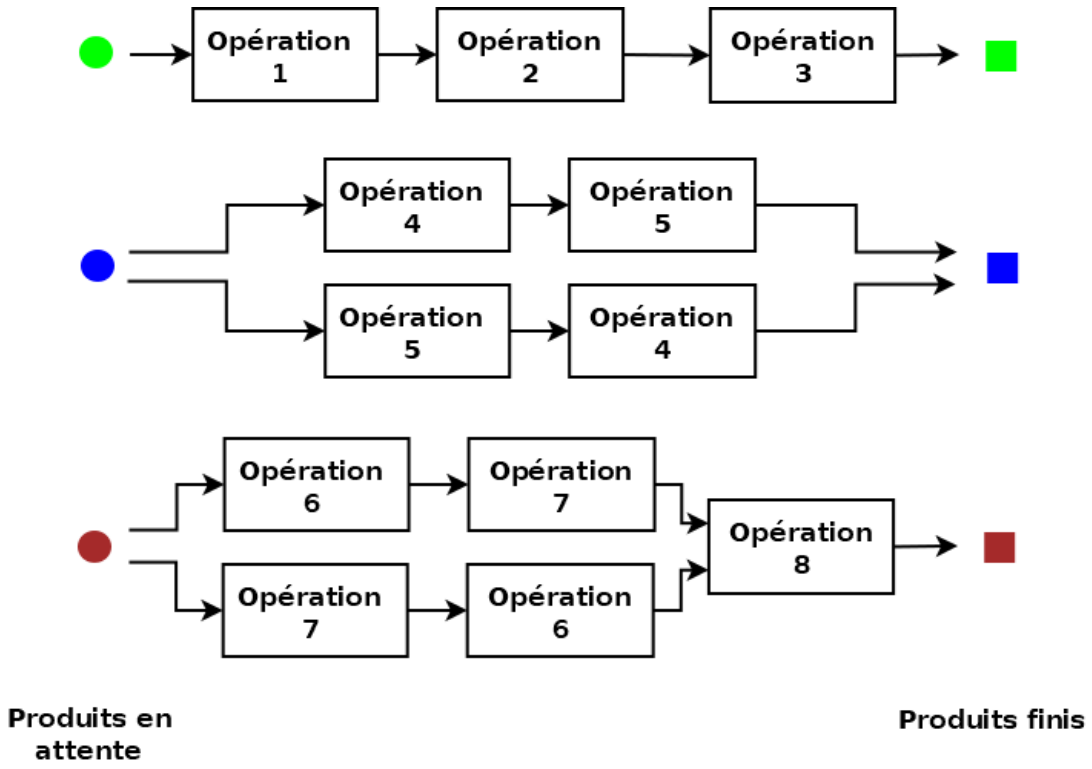


Figure 7 : Atelier complexe

2.4. *Représentation d'un problème d'ordonnancement*

La représentation des problèmes d'ordonnancement permet d'avoir une visualisation graphique de la réalisation de chaque tâche dans le temps.

Afin d'obtenir l'efficacité maximale pour l'agencement du projet, plusieurs représentations ont fait leur apparition. L'objectif est de disposer d'une méthode systématique de planification, de contrôle et de correction. Parmi ces méthodes de représentation, on peut citer les plus connues telles que la méthode PERT et le diagramme de GANTT.

2.4.1 *Méthode PERT*

La méthode PERT, acronyme de «Programm Evaluation and Review Technique», a été mise en œuvre par la marine américaine à la fin des années 50. Dans un graphe PERT, une tâche est représentée par un arc. Pour chaque tâche, une valeur numérique est associée entre parenthèses représentant la durée de la tâche. Entre les arcs figurent des «sommets» représentés par des cercles

marquant l'aboutissement d'une ou plusieurs tâches appelées étapes. Ces cercles sont numérotés afin de suivre l'ordre de succession des divers événements, ou des antécédents, pour une tâche donnée.

Exemple 1 :

Soit les tâches suivantes qui constituent un projet : A(5), B(3), C(1), D(4) et E(6). Les contraintes de précédences sont : A enclenche B, B enclenche C, D enclenche E et E enclenche C.

La méthode des niveaux est utilisée pour construire un graphe PERT. Tout d'abord, les tâches sans antécédents qui constituent le niveau 1 sont identifiées. Par la suite, on détermine les tâches constituant le niveau 2 dont les antécédents sont exclusivement du niveau 1. De la même manière, les autres niveaux sont constitués.

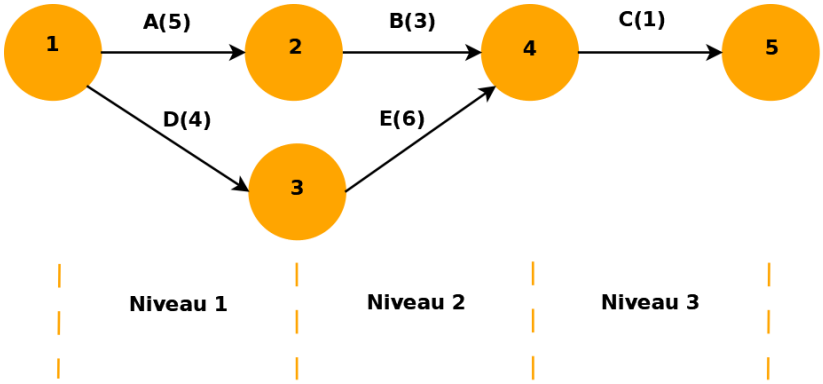


Figure 8 : Exemple de la méthode PERT

2.4.2 Diagramme de Gantt

Ce type de diagramme a été mis au point par Henry Gantt. Au sein d'un tableau, les différentes tâches sont représentées en ligne alors que les unités de temps (exprimées en mois, semaines, jours, heures...) sont représentées en colonnes. Par ailleurs, la durée d'exécution d'une tâche est matérialisée par un trait au sein du diagramme.

La représentation sous forme d'un diagramme de Gantt est la plus utilisée puisque cette représentation a l'avantage d'être beaucoup plus simple et lisible que les autres techniques.

Exemple 2 :

Reprenons l'exemple 1.

La réalisation d'un diagramme de Gantt se fait en plusieurs étapes :

Étape 1 : Identification des relations d'antériorité entre les tâches.

Étape 2 : Représentation des tâches n'ayant aucune antériorité.

Étape 3 : Représentation des tâches dont les tâches antérieures sont déjà représentées jusqu'à représenter toutes les tâches.

Unités de temps	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
A											
B											
C											
D											
E											

Figure 9 : Exemple d'un diagramme de Gantt

2.5. Complexité des problèmes d'ordonnancement

Pour résoudre un problème d'ordonnancement, plusieurs algorithmes peuvent être utilisés. Ces algorithmes ne sont pas équivalents, les différences entre les algorithmes de résolution peuvent être mesurées selon les critères suivants :

o Durée d'exécution ; un algorithme est plus efficace qu'un autre si pour les mêmes données, le temps de calcul est plus court.

o Espace mémoire ; un algorithme est plus efficace qu'un autre s'il utilise moins d'espace mémoire pour résoudre le même problème.

o Robustesse ; un algorithme est plus robuste qu'un autre s'il conserve ses performances malgré l'occurrences de perturbations.

Le but de la théorie de la complexité [Cook, 1971] est d'analyser les coûts de résolution des problèmes étudiés. En effet, cela permet d'établir une classification des problèmes selon les niveaux de difficulté. Selon son degré de complexité en terme durée d'exécution, un problème d'optimisation peut être classé dans l'une des quatre classes suivantes [Sakarovitch, 1984] :

• Les problèmes indécidables : des problèmes pour lesquels il n'existe aucune méthode de résolution.

• Les problèmes de la classe P (dits problèmes polynomiaux) : des problèmes pour lesquels il existe un algorithme polynomial pour leur résolution (la complexité en temps d'exécution est un polynôme).

• Les problèmes de la classe NP (dits problèmes NP-difficiles) : des problèmes qui ne peuvent pas être résolu en un temps polynomial par des algorithmes déterministes [Carlier & Chrétienne, 1988] [Lopez & Esquirol, 1999]. Ces problèmes peuvent être résolus en temps polynomial par des méthodes approchées dont l'optimalité n'est pas démontrable.

• Les problèmes NP-Complets, qui appartiennent à la classe NP et sont résolus, au mieux, en un temps exponentiel.

Pour résoudre un problème d'ordonnancement de manière efficace, il faut prendre en compte la particularité de chaque problème. C'est pourquoi il y a autant d'approches que de problèmes d'ordonnancement. Néanmoins, il existe quelques approches qui fournissent des solutions approchées de bonne qualité pour un certain nombre de problèmes. Dans ce qui suit, nous passons en revue différentes approches exactes ou approchées pour la résolution des problèmes d'ordonnancement.

2.6. Méthodes d'optimisation d'un problème d'ordonnancement

Etant donnés un atelier constitué d'un ensemble de tâches et d'un ensemble de ressources, l'idée est de programmer les tâches et d'affecter les ressources nécessaires afin d'optimiser un ou plusieurs objectifs (critère de performance). Cela se fait en respectant un ensemble de contraintes.

Pour un problème d'optimisation donné, la principale difficulté est celui du choix d'une méthode efficace capable de fournir une solution suffisamment proche de la solution optimale avec un temps de calcul raisonnable.

Les différentes méthodes utilisées pour la résolution des problèmes d'ordonnancement peuvent être classées principalement en deux catégories [Barichard, 2003] (figure 10) : les méthodes exactes dont la complétude de la résolution est garantie et les méthodes approchées qui perdent en complétude pour gagner en efficacité.

Pour les problèmes d'ordonnancement de petite taille, obtenir une solution optimale en un temps raisonnable est garantie en utilisant les méthodes exactes. Cependant, l'obtention d'un ordonnancement optimal, en un temps raisonnable pour les problèmes de grande taille, nécessite l'utilisation de méthodes approchées.

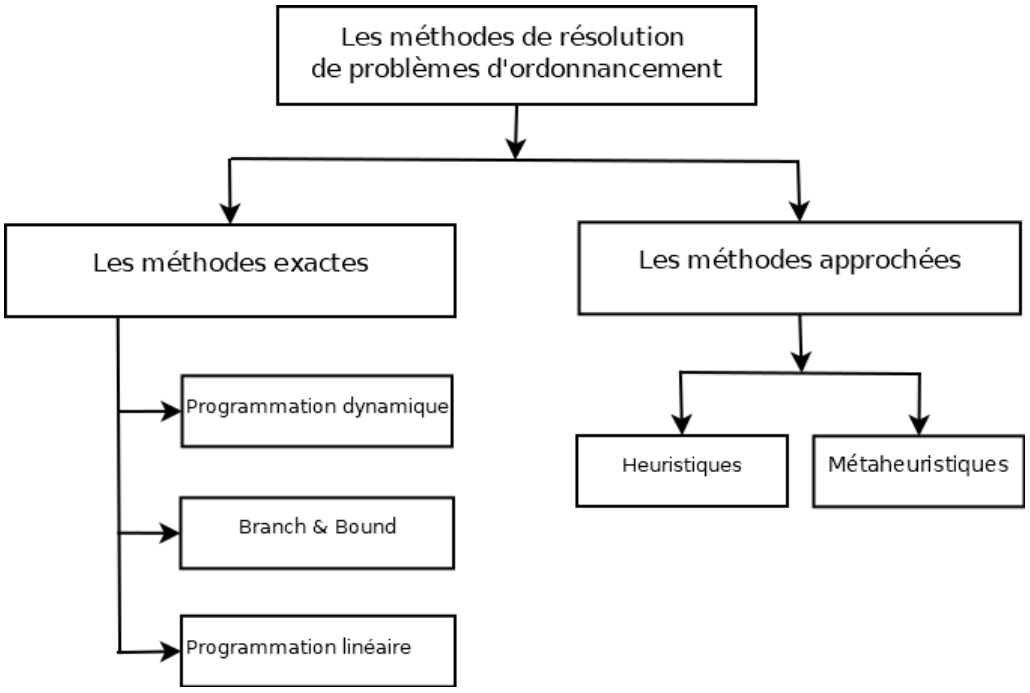


Figure 10 : Méthodes de résolution

2.6.1 Les méthodes exactes

Une méthode exacte peut être définie comme étant une méthode qui fournit, en principe, une solution optimale pour un problème d'optimisation. L'optimalité est mesurée par rapport aux critères de performance établis par le contrôleur [Gargouri, 2003]. Le but des méthodes exactes est de trouver, en un temps de calcul le plus court possible, la solution optimale du problème.

Généralement, il n'est pas possible de résoudre un problème d'optimisation NP-difficile avec un algorithme polynomial. Toutefois, des méthodes efficaces peuvent être mise au point pour une grande partie des énoncés. Pour certains sous-problèmes, la Procédure par Séparation et Evaluation (PSE) (Branch and Bound) [Le Pape, 1995], [Baptiste, 1996] peut être appliquée, pour d'autres, la programmation dynamique ou encore la programmation linéaire.

2.6.1.1 Programmation dynamique

La programmation dynamique est une méthode d'optimisation qui consiste à résoudre un problème en le décomposant en sous-problèmes. Elle est basée sur le principe de Bellman [Bellman, 1986] : « Si un point C appartient à un chemin optimal entre deux points A et B, alors la portion de ce même chemin allant de A à C est le chemin optimal entre A et C ». L'idée est de construire tout d'abord les sous-chemins optimaux ce qui permet ensuite de construire, par récurrence, le chemin optimal pour le problème entier.

La programmation dynamique peut s'avérer couteuse en temps d'exécution et en mémoire. Cependant, elle permet de fournir une solution optimale en temps polynomial pour des problèmes de petite taille [Aggoune, 2002].

2.6.1.2 Procédure par Séparation et Evaluation (Branch and Bound)

La méthode PSE consiste à placer progressivement les tâches sur les ressources par exploration de l'arbre de recherche qui présente toutes les combinaisons possibles. La démarche consiste à trouver la meilleure configuration de manière à éliminer les branches de l'arbre conduisant à de mauvaises solutions [Collette & Siarry, 2002]. Cette méthode est la plus utilisée pour résoudre les problèmes d'optimisation multi-objectifs [Sakarovitch, 1984].

Cette méthode a été définie pour la première fois par [Dantzig et al., 1954] afin de résoudre le problème du voyageur de commerce. Depuis, plusieurs procédures par séparation et évaluation ont été proposées pour optimiser de nombreux problèmes d'ordonnancement [Mazdeh & Rostami, 2014 ; Carlier & Rebaï, 1996].

2.6.1.3 Programmation linéaire

La programmation linéaire présente l'une des techniques classiques et puissantes de la recherche opérationnelle. La méthode repose sur une modélisation mathématique du problème et sur l'utilisation de la méthode du simplex ainsi l'utilisation des algorithmes de points intérieurs de Karmarkar [Sakarovitch, 1984]. Elle consiste à optimiser une fonction coût en respectant des contraintes (ensemble d'équations et/ou d'inéquations linéaires) [Mellouli et al., 2004].

Toutes ces méthodes examinent d'une manière implicite, la totalité de l'espace de recherche pour produire la solution optimale. L'utilisation de ce type de méthodes d'optimisation s'avère particulièrement intéressante dans le cas de problèmes d'ordonnancement de petites tailles. Contrairement aux problèmes de petite taille, lorsque le temps de calcul nécessaire pour obtenir une solution optimale devient excessif, les méthodes approchées constituent une solution efficace pour ces problèmes qui sont souvent NP-difficiles. En effet, les méthodes approchées fournissent une solution quasi-optimale en un temps de calcul raisonnable.

2.6.2 Les méthodes approchées

Les méthodes approchées sont utilisées pour traiter les problèmes d'ordonnancement dans lesquels une solution optimale ne peut pas être trouvée en un temps raisonnable [Liouane, 1998]. Le principe de ces méthodes est de trouver la meilleure solution possible en un temps polynomial. Les méthodes

approchées peuvent être divisées en deux classes : les méthodes basées sur des heuristiques et les méthodes basées sur les métaheuristiques. Dans cette partie, les principales heuristiques et métaheuristiques pour les problèmes d'atelier sont présentées.

2.6.2.1 Les heuristiques

Les heuristiques, des méthodes empiriques, permettent d'obtenir généralement de bons résultats mais ne peuvent pas être démontrables. Ces méthodes sont basées sur des règles simplifiées afin d'optimiser un ou plusieurs critères de performances. Le principe des méthodes heuristiques est d'intégrer des stratégies de décision qui permettent de construire une solution généralement proche de la solution optimale. L'idée est de réduire la performance en faveur du temps de calcul [Kacem, 2003 ; Bel, 2001]. Parmi les stratégies les plus connues, nous distinguons :

- *FIFO* (First In First Out) : la première tâche prête à être traitée est la première à être ordonnancée [Liu, 2000].
- *SPT* (Shortest Processing Time) : parmi les tâches exécutables, celle ayant le temps opératoire le plus petit est exécutée en priorité.
- *LPT* (Longest Processing Time) : parmi les tâches exécutables, celle ayant le temps opératoire le plus grand est exécutée en priorité.
- *EDF* (Earliest Deadline First) ou EDD (Earliest Due Date) : parmi les tâches exécutables, celle ayant la date due la plus petite est exécutée en priorité. Dans le cas où aucune tâche n'est disponible, un temps d'attente est généré [Wang & Kwei-Jay, 1999 ; Mok, 1983].
- *LRF* (Last Release Time First) : parmi les tâches exécutables, celle ayant le plus grand temps écoulé depuis son apparition est la plus prioritaire pour être traitée. Dans le cas où aucune tâche est disponible, un temps libre est généré [Schwiegelshohn, 2004].
- *HPF* (High Priority First) : parmi les tâches exécutables, celle ayant la priorité la plus grande est exécutée en premier. Dans le cas où aucune tâche n'est disponible, alors un temps libre est généré.

En plus de ces stratégies, il existe d'autres heuristiques utilisées principalement pour la recherche de chemin dans un graphe de sommets. Parmi les méthodes les plus connues, on peut citer l'algorithme A* (A star) et l'algorithme de recherche en faisceau.

- La méthode A*

A* est une méthode itérative permettant de rechercher un chemin dans un graphe à partir d'un sommet initial vers un état final [Hart, Nilsson & Raphael, 1968]. Cette méthode ne garantit pas l'obtention

de la solution optimale mais retourne très rapidement une bonne solution (qui peut être optimale dans certains cas).

La méthode utilise une évaluation heuristique pour chaque sommet afin d'estimer le meilleur chemin y passant, classe les sommets par ordre croissant, et visite ensuite les sommets selon cet ordre. C'est une méthode simple qui ne nécessite pas un prétraitement et ne consomme que peu de mémoire.

De par sa simplicité elle est célèbre dans des applications telles que les jeux vidéo [Rabin, 2000] qui privilégient le temps de calcul par rapport à l'exactitude des résultats.

- Algorithme de recherche en faisceau

La recherche en faisceau est une heuristique de recherche qui explore un arbre en traitant un ensemble limité de fils pour chaque sommet [Ow, & Morton, 1988].

L'algorithme utilise la recherche en largeur [Cormen et al., 2001] tout d'abord pour explorer l'arbre de recherche. A chaque niveau de l'arbre, il classe les successeurs des sommets au niveau actuel par ordre croissant selon une fonction coût et garde seulement un nombre prédéterminé (appelé la largeur du faisceau) de successeurs (ceux qui ont le meilleur coût). Seuls les sommets conservés sont développés à l'étape suivante.

Avec une largeur du faisceau infinie, tous les sommets sont traités et l'algorithme devient identique à l'algorithme de recherche en largeur. La réduction de la largeur du faisceau permet de réduire la mémoire nécessaire pour exécuter la recherche. La recherche en faisceau présente une optimisation de l'algorithme de recherche en largeur.

La largeur du faisceau peut être fixe ou variable. Dans le cas où la largeur est variable, si aucune solution n'est trouvée, la procédure est répétée avec une plus grande largeur.

L'exploration partielle de l'arbre de recherche ne garantit pas que l'algorithme retourne la solution si elle existe (la complétude n'est pas garantie) et ne garantit pas que l'algorithme trouve la meilleure solution possible (l'optimalité n'est pas garantie).

La recherche en faisceau est utilisée généralement pour les grands systèmes lorsque l'espace mémoire est insuffisant pour le stockage de l'ensemble de l'arbre de recherche. Par exemple, l'algorithme de recherche en faisceau est utilisé dans de nombreux traducteurs automatiques [Tillmann & Ney, 2018]. La première utilisation d'une recherche en faisceau a été faite par le Harpy Speech Recognition System en 1976, à l'UCM (Université Carnegie-Mellon) [Lowerre, 1976].

Plusieurs variantes de recherche en faisceau ont été proposées pour la résolution de problèmes d'ordonnancement [Mejía & Niño, 2017 ; Luo et al., 2015 ; Mejía, & Montoya, 2009]. Dans ce contexte, cette méthode ainsi que les différentes variantes seront étudiées en détails dans le chapitre 4.

2.6.2.2 Les métaheuristiques

Pour ces méthodes, l'objectif n'est plus la construction d'un ordonnancement à partir des données initiales du problème, mais plutôt la modification d'une solution admissible dont le but est d'améliorer au maximum la valeur de la fonction objectif.

La plupart de ces méthodes utilisent la notion de voisinage (i.e., recherche locale) : à partir d'une solution initiale, générée aléatoirement ou avec une heuristique, un processus itératif est utilisé afin de remplacer la solution initiale par l'un de ses voisins dans le but d'améliorer le résultat selon la fonction objectif. Le critère d'arrêt pour ces méthodes peut être un nombre d'itérations, un temps de calcul maximal ou encore une valeur seuil de la fonction objectif. Dans ce cas, la méthode retourne la meilleure solution trouvée.

Les métaheuristiques diffèrent essentiellement par le système de voisinage utilisé ainsi que par la stratégie de parcours du graphe. Une comparaison de différentes méthodes est illustrée dans [Taillard et al., 2001].

Les métaheuristiques peuvent être classées principalement en deux grandes classes (figure 11) :

- les métaheuristiques à solution unique.
- les métaheuristiques à solutions multiples ou population de solutions.

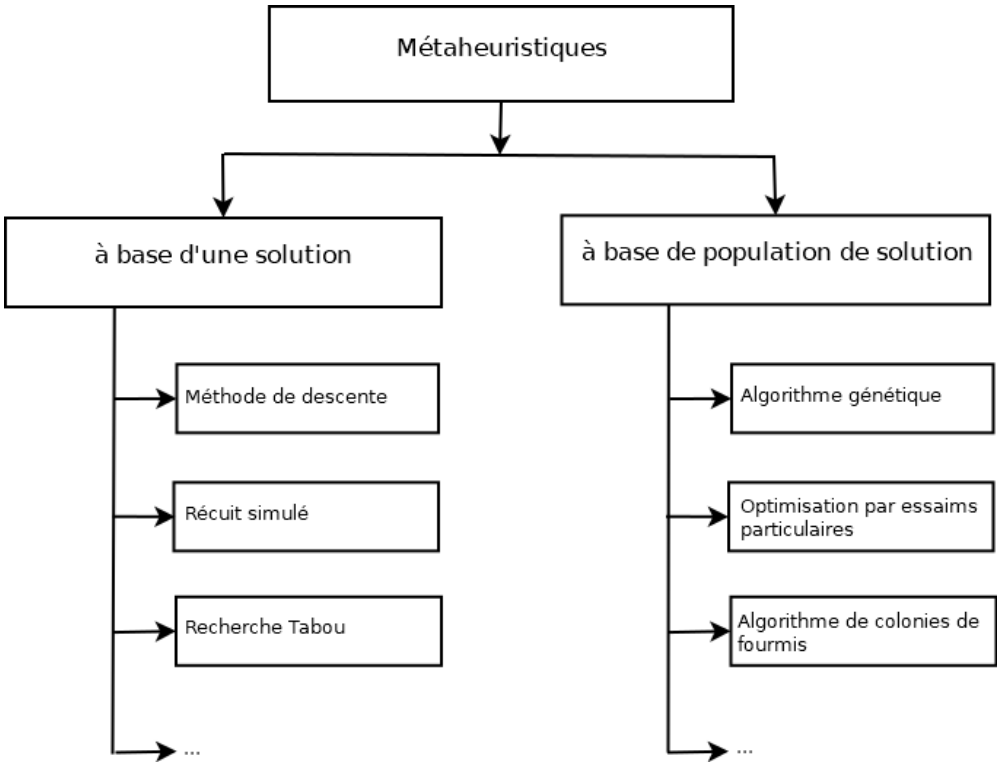


Figure 11 : Classification des métaheuristiques

❖ ***Les métaheuristiques à solution unique***

Les métaheuristiques à solution unique sont aussi appelées méthodes de trajectoire. Elles commencent avec une solution initiale, générée aléatoirement ou par une heuristique, et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche.

Parmi ces méthodes, les plus connues sont le recuit simulé [Kirkpatrick et al., 1983], la recherche Tabou [Glover, 1990 ; Glover, 1989] et la méthode de descente [Allaire, 2005].

➢ Le recuit simulé

Le recuit simulé (Simulated Annealing) est connue pour être la plus ancienne métaheuristique [Blum et al., 2005]. Cette méthode a été introduite par [Kirkpatrick et al., 1983] afin de résoudre des problèmes d'optimisation combinatoire. L'origine de la méthode remonte aux expériences réalisées en métallurgie pour permettre la solidification du métal dans une structure d'énergie minimale [Metropolis et al., 1953].

Le recuit simulé a été utilisé pour résoudre les problèmes d'ordonnancement [Zhang & Wu, 2010 ; Andresen et al., 2008 ; Low, 2005 ; Osman & Potts ,1989]. Le principe de la méthode de recuit simulé est basé sur une procédure itérative qui cherche des solutions avec un coût plus faible.

Le principe de la recherche est d'accepter de manière contrôlée des solutions intermédiaires dégradant la fonction objectif. L'acceptation de solutions moins bonnes par rapport à la solution courante permet à l'algorithme de sortir d'optima locaux.

- La recherche Tabou

Cette méthode est similaire aux méthodes de recherche locales itératives : elle est basée sur la création et l'évaluation d'un voisinage [Karray et al., 2008 ; Kammarti et al., 2005]. Sa principale particularité est la mise en œuvre d'un mécanisme qui combine la recherche locale avec des règles et mécanismes permettant de surmonter le problème d'optima locaux.

La recherche Tabou a été utilisée pour l'optimisation de problèmes d'ordonnancement de type flow shop [Eksioglu et al., 2008], job shop [Zhang & Wu, 2010] ou open job [Liaw, 1999].

- La méthode de descente

La méthode de descente [Bonnans et al., 2006] s'articule autour d'un principe simple : partir d'une solution courante, chercher une solution dans le voisinage et ne l'accepter que si elle est meilleure que la solution existante.

La méthode est basée sur l'amélioration progressive de la solution jusqu'à atteindre un minimum local. Cela présente la limite de la méthode qui souffre de l'absence de diversification.

- ***Les métaheuristiques à solutions multiples***

Les méthodes à solutions multiples traitent un ensemble de solutions. L'idée est d'améliorer, au fur et à mesure des itérations, une population de solutions. L'intérêt des méthodes à solutions multiples est d'utiliser la population comme facteur de diversité.

Parmi ces méthodes, les plus connues sont les algorithmes génétiques [Holland, 1975], l'algorithme de colonies de fourmis [Colorni et al., 1991] et les algorithmes d'optimisation par essaim particulaire [Kennedy et al, 1995].

- Les algorithmes génétiques

Les algorithmes génétiques ont été développés par John Holland [Holland, 1975]. Ces méthodes partent d'une population de solutions initiales de taille déterminée. Cet ensemble de solutions initiales est généré aléatoirement ou avec des méthodes heuristiques.

Les solutions appelées aussi individus ou chromosomes sont caractérisées par une fonction fitness (fonction d'évaluation). L'idée de base consiste à sélectionner les meilleurs chromosomes (appelés parents) pour la reproduction. Par la suite, un opérateur de croisement est utilisé sur les parents afin de produire des enfants.

Les chromosomes peuvent subir une transformation individuelle (appelée opérateur de mutation) pour apporter certaines modifications sur le code génétique de la solution [Mesghouni, 1999]. Ainsi, seules les meilleures solutions sont gardées pour former la population de la génération suivante. Le processus est répété tant que la condition d'arrêt n'est pas satisfaite.

Les algorithmes génétiques sont appliqués dans nombreux domaines tels que la distribution (le problème du voyageur de commerce) [Singh & Baghel, 2006] et l'ordonnancement [Rahman et al., 2015 ; Asadzadeh & Zamanifar, 2010 ; Reeves, 1995].

- Algorithme de colonies de fourmis

Le principe des algorithmes d'Optimisation par Colonies de Fourmis est inspiré du comportement des fourmis qui sont capables d'élaborer des chemins qui s'avèrent souvent être les plus courts entre le nid et une source de nourriture. Ceci est possible grâce à la phéromone : chaque fourmi dépose une quantité de phéromone sur son parcours afin de communiquer avec les autres fourmis. Les fourmis choisissent les chemins avec les plus fortes concentrations de phéromones.

Le but de ces algorithmes est de reproduire le comportement des fourmis afin d'obtenir la meilleure solution possible pour plusieurs types de problèmes [Gagnié, 2001 ; Colorni et al., 1992 ; Colorni et al., 1991].

Les algorithmes de colonies de fourmis ont été appliqués pour la première fois afin de résoudre le problème du voyageur de commerce [Colorni et al., 1992]. La particularité de ces algorithmes est que tous les individus de la population partagent un savoir commun permettant d'orienter leurs futurs choix. Ce savoir commun permet d'indiquer aux autres individus de la population les choix à suivre ou à éviter.

- Les algorithmes d'optimisation par essaim particulaire

La méthode d'Optimisation par Essaim Particulaire (Particle Swarm Optimization algorithm) est une métaheuristique issue d'une analogie avec les animaux qui se déplacent en essaim comme les oiseaux migrateurs et les bancs de poissons [Kennedy, 2011].

Cette méthode possède des similarités avec les algorithmes génétiques et les algorithmes de colonies de fourmis, dans le sens où elles se basent sur le concept d'auto-organisation. En effet, une population d'individus peu intelligents peut trouver la meilleure solution possible et résoudre collectivement un problème d'optimisation.

Cependant, contrairement aux algorithmes basés sur la compétition (élimination des candidats les moins performants) tels que les algorithmes génétiques, l'optimisation par essaim particulaire est basée sur la coopération entre les candidats d'une population [Lemamou, 2009].

L'algorithme d'optimisation par essaim particulaire a été conçu pour résoudre les problèmes d'optimisation continue. Par la suite, l'algorithme a été adapté par divers auteurs afin de résoudre les problèmes d'ordonnancement. Dans ce contexte, [Tasgetiren et al., 2007] et [Jarboui et al., 2008] ont proposé un algorithme d'optimisation par essaim particulaire discret pour résoudre un problème d'ordonnancement Flow-Shop avec la minimisation de l'ordonnancement ainsi que de sa durée.

2.7. Conclusion

Dans la première partie de ce chapitre, nous avons présenté les problèmes d'ordonnancement. Dans une deuxième partie, les organisations usuelles d'atelier ont été discutées. Par la suite, nous avons détaillé des outils de formalisation des problèmes d'ordonnancement. Enfin, les méthodes utilisées pour résoudre ces problèmes ont été introduites dans la dernière partie. Pour conclure, les méthodes exactes présentent un problème d'explosion du nombre des états. Par ailleurs, les méthodes basées sur des heuristiques posent des problèmes de modélisation. Pour ces raisons, nous allons procéder tout d'abord à la modélisation d'un problème d'ordonnancement en utilisant le formalisme structuré des PN qui est à la fois compact et suffisamment simple pour être manipulé (chapitre 3). Par la suite, la recherche d'ordonnancement sera proposée en utilisant une méthode approchée de recherche en faisceau (chapitre 4).

Chapitre 3 : Modélisation des ateliers hybrides par réseau de Petri

3.1. Introduction

Le réseau de Petri (PN) a été introduit par le mathématicien allemand Carl Adam Petri dans sa thèse de doctorat [Petri, 1962]. Cet outil représente un formalisme mathématique avec une représentation graphique intuitive et modulaire. Le PN a été dédié au début à la modélisation des séquences d'événements pour les protocoles de communication [Petri, 1962]. Depuis ce jour, l'utilisation de cet outil a été élargie dans différents domaines grâce à sa capacité d'aide à la conception, la modélisation et l'analyse des SED dans différents domaines. En effet, de nombreux travaux ont utilisés les PN afin de déterminer l'ordonnancement des activités dans les FMS pour profiter pleinement de leur flexibilité dans le but d'atteindre le meilleur rendement [Kim et al., 2007 ; Yu et al., 2003b ; Yu et al., 2003a ; Moro & Kelleher, 2002 ; Lee & DiCesare, 1994]. De plus, les PN ont été utilisés dans les réseaux urbains pour la commande des carrefours à feux [Dezani et al., 2014 ; List & Cetin, 2004]. Les sous-classes de PN utilisées pour cette raison ont été présentées dans la littérature [Liu & barkaoui, 2016].

Dans ce chapitre, nous présentons l'utilisation des PN pour la modélisation d'un SED ainsi que certaines organisations d'ateliers classiques tels que les job shops et open shops. Par la suite une nouvelle méthode systématique est proposée afin de permettre la modélisation d'une nouvelle organisation d'ateliers dont les contraintes de précédences sont partielles. La méthode proposée utilise une technique systématique basée sur des fonctions de base afin de permettre la modélisation de toute organisation à partir de la description synthétique du système.

3.2.Modélisation par réseau de Petri

Cette première partie du chapitre fait appel aux concepts de base relatifs aux réseaux de Petri discrets autonomes et temporisés. Les notations sont introduites pour être utilisées tout au long du travail.

3.2.1 Définition d'un réseau de Petri discret autonome

Un réseau de Petri discret autonome est un graphe biparti orienté composé de deux types de sommets : un ensemble de places et un ensemble de transitions. Les places représentent les variables d'état du système alors que les transitions représentent les événements qui entrainent le passage du système d'un état à un autre. Des arcs orientés associent les places (d'entrée) aux transitions et les transitions aux places (de sortie). Un arc ne relie jamais deux sommets de la même famille.

Dans le cas d'un réseau de Petri marqué, la valeur d'une variable d'état correspond au marquage (nombre de jetons ou de marques) dans la place concernée. Un arc dont l'origine est une transition indique la libération d'un jeton suite à l'occurrence de l'événement correspondant [Silva, 2013].

L'ensemble des marquages de toutes les places représente l'état global d'un système. Le marquage global d'un système est donné sous forme d'un vecteur de marquages.

Pour la présentation graphique :

- Les places sont représentées par des cercles.
- Les transitions sont représentées par des barres horizontales.
- Les arcs sont représentés par des flèches indiquant le sens de circulation des marques.
- Les jetons, ou marques, sont représentés par des cercles pleins placés à l'intérieur des places.

Les arcs peuvent être pondérés en leur attribuant un poids, indiqué au milieu de l'arc, définissant le nombre de ressources libérées suite à l'occurrence de l'événement ou nécessaires à la réalisation de l'événement. Dans ce cas, le PN est dit généralisé. Par défaut, le poids vaut 1 pour tous les arcs du graphe, dans ce cas, le PN est dit ordinaire. Un exemple d'un PN généralisé est donné sur la figure 12.

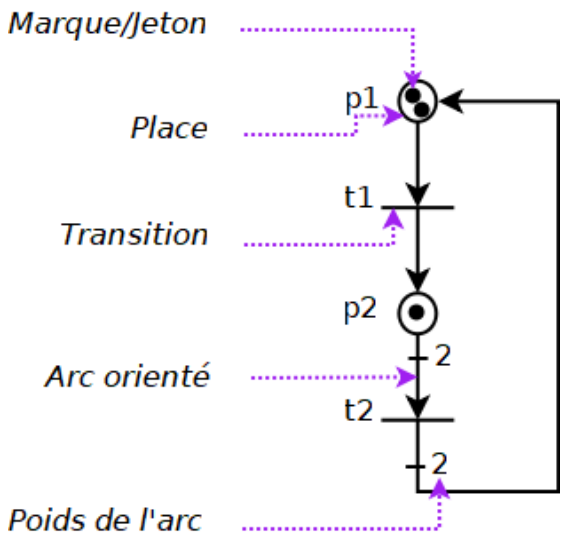


Figure 12 : Réseau de Petri généralisé

Formellement, un réseau de Petri est défini comme une structure $G=<P, T, W_{PR}, W_{PO}>$ dont P et T sont deux ensembles finis non vides où $P=\{p_1,\ldots,p_m\}$ est l'ensemble de m places et $T=\{t_1,\ldots,t_q\}$ est l'ensemble de q transitions. $W_{PO} \in (\mathbf{N})^{m\times q}$ et $W_{PR} \in (\mathbf{N})^{m\times q}$ sont les matrices Pré-incidence et Post-incidence avec $W = W_{PO} - W_{PR} \in (\mathbf{Z})^{m\times q}$ la matrice d'incidence du réseau. Chaque matrice d'incidence est une matrice de m lignes (nombre de places dans le PN) et q colonnes (nombre de transitions dans le PN). $W_{PR}(p_i,t_j)$ correspond au nombre de jetons à enlever de p_i après franchissement de t_j (poids de l'arc allant de p_i vers t_j) et $W_{PO}(p_i,t_j)$ correspond au nombre de jetons à rajouter dans p_i après

franchissement de t_j (poids de l'arc allant de t_j vers p_i). Chaque élément $x \in T \cup P$, possède un ensemble d'éléments d'entrée (placés en amont) symbolisé par $\bullet x$ et un ensemble d'éléments de sortie (placés en aval) symbolisé par $x\bullet$.

Le franchissement d'une transition influence l'état du système et lui permet d'atteindre un autre état. Les conditions d'évolution d'un PN discret autonome seront présentées dans la partie suivante.

3.2.2 Dynamique d'un réseau de Petri discret autonome

Pour définir l'état d'un système à un moment donné, le PN est complété par un marquage. Un marquage M est défini par le nombre de jetons dans chaque place du PN.

$<G, M_0>$ est un PN avec un marquage initial M_0, qui décrit l'état initial du système. A titre d'exemple, le marquage initial de l'exemple de la figure 12 est $M_0=[2\ 1]^T$.

$M \in (\mathbf{N})^m$ représente le vecteur de marquage du réseau désigné par $M=[M(p_1), M(p_2),\ldots, M(p_m)]^T$ qui représente le nombre de jetons dans chacune des places de réseau.

L'évolution de l'état du système correspond à une variation du marquage, les jetons, qui matérialisent l'état du réseau à un instant donné, peuvent passer d'une place à l'autre suite au franchissement ou au tir d'une transition.

Le tir ou franchissement d'une transition t_j entraine l'évolution de l'état du modèle. Une transition t_j doit être validée avant d'être franchie et une transition t_j est validée pour un marquage M s'il y a suffisamment de jetons dans les places en entrée de t_j : $\forall\ p_i \in P$, $M(p_i) \geq W_{PR}(p_i, t_j)$. Le tir d'une transition t_j pour un marquage M enlève un nombre de jetons égale à $W_{PR}(p_i, t_j)$ de chaque place p_i dans $\bullet t_j$ et ajoute un nombre de jetons égale à $W_{PO}(p_i, t_j)$ pour chaque place p_i dans $t_j\bullet$. On écrit $M\ [t_j> M'$ [Ezpeleta, Colom & Martinez, 1995 ; Mejía, Caballero, & Montoya, 2017] où M' est le marquage résultant lorsque la transition activée t_j se déclenche au marquage M.

Pour un PN, l'ensemble des marquages accessibles correspond à toutes les situations possibles que le réseau peut avoir au cours de son évolution à partir d'un marquage initial ; c'est l'ensemble de tous les états atteints après franchissements successifs d'une ou plusieurs transitions. Les marquages accessibles de l'exemple de la figure 12 sont donnés dans la figure 13.

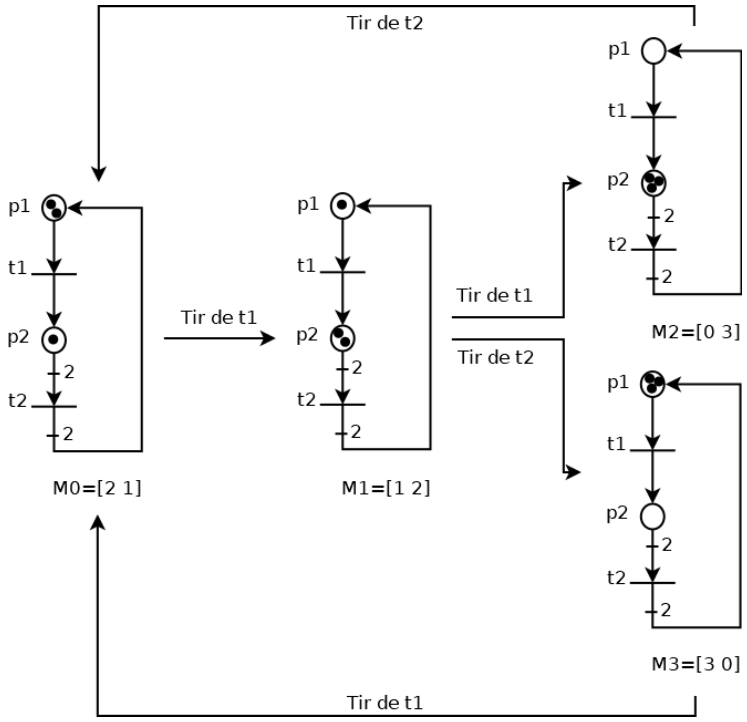


Figure 13 : Marquages accessibles d'un PN

L'évolution d'un système, modélisé par un PN, peut être représentée sous forme d'un graphe des marquages ou graphe d'atteignabilité GA(G ; M_0) dont les sommets correspondent aux marquages accessibles du système [René et al., 2010 ; Compos et al., 1989]. Le graphe des marquages de l'exemple de la figure 12 est donné par la figure 14.

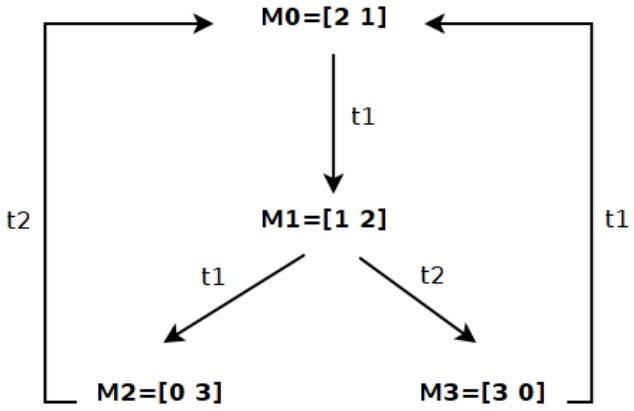


Figure 14 : Graphe des marquages d'un PN

Pour déterminer l'ordre de franchissement des transitions dans un PN, on suppose qu'une seule transition peut être franchie à la fois.

Le franchissement successif de transitions dans un ordre donné à partir d'un marquage donné, pour atteindre un nouveau marquage, constitue une séquence de franchissements. Soit σ une séquence de transitions franchies successivement à partir d'un marquage initial M_0, pour atteindre un marquage M, on écrit : $M_0[\sigma > M$. A titre d'exemple, la séquence de franchissements $\sigma = t_1 t_2$ permet d'atteindre le marquage $M_3 = [3\ 0]^T$ à partir du marquage initial $M_0 = [2\ 1]^T$ de l'exemple de la figure 12.

Afin de calculer un nouveau marquage atteint après franchissement d'une séquence de transitions σ à partir d'un marquage initial, un vecteur de franchissement Γ est défini, Γ représente le vecteur caractéristique de σ, (vecteur de Parikh) :

$$\Gamma = (Y_j)_{\ j=1,\dots,q} \in (\mathbf{N})^q$$

où Y_j désigne le nombre d'occurrences de la transition t_j dans la séquence σ. L'équation d'état de variation du marquage qui permet de calculer le nouveau marquage s'écrit comme suit :

$$M = M_0 + W.\Gamma$$

A titre d'exemple, le marquage M obtenu après franchissement de la séquence $\sigma = t_1 t_2$, à partir du marquage initial $M_0 = [2\ 1]^T$ est calculé de la façon suivante :

$$M = M_0 + W.\Gamma$$

avec : $W = \begin{bmatrix} -1 & 2 \\ 1 & -2 \end{bmatrix}$ et $\Gamma = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Alors, $M = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 & 2 \\ 1 & -2 \end{bmatrix} . \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$

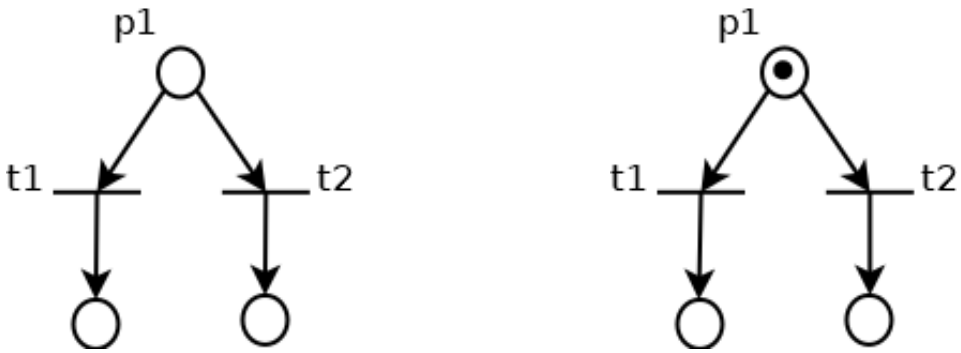
3.2.3 Comportement d'un réseau de Petri discret autonome

Les PN possèdent une grande capacité de modélisation permettant de représenter explicitement certains phénomènes caractéristiques des SED : concurrence, parallélisme, synchronisation ou partage de ressources.

Conflit :

Un conflit structurel (figure 15.a) peut être représenté par PN lorsqu'une place p_i possède au moins deux transitions de sortie : $|p_i^\bullet| > 1$. Cette situation correspond à l'existence d'une concurrence à la consommation des jetons dans une place.

Lorsque la place p_i contient un nombre de jetons qui sensibilise toutes les transitions de sortie mais ne permet pas de toutes les franchir, le conflit est dit effectif (figure 15.b).



a- Conflit structurel b- Conflit effectif

Figure 15 : Exemple de conflit

Synchronisation :

La synchronisation ou rendez-vous, permet de synchroniser les opérations de deux ou plusieurs processus. Cela peut être modélisé par PN lorsqu'une transition t_j possède au moins deux places d'entrée : $\left| {}^{\bullet}t_j \right| > 1$. Un exemple est présenté dans la figure 16 où deux processus sont représentés, le processus 1 (p_1 et p_2) et le processus 2 (p_3 et p_4).

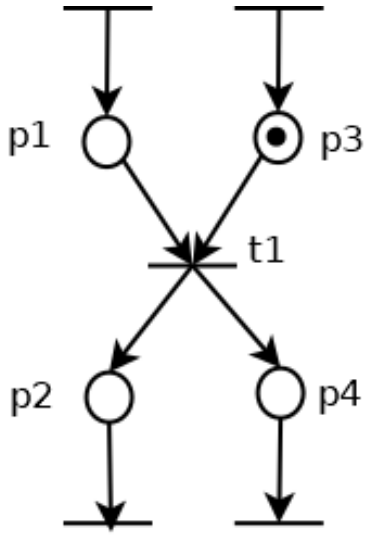


Figure 16 : Exemple de synchronisation

Dans l'exemple présenté figure 16, le franchissement de la transition t_1 ne peut se faire que dans le cas où les deux places d'entrée sont marquées : la place p_1 du processus 1 et la place p_3 du processus 2 contiennent chacune au moins un jeton. Dans le cas contraire, par exemple la place p_1 ne contient pas de jeton, et la place p_3 contient un jeton, le processus 1 reste " bloqué " sur la place p_1.

Parallélisme :

Le parallélisme dans un système représente la possibilité d'évolution de deux ou plusieurs processus de manière simultanée. Cela peut être modélisé par PN selon l'exemple présenté dans la figure 17.

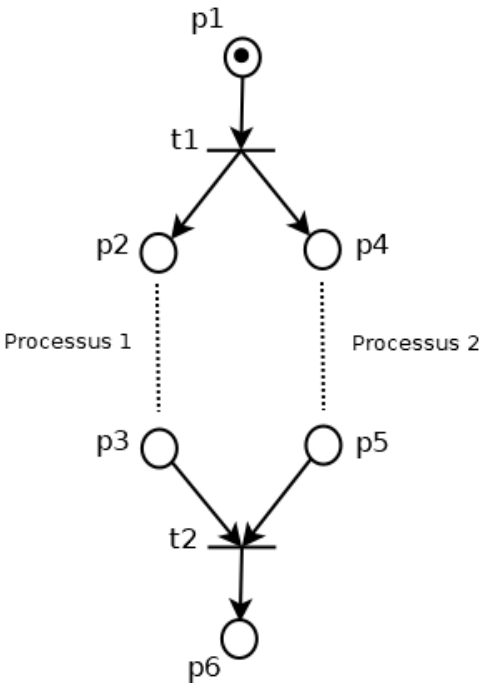


Figure 17 : Exemple de parallélisme

Le départ simultané des processus est provoqué par le franchissement de la transition t_1 qui marque les places en sortie : un jeton dans la place p_2 (qui déclenche le processus 1) et un jeton dans la place p_4 (qui déclenche le processus 2). La fin des deux processus est indiquée par la présence d'un jeton dans chacune des places p_3 et p_5. Pour cela, les deux places p_3 et p_5 doivent être marquées pour l'achèvement des deux processus synchronisés par la transition t_2.

Partage de ressources :

Au sein d'un système, plusieurs processus partagent des ressources tel que le partage d'un robot entre deux gammes opératoires. Un exemple de partage d'une ressource entre deux processus est présenté sur la figure 18.

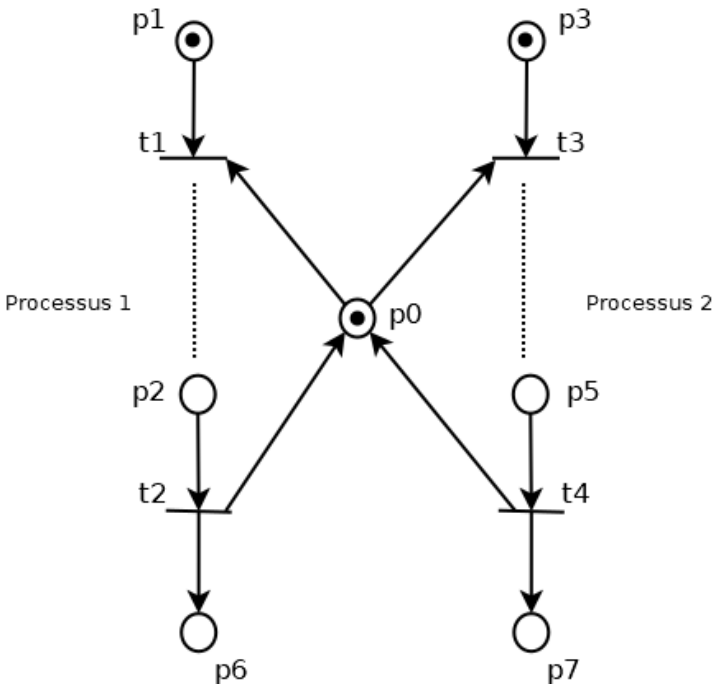


Figure 18 : Exemple de partage de ressource

Le jeton dans la place p_0 représente une ressource mise en commun entre les deux processus 1 et 2. Le franchissement de la transition t_1 lors de l'évolution du processus 1 consomme le jeton présent dans la place p_0. La ressource est alors allouée au processus 1 et n'est plus disponible pour l'évolution du processus 2 (le franchissement de la transition t_3 n'est plus possible). Lorsque la transition t_2 est franchie lors de l'évolution de processus 1, un jeton est redonné à la place p_0 et la ressource est à nouveau disponible pour l'évolution des deux processus.

3.2.4 Propriétés d'un réseau de Petri discret autonome

Dans cette section, les principales propriétés d'un PN sont présentées : les concepts de bornitude, de vivacité, de blocage et réinitialisation.

PN borné :

Cette propriété répond à la question si le nombre de jetons qui circulent dans le réseau reste borné ou non.

Une place p_i du réseau est dite *k-bornée* si pour tout marquage M accessible depuis le marquage initial M_0, $M(p_i) \leq k$. Dans le cas contraire, si $\forall\, k \geq 0$, il existe un marquage M tel que $M(p_i) > k$, la place p_i est dite non bornée.

Un PN est dit borné si quel que soit le marquage accessible atteint à partir du marquage initial et quelle que soit la place p_i, le nombre de jetons dans cette place est inférieur à une borne k. Si k=1,

alors le PN est dit *sauf* ou *binaire* [David et al., 2010]. Un exemple d'un PN binaire est présenté dans la figure 19.

Un exemple de PN non borné est présenté dans la Figure 19. La transition t_1 est toujours validée. Pour chaque franchissement de t_1, un jeton est ajouté dans la place p_2 et $M(p_2)$ peut tendre vers l'infini.

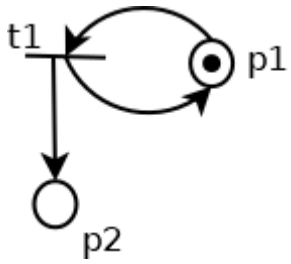


Figure 19 : Exemple d'un PN non borné

PN vivant :

Une transition t_j est dite vivante pour un marquage initial M_0 si pour tout marquage accessible M, il existe une séquence de franchissements à partir de M contenant t_j [David et al., 2010].

Un PN est dit *vivant* pour un marquage initial M_0 si toutes ses transitions sont vivantes pour ce marquage initial : chaque transition d'un PN peut toujours être franchie quel que soit l'état atteint.

Un PN est dit *conforme* s'il est *sauf* et *vivant*.

PN avec blocage :

Un blocage est un marquage pour lequel aucune transition n'est validée. Un PN est dit sans blocage pour un marquage initial M_0 si aucun marquage accessible M n'est un blocage [David et al., 2010].

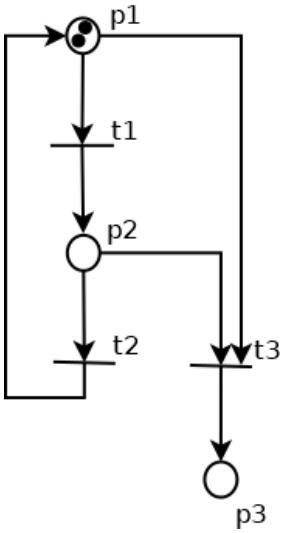


Figure 20 : Exemple d'un PN avec blocage

Un exemple d'un PN avec blocage est présenté dans la figure 20. Le franchissement de la transition t_1 suivi de celui de la transition t_3 amène à une situation de blocage. Ce PN est donc avec blocage.

D'une manière générale, il a été montré que les situations de blocage sont dues soit à la présence de marquages absorbant soit à l'existence de structures particulières appelées siphons [Huang, Jeng, & Chung, 2006]. Dans un PN, un siphon est en ensemble de places tel que s'il est non marqué pour un marquage accessible à partir du marquage initial, il restera indéfiniment non marqué [Murata, 1989]. Un siphon est minimal s'il ne contient pas d'autre siphon.

PN propre (réinitialisable) :

Un PN est dit propre si pour tout marquage *M* accessible à partir du marquage initial M_0, il existe une séquence de franchissements σ qui ramène au marquage initial.
Dans cette partie, le réseau de Petri discret autonome a été présenté, pour lequel l'ordre des événements est pris en compte mais pas leur date d'occurrence. Des contraintes temporelles peuvent être ajoutées afin de conditionner l'évolution d'un système en fonction des dates d'occurrence des événements et des durées nécessaires pour leur réalisation. La partie suivante s'intéresse à une classe de PN qui permet de prendre en compte l'aspect temporel dans un système.

3.2.5 Réseaux de Petri temporisés

Il existe plusieurs classes de PN temporisés (*TPN*), celle qui associe une durée de tir à chaque transition du réseau [Ramchandani, 1974], notée *T-TPN*, et celle qui associe une durée à chaque place de réseau [Sifakis, 1979], notée *P-TPN*. Les auteurs de [David & Alla, 2001 ; David & Alla, 1992] ont démontré qu'il existe une équivalence entre ces deux modèles. Dans ce travail, seuls les *T-TPN* sont utilisés pour représenter le temps nécessaire pour effectuer les opérations.

Un *T-TPN* est un 6-tuple $N = <P, T, W_{PR}, W_{PO}, D, M_0>$ où $<P, T, W_{PR}, W_{PO}, M_0>$ est un PN marqué et $D : T \rightarrow$ R+ est une fonction du temps qui attribue une durée réelle positive $D(t)$ à chaque transition *t* du réseau. Cette durée $D(t)$ représente le temps minimal de franchissement de la transition *t*.

La durée d'activation pour le tir d'une transition *t* est défini pour chaque transition à chaque marquage *M*. Si une transition *t* est activée à l'instant $\tau_{enabled}(t)$, elle se déclenchera au plus tôt à la date $\tau_{enabled}(t) + D(t)$. Dans ce travail, la durée d'activation pour le tir d'une transition *t* (Enabling firing time) $EFT(t) \in [0, D(t)]$ est défini en fonction de temps courant τ par : $EFT(t) = \min(\tau - \tau_{enabled}(t), D(t))$.

Hormis les *TPN*, il existe une autre classe de réseaux de Petri associant le temps. Cette classe est celle des réseaux de Petri temporels qui associe un intervalle de temps à chaque transition [Merlin, 1974].

Le franchissement d'une transition de ce type de réseau doit s'effectuer après une durée qui doit être incluse dans un intervalle de temps [*min* : *max*] dont la borne inférieure *min* correspond au délai minimal de franchissement et la borne supérieure *max* correspond au délai maximal de franchissement.

Dans cette partie du chapitre, le formalisme utilisé dans ce travail a été introduit. La partie suivante s'intéresse à l'utilisation de ce formalisme pour la modélisation d'un SED.

3.3. Spécifications d'un atelier

Un atelier est un système constitué de plusieurs gammes opératoires et un ensemble de ressources. Chaque gamme opératoire est composée d'un ensemble d'opérations qui s'exécutent sur les ressources. Les FMS classiques sont des job shops et des open shops, mais une organisation plus générale pourrait être considérée comme les FMS hybrides étudiés dans ce travail.

Soit $\boldsymbol{J}$ et $\boldsymbol{R}$ respectivement un ensemble de gammes opératoires (constituant un atelier) et un ensemble de ressources. Chaque gamme opératoire $J_h \in \boldsymbol{J}$ est constituée d'un ensemble d'opérations $O_J = \{o_i, i = 1,\ldots,n\}$ dont o_i est l'opération i de la gamme opératoire J. L'ensemble des opérations d'un atelier est notée par $O = \cup \{O_J, J \in \boldsymbol{J}\}$.

Notez que pour des raisons de simplicité, la référence de l'opération d'indice i (i.e. l'opération o_i) à la gamme opératoire correspondante (i.e. la gamme opératoire J) n'est pas explicitement indiquée, mais comme chaque opération o_i appartient à une seule gamme opératoire J, il n'y a pas de confusion.

3.3.1 Modélisation d'une opération

Un FMS est composé d'un ensemble d'opérations qui s'exécutent sur des ressources. Les opérations peuvent être organisées de différentes manières selon le type d'atelier considéré. Les différents types d'ateliers seront détaillés dans la partie suivante.

Une opération $o_i = (d_i, R_i)$ est définie par une durée d_i qui représente le temps nécessaire pour l'exécution de l'opération o_i et un ensemble de ressources $R_i \subseteq \boldsymbol{R}$ utilisées pour effectuer l'opération o_i.

Une opération o_i est modélisée par une place p_i et une transition temporisée to_i avec $D(to_i) = d_i$ appelée transition d'opération, comme le montre la figure 21. Les places des ressources ($R_i = \{r_1,\ldots,r_r\}$ dans l'exemple ci-dessous) sont connectées à la transition to_i à l'aide de boucles (selfloops) : lorsque l'opération o_i est traitée, la ressource r_k, $k = 1,..,r$ est réservée de l'instant d'activation de t_{Oi} jusqu'à l'instant de son franchissement.

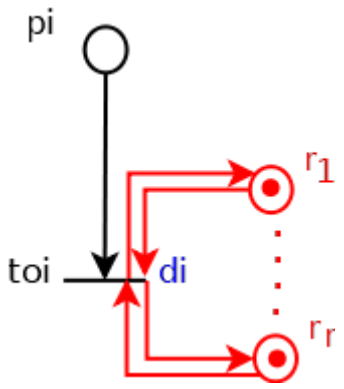


Figure 21 : Modélisation d'une opération

3.3.2 Sémantique temporelle

Les T-TPN sont utilisés afin de modéliser l'ordre des événements et leurs dates d'occurrence dans un SED. Les transitions se déclenchent selon une politique de tir au plus tôt : pour chaque marquage, une transition *t* validée est tirée au plus tôt après un délai minimal *D(t)* à partir de la date à laquelle *t* a été activée. Le déclenchement d'une transition *t* peut se produire immédiatement si aucune spécification de temps n'existe, donc si $D(t) = 0$.

Prenant l'exemple de la figure 22, la transition to_1 peut être franchie à la date τ si elle est validée par le marquage courant au moins depuis la date τ-d_1. En effet, un jeton a été bloqué dans la place p_1 pendant une durée qui n'est pas inférieure à d_1 [David et al., 1992].

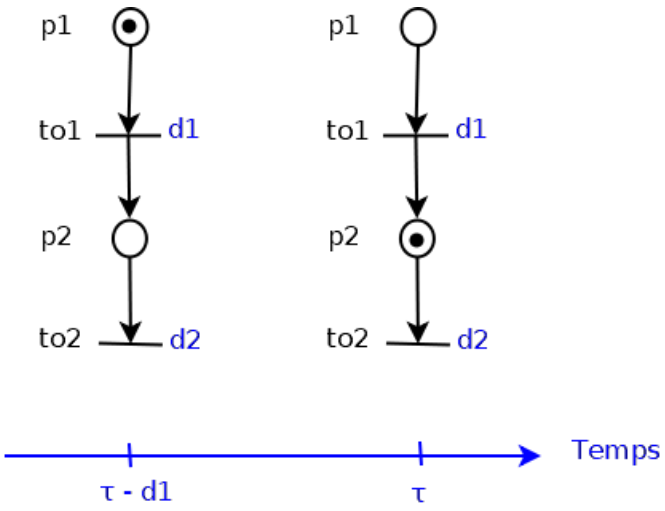


Figure 22 : Franchissement d'une transition temporisée

Une séquence de franchissements temporisée σ de longueur $|\sigma| = K$ et de durée τ_K est définie comme $\sigma = t(\tau_1)t(\tau_2)\ldots t(\tau_K)$ où $\tau_1,\ldots, \tau_K$ représentent les temps des tirs qui satisfont $0 \leq \tau_1 \leq \tau_2 \leq \ldots \leq \tau_K$. La

séquence de tirs temporisée σ, à partir d'un marquage M, conduit à la trajectoire temporisée $(\sigma,M) = M(0)\ [t(\tau_1) > M(1)\ldots.\ M(K\text{-}1)\ [t(\tau_K) > M(K)$ avec $M(0) = M$. La notation $M\ [\sigma > M(K)$ indique qu'il existe une telle séquence de tir chronométrée valide σ du marquage M vers marquage $M(K)$.

Dans un PN, *Path* représente une succession ordonnée de K nœuds $x_1\ x_2\ \ldots\ x_K$ avec $x_k \in T \cup P$ tels que $x_{k+1} \in x_k\bullet$ pour $k = 1,\ldots,K\text{-}1$. La durée de *Path* est définie par :

$$\chi(Path) = \sum_{t \in Path} D(t) \qquad (1)$$

Le chemin *Path* est dit de durée minimale s'il n'existe aucun autre chemin de x_1 à x_K de durée inférieure. Le chemin de durée minimale de x_1 à x_K est noté *Path**. La durée de *Path**(x_1, x_K) est notée par χ_d*(x_1, x_K).

3.3.3 Partage de ressources

Au sein d'un atelier, les ressources sont partagées entre les opérations afin d'optimiser leur utilisation. Une ressource peut être partagée entre deux ou plusieurs opérations, chacune a besoin de l'allocation de cette ressource pour être traitée. A titre d'exemple, deux opérations o_1 et o_2 qui partagent des ressources sont modélisées dans la figure 23, $o_1 = (d_1, R_1)$ avec $R_1 = \{r_1, r_2, r_3\}$ et $o_2 = (d_2, R_2)$ avec $R_2 = \{r_2, r_3, r_4\}$. Deux ressources r_2 et r_3 sont partagées entre les deux opérations o_1 et o_2.

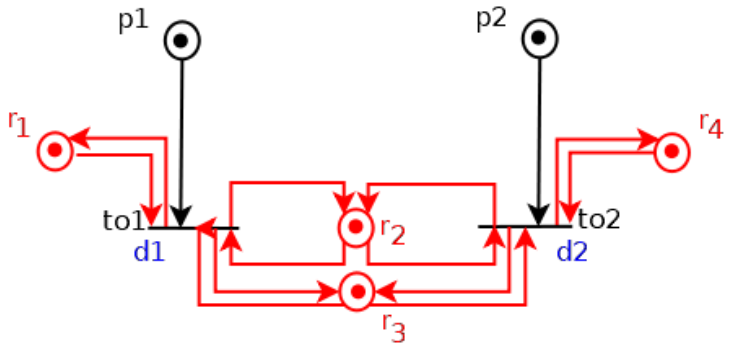


Figure 23 : Partage de ressources entre les opérations

Le franchissement de la transition to_1 de l'opération o_1 entraîne la consommation des jetons présents dans la place p_1 ainsi que la présence d'un jeton dans chacune des places ressources $\{r_1, r_2, r_3\}$ nécessaires pour le traitement de l'opération o_1. Les ressources $\{r_1, r_2, r_3\}$ ne sont alors plus disponibles pendant une durée minimale d_1 durant laquelle l'opération o_2 ne peut pas être traitée puisque le franchissement de la transition to_2 n'est plus possible. Lors du traitement de l'opération

o_1, lorsque la transition to_1 est franchie, un jeton est alors rendu à chacune des places ressources utilisées par o_1, les ressources $\{r_1, r_2, r_3\}$ redeviennent alors à nouveau disponibles et le contrôleur peut traiter l'opération o_2.

Dans un atelier, les opérations peuvent être regroupées de différentes manières selon le type d'atelier auquel elles appartiennent. La section suivante détaille la modélisation de différents types de gammes opératoires.

3.4. Modélisation d'une gamme opératoire par réseau de Petri

Une gamme opératoire est un ensemble d'opérations qui peuvent être liées de différentes façons selon le type d'atelier. Dans cette partie, la modélisation de différentes gammes opératoires est présentée. Par la suite une nouvelle méthode est proposée afin de permettre la modélisation d'une nouvelle organisation d'atelier où les contraintes de précédence sont partielles. Cette méthode utilise une technique systématique basée sur des fonctions de base afin de permettre la modélisation à partir de la description synthétique du système. La technique utilisée construit automatiquement les matrices d'incidence, le vecteur de marquage initial et le vecteur des paramètres temporels de l'atelier. Dans un premier temps, des notations communes utilisées pour modéliser les différentes gammes opératoires d'un FMS sont présentées.

Places :

- $PS = \{s_J, J \in \boldsymbol{J}\}$: ensemble de places de démarrage représentant que la gamme opératoire J est prête à commencer son traitement.

- $CP = \{cp_J, J \in \boldsymbol{J}\}$: ensemble de places de capacité indiquant le nombre maximal de produits qui circulent dans la gamme opératoire J.

-$PI = \{pi_i, o_i \in O_J, J \in \boldsymbol{J}\}$: ensemble de places d'entrée indiquant que les ressources R_i nécessaires pour exécuter l'opération o_i sont disponibles. Notez que pour des raisons de simplicité, la référence des places d'entrée (i.e. la place pi_i) à la gamme opératoire correspondante (i.e. la gamme opératoire J) n'est pas explicitement indiquée, mais comme chaque opération o_i appartient à une seule gamme opératoire J, il n'y a pas de confusion.

- $PE = \{e_J, J \in \boldsymbol{J}\}$: ensemble de places de fin représentant la fin de la gamme opératoire J.

- $PR = \{r_r, r \in R\}$: ensemble de places de ressource représentant le nombre de ressources disponibles de type r.

Transitions :

- $TS = \{ts_J, J \in \boldsymbol{J}\}$: ensemble de transitions de démarrage représentant l'événement par lequel la gamme opératoire J démarre. Ces transitions sont supposées être immédiates (i.e. $f(ts_J) = 0$).

- $TO = \{to_i, o_i \in O_J, J \in \boldsymbol{J}\}$: ensemble de transitions d'opération représentant l'événement par lequel l'opération o_i est exécutée. Une durée d_i représentant le temps minimal nécessaire pour l'exécution de l'opération o_i est associée à chaque transition de l'ensemble TO.

Notez que pour plus de simplicité, la référence des transitions d'opération (i.e. la transition to_i) à la gamme opératoire correspondante (i.e. la gamme opératoire J) n'est pas explicitement indiquée car chaque opération o_i appartient à une seule gamme opératoire J, il n'y a pas de confusion.

- $TES = \{tes_J, J \in \boldsymbol{J}\}$: ensemble de transitions fin-début, qui sont immédiates (i.e. $f(tes_J) = 0$), ce qui signifie que la gamme opératoire J est terminé.

La structure de la matrice d'incidence d'un FMS est présentée dans la figure 24. La matrice est composée de plusieurs blocs, où chaque bloc présente une gamme opératoire qui sera spécifiée ultérieurement, et un bloc particulier représente les ressources partagées entre les gammes opératoire d'un même FMS.

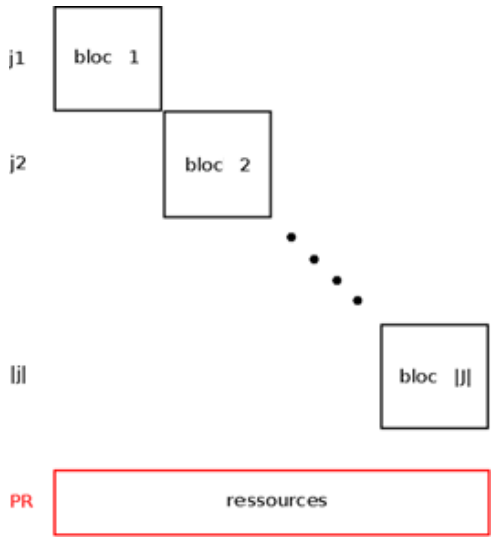


Figure 24 : Structure de la matrice d'incidence d'un FMS

3.4.1 Modélisation d'un job shop

Un job shop est un atelier à cheminement multiple dans lequel les opérations sont organisées avec des contraintes de précédence totale. Une classe de PN appelée *System of Simple Sequential Processes*

with Resources (S3PR) a été définie afin de modéliser les problèmes d'ordonnancement dans les job shops [Xing et al., 1996 ; Ezpeleta et al., 1995].

Plus tard, pour étendre l'utilisation des ressources dans les job shops, une généralisation du modèle S3PR appelée *Systems of Sequential Systems with Shared Resources* (S4R) a été définie par Barkaoui et Ben Abdallah (1996). Cette extension concerne la modélisation de processus séquentiels cycliques partageant plusieurs ressources communes. Dans un modèle S4R, plusieurs ressources partagées sont utilisées par une même opération, ce qui n'est pas le cas pour S3PR où une seule ressource partagée est autorisée à être utilisée pour chaque tâche.

Un job shop est constitué d'une ou plusieurs gammes opératoires où chacune est constituée d'une séquence d'opérations avec des contraintes de précédence totale. Les opérations peuvent avoir des stocks intermediaires pour accumuler les produits entre deux opérations successives.

Un ensemble d'opérations avec des contraintes de précédence totale est modélisé avec un système $<P, T, W_{PR}, W_{PO}, D, M_0>$ où $P = PS \cup CP \cup PI \cup PE \cup PR$ et $T = TS \cup TO \cup TES$.

La matrice d'incidence d'une séquence d'opérations traitée avec des contraintes de précédence totale est présentée sur la figure 25. La matrice est composée de plusieurs blocs, où chaque bloc présente l'existence d'arcs entre un ensemble de places et un ensemble de transitions. Les blocs de type 1 définissent les relations de post-incidence (arcs dirigés des transitions vers les places) et les blocs de type -1 définissent les relations de pré-incidence (arcs dirigés des places vers les transitions).

Pour une gamme opératoire $J \in \boldsymbol{J}$,

- $|PS|=|PE|=|CP|=1$
- $|PI|=|O|$
- $|TS|=1$
- $|TO|=|O|$
- $|TES|=1$

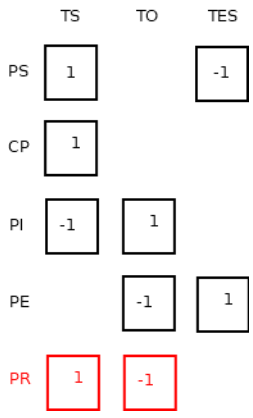


Figure 25 : Matrice d'incidence d'une séquence d'opérations avec des contraintes de précédences totale

Pour chaque séquence d'opérations avec des contraintes de précédence totale $J \in \boldsymbol{J}$, la place de démarrage s_J satisfait $s_J\bullet = \{ts_J\}$, $\bullet s_J = \{tes_J\}$. La place de capacité cp_J satisfait $cp_J\bullet = \{ts_J\}$, $\bullet cp_J = \{tes_J\}$. La place de fin e_J satisfait $e_J\bullet = \{tes_J\}$, $\bullet e_J = \{to_i, o_i$ est la dernière opération de la séquence $J\}$. Chaque opération $o_i \in O_J$ est associée à une place d'opération pi_i. Pour la première opération de la séquence, $pi_i\bullet = \{to_i\}$, $\bullet pi_i = \{ts_J\}$. Pour les autres opérations de la séquence J, $pi_i\bullet = \{to_i\}$, $\bullet pi_i = \{to_{i'}\ o_{i'}$ précède l'opération $o_i\}$.

Une place de ressource r_r est liée à la transition d'opération to_i à l'aide de boucles (selfloops) telles que $r_r\bullet = \bullet r_r = \{to_i$ telle que $o_i \in O_J$, $J \in \boldsymbol{J}$ et $r_r \in R_i\}$.

Seules les places de départ, de capacité et de ressource sont initialement marquées. Pour chaque place de départ, $M(s_J) = 1$, pour chaque place de capacité, $M(cp_J) \geq 1$ et pour chaque place de ressource r, $M(r_r) \geq 1$.

Un exemple de séquence cyclique avec deux opérations J avec une durée minimale $D(J) = d_1 + d_2$ est illustré à la figure 26. Les opérations de la gamme opératoire sont ordonnées : la tâche o_1, qui nécessite la ressource r_1, est traitée en premier puis o_2, qui nécessite la ressource r_2, est traitée en second.

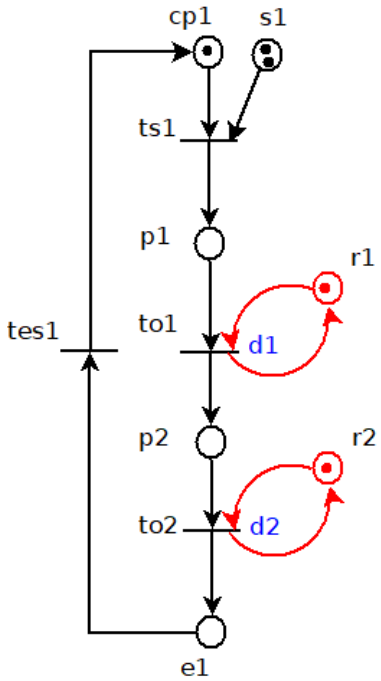


Figure 26 : Modélisation d'un job shop

La durée minimale d'une seule exécution du job *J* composé d'une séquence d'opérations et sans ressource partagée est donnée par (2) :

$$D(J) = \sum_{to_i \in J} D(to_i) \tag{2}$$

3.4.2 Modélisation d'un open shop

Les *T-TPN* sont également utilisés pour modéliser un ensemble d'opérations avec une flexibilité totale (ordre totalement libre). Un open shop pourrait être modélisé à l'aide de S3PR ou S2OPR (*set of simple open processes with resources*). Une comparaison des tailles du S3PR et du S2OPR équivalent a été effectuée [Mejía et al., 2017] : pour les systèmes de grande taille où les opérations sont effectuées avec une flexibilité totale, S2OPR est beaucoup plus petit que son modèle S3PR équivalent mais la capacité de celui-ci doit être limitée à 1.

Dans le cas de l'utilisation de S2OPR, des contrôleurs composés de « drapeaux » et de compteurs (définis par la suite) sont utilisés pour la modélisation de la flexibilité totale [Mejía et al., 2017]. Un ensemble d'opérations avec une flexibilité totale sans ressource partagée a une durée minimale *D(J)* calculée avec (2).

Un ensemble d'opérations avec une flexibilité totale est modélisé avec un système $<P, T, W_{PR}, W_{PO}, D, M_0>$ où $P = PS \cup CP \cup PI \cup PN \cup PE \cup PC \cup PF \cup PR$ et $T = TS \cup TB \cup TO \cup TI \cup TES$.

En plus des ensembles de places et transitions précédemment définis, les ensembles suivants sont considérés :

$PN = \{pn_{i\ldots i'}, \{o_{i,\ldots,}o_{i'}\} = O_J\}$: ensemble de places indiquant que la première opération est prête à commencer.

$PO = \{po_{i\ldots i'}, \{o_{i,\ldots,}o_{i'}\} = O_J\}$: ensemble de places indiquant que toutes les opérations sont traitées.

$PF = \{f_i, o_i \in O_J\}$: ensemble de places « drapeau » qui appliquent la contrainte de non-recirculation. Une place « drapeau » définie dans [Mejía et al., 2017] garantit que le système effectue chaque opération une seule fois.

$PC = \{c_J\}$: ensemble de places compteur qui évitent une réalisation anticipée. Une place compteur définie dans [Mejía et al., 2017] garantit que le système passe par toutes les opérations avant de sortir.

$TB = \{tb_i, o_i \in O_J\}$: ensemble de transitions de début représentant l'événement par lequel l'opération o_i commence. Ces transitions sont supposées être immédiates (i.e. $D(tb_i) = 0$).

$TI = \{ti_{ii'}, o_i \in O_J, o_{i'} \in O_J\}$: ensemble de transitions représentant les événements de passage d'une opération à une autre dans lesquels l'opération o_i de la gamme opératoire J est exécutée, et que la prochaine opération $o_{i'}$ est prete à etre exécutée. Ces transitions sont temporisées avec une durée d_i représentant le temps nécessaire pour exécuter l'opération o_i.

La matrice d'incidence d'un ensemble d'opérations avec une flexibilité totale est similaire à la matrice illustrée à la figure 27.

Remarque : On note A_n^K l'arrangement de k éléments parmi n.

Pour une gamme opératoire $J \in \boldsymbol{J}$,

- $|PS|=|CP|=|PE|=|PC|=|PN|=|PO|=1$
- $|PI|=|O_J|$
- $|PF|=|O_J|$
- $|TS|=|TES|=1$
- $|TB|=|TO|=|O_J|$
- $|TI|=\boldsymbol{A^2_{|OJ|}}$

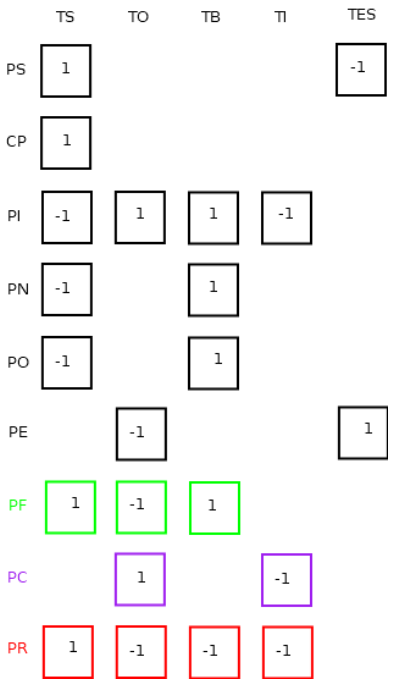


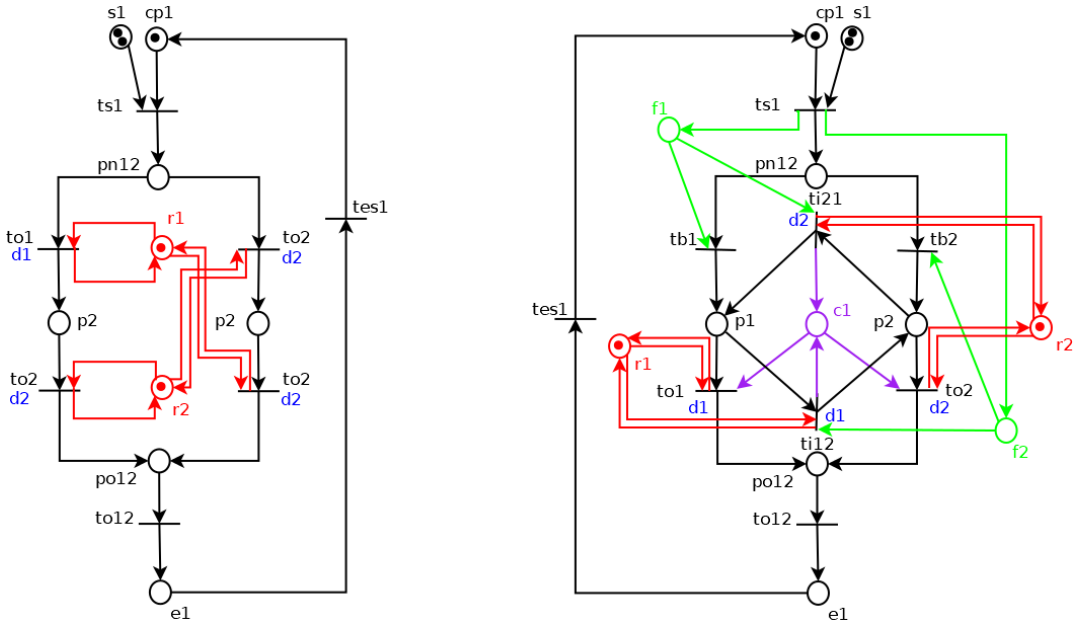
Figure 27 : Matrice d'incidence d'un ensemble d'opérations avec flexibilité totale

Pour chaque ensemble d'opérations avec flexibilité totale $J \in \boldsymbol{J}$, la place de démarrage s_j satisfait $s_J\bullet = \{ts_J\}$ et $\bullet s_J = \{tes_J\}$. La place de capacité cp_J satisfait $cp_J\bullet = \{ts_J\}$ et $\bullet cp_J = \{tes_J\}$. La place de fin e_J satisfait $e_J\bullet = \{tes_J\}$ et $\bullet e_J = \{to_i, o_i \in O_J\}$. La place $pn_{i...i'}$, telle que $\{o_i,...,o_{i'}\} = O_J$, satisfait $pn_{i...i'}\bullet = \{tb_i, o_i \in O_J\}$ et $\bullet pn_{i...i'} = \{ts_J\}$. La place $po_{i...i'}$, telle que $\{o_i,...,o_{i'}\} = O_J$, satisfait $po_{i...i'}\bullet = \{to_{i...i'}, \{o_i,...,o_{i'}\} = O_j\}$ et $\bullet po_{i...i'} = \{to_i, o_i \in O_j\}$. Chaque opération $o_i \in O_J$ est associée à une place d'opération pi_i telle que $pi_i\bullet = \{to_i\} \cup \{ti_{ii'}$ telle que l'opération o_i précède l'opération $o_{i'}\}$, $\bullet pi_i = \{tb_i, o_i \in O_J\} \cup \{ti_{i'i}$ telle que l'opération $o_{i'}$ précède l'opération $o_i\}$.

Une place de ressource r_r est liée à la transition d'opération to_i à l'aide de boucles (selfloops) telles que $r_r\bullet = \bullet r_r = \{to_i$ telle que $o_i \in O_J$, $J \in \boldsymbol{J}$ et $r_r \in R_i\} \cup \{ti_{i'i}$ telle que $o_i \in O_J$ avec $J \in \boldsymbol{J}$, $r_r \in R_i$ et l'opération o_i précède l'opération $o_{i'}\}$.

Seules les places de départ, de capacité et de ressource sont initialement marquées. Pour chaque place de départ, M (s_J) = 1, pour chaque place de capacité, M (cp_J) ≥1 et pour chaque place de ressource r, M (r_r) ≥1.

Un exemple de deux opérations avec une flexibilité totale et une durée minimale $D(J) = d_1 + d_2$ est illustré à la figure 28. La modélisation utilisant S3PR est représentée sur la figure 28.a tandis que son équivalent S2OPR est représenté sur à la figure 28.b. Les opérations de la gamme opératoire ne sont pas ordonnées : le contrôleur peut traiter o_1, qui nécessite la ressource r_1, puis o_2, qui nécessite r_2, ou o_2 puis o_1. Le contrôleur est composé des places « drapeau » f_1 et f_2 et de la place compteur c_1.



a- Modélisation avec S3PR b- Modélisation avec S2OPR

Figure 28 : Modélisation d'un open shop

Les deux gammes opératoires présentées permettent de lier les opérations soit avec des contraintes de précédence totale (le cas d'un job shop) soit avec une flexibilité totale (le cas d'un open shop). Afin d'étendre la modélisation d'un SED, une nouvelle classe de gammes opératoires permettant de regrouper les opérations avec des contraintes de précédence partielle sera définie dans la partie suivante.

3.4.3 Modélisation d'une gamme opératoire hybride

Une classe particulière de gammes opératoires, considérée comme une gamme opératoire hybride, qui satisfait certaines propriétés d'organisation spécifiques, est considérée. Une gamme opératoire hybride est composée d'un ensemble d'opérations avec des contraintes de précédence partielle : certaines opérations ou sous ensemble d'opérations sont traitées avec des contraintes de précédence totale et d'autres opérations avec une flexibilité totale.

L'organisation d'une gamme opératoire hybride peut être décomposée de manière itérée en un ensemble de gammes opératoires de base. Ces gammes opératoires sont soit des séquences d'opérations avec des contraintes de précédence totale, soit des ensembles d'opérations avec une flexibilité totale. Une méthode de conception systématique a été introduite dans nos travaux [Cherif, Leclercq & Lefebvre, 2021a ; Cherif, Leclercq & Lefebvre, 2019a ; Cherif, Leclercq & Lefebvre, 2018] pour coder l'organisation itérée. Il est basé sur certaines fonctions de base.

Operation (d_i, R_i) est la fonction, présentée dans la figure 29, qui modélise l'opération o_i. Elle utilise en entrée la durée de l'opération d_i et l'ensemble de ressources R_i nécessaires à cette opération et retourne l'objet «opération». Une opération peut être considérée comme une gamme opératoire élémentaire.

Figure 29 : La fonction Operation( )

Sequential (J_1, J_2, ..., J_h) est la fonction, présentée dans la figure 30, qui utilise en entrée *h* gammes opératoires (éventuellement les gammes opératoires élémentaires) et retourne la «Séquence» de gammes opératoires résultante *J*, modélisée par la séquence de J_1 à J_h.

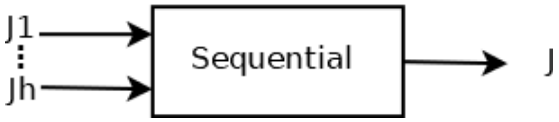


Figure 30 : La fonction Sequential( )

Open (J_1, J_2, ..., J_h) est la fonction, présentée dans la figure 31, qui utilise en entrée *h* gammes opératoires (éventuellement les gammes opératoires élémentaires) et retourne la gamme opératoire résultante «Open» *J* comme un ensemble de gammes opératoires J_1 à J_h avec une flexibilité totale.

Figure 31 : La fonction Open( )

De plus, les opérations incluses dans les gammes opératoires hybrides partagent certaines ressources.

Afin de modéliser une gamme opératoire hybride, une organisation itérée basée sur les fonctions précédentes et un ensemble de gammes de base sont utilisées. L'idée est d'introduire des opérations dynamiques (qui seront représentées par des doubles cercles et des doubles rectangles dans les figures suivantes). Chaque opération dynamique d'un niveau l donné intègre les modèles de sous ensembles d'opérations obtenus aux niveaux *1*,…, *l*-1. Le modèle entier avec la gamme de base unique est obtenu au niveau final *L*. Les autres opérations du modèle seront appelées opérations statiques. Une gamme de base est modélisée avec une combinaison d'opérations statiques et dynamiques. Soit $J_l = \{J_{1l}, .., J_{kl}\}$ l'ensemble des gammes de base au niveau l et {1, .., *L*} l'ensemble des niveaux. La construction est illustrée en utilisant l'exemple ci-dessous.

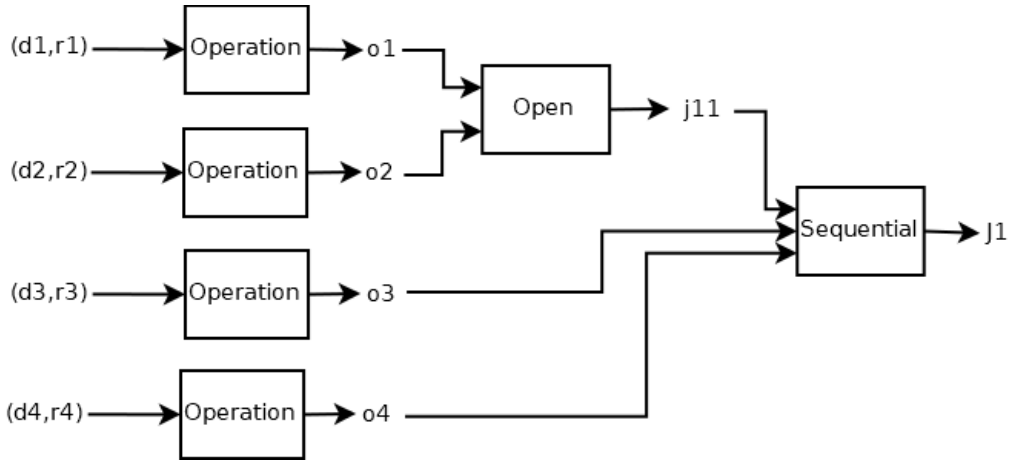


Figure 32 : Conception d'une gamme opératoire hybride

Comme le montre la figure 32, l'exemple est composé de quatre opérations o_1, o_2, o_3 et o_4. Les opérations o_1 et o_2 sont traitées avec une flexibilité totale (le contrôleur peut traiter o_1, qui nécessite la ressource r_1, puis o_2, qui nécessite r_2, ou o_2 puis o_1). Par la suite, l'opération o_3, qui nécessite la ressource r_3, est traitée et enfin l'opération o_4, qui nécessite la ressource r_4, est traitée. Le contrôleur est composé des places drapeau f_1 et f_2 et de la place compteur c_{11}.

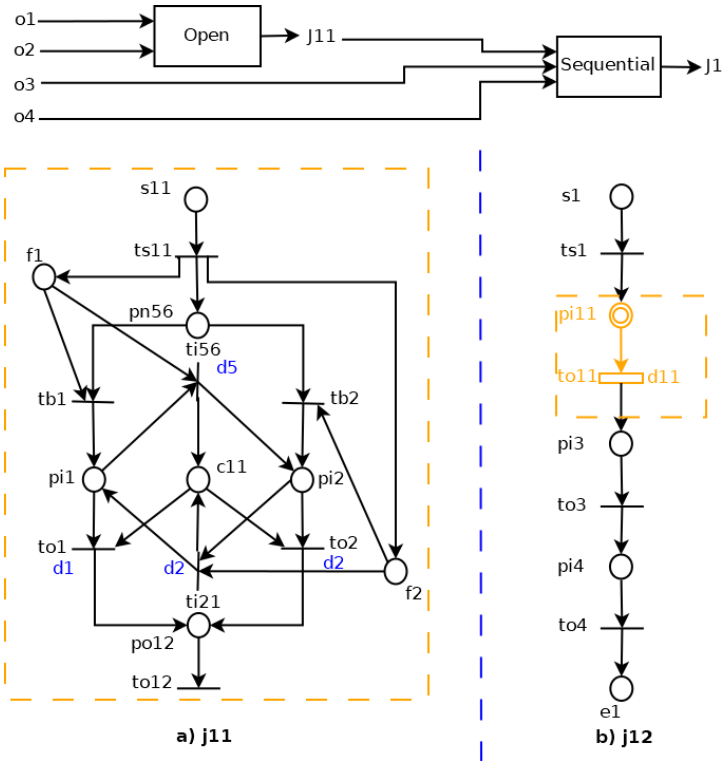


Figure 33 : Construction d'une gamme opératoire hybride

L'organisation de la gamme opératoire J_1, composée de deux gammes de base $\{J_{11}, J_{12}\}$, est présentée dans la figure 33 où J_{11} est la seule gamme de base du premier niveau et J_{21} appartient au deuxième et dernier niveau.

La gamme de base du premier niveau J_{11} est composée de deux opérations statiques avec une flexibilité totale $\{o_1, o_2\}$. La gamme de base du deuxième niveau J_{12} est une séquence de trois opérations : une opération dynamique o_{11} et deux opérations statiques $\{o_3, o_4\}$. Afin d'obtenir le modèle complet J_1, l'opération dynamique o_{11} est remplacée par la gamme de base J_{11}. Le modèle T-TPN résultant de la gamme opératoire hybride est présenté sur la figure 34. Notez que les ressources sont également modélisées.

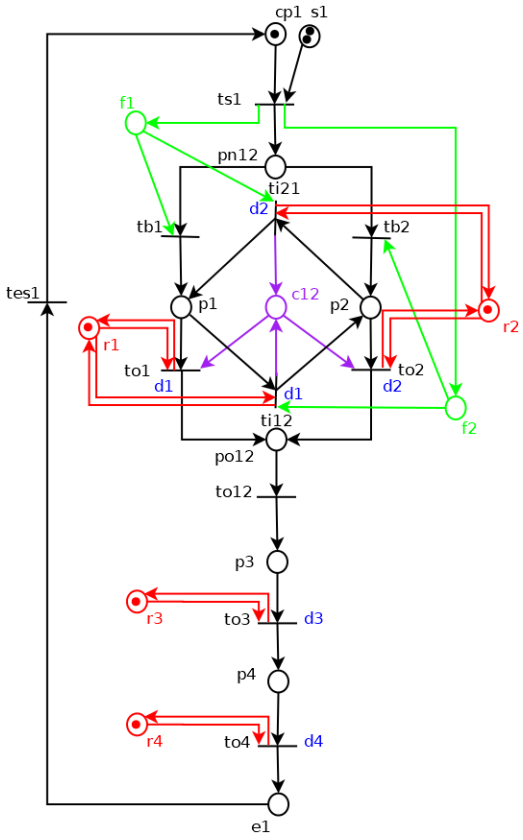


Figure 34 : Gamme opératoire hybride

La structure de la matrice d'incidence d'une gamme opératoire hybride est similaire à celle d'une gamme opératoire avec flexibilité totale présentée sur la figure 27, et les mêmes sous ensembles de places et transitions sont utilisés.

Seules les places de départ, de capacités et de ressources, sont initialement marquées. Pour chaque place de départ, $M(s_J) = 1$, pour chaque place de capacité, $M(cp_J) \geq 1$ et pour chaque place de ressource r, $M(r_r) \geq 1$.

La notion de gamme opératoire hybride est étendue aux ateliers composés de différents types de gammes opératoires. Ce type d'ateliers qui sera défini en détails dans la partie suivante est nommé *atelier complexe*.

3.5. Modélisation d'un atelier complexe

Un FMS est composé d'un ensemble de gammes opératoires liées à travers les ressources partagées. Dans un atelier complexe, les différents types de gammes opératoires présentés dans la partie précédente peuvent exister. En effet, chacune des gammes opératoires incluses dans le FMS peut être :

- Une séquence d'opérations avec des contraintes de précédence totale.
- Un ensemble d'opérations avec une flexibilité totale.
- Un ensemble d'opérations avec des contraintes de précédence partielle.

Le formalisme systématique à plusieurs niveaux, introduit pour modéliser une gamme opératoire hybride, est aussi utilisé pour modéliser le FMS complexe. Ce formalisme est basé sur la structuration hiérarchique de différentes opérations de l'atelier.

Un atelier est constitué d'un ensemble de ressources $\boldsymbol{R}$ et de gammes opératoires $\boldsymbol{J}$ où chaque gamme opératoire $J \in \boldsymbol{J}$ est constituée d'un ensemble d'opérations $O_J = \{o_i, i = 1,\ldots,n\}$ où o_i est l'opération i de la gamme opératoire J. L'ensemble des opérations d'un atelier est notée par $O = \cup \{O_J, J \in \boldsymbol{J}\}$. Une opération o_i est définie par une durée d_i et un ensemble de ressources $R_i \subseteq \boldsymbol{R}$ nécessaires pour son exécution.

Un exemple de FMS complexe est présenté sur la figure 36. L'atelier est composé de trois différentes gammes opératoires J_1, J_2 et J_3 avec 8 ressources dont 3 ressources partagées r_1, r_3 et r_6. J_1 consiste en une séquence d'opérations o_1, o_2 et o_3 avec des contraintes de précédence totale. J_2 est une gamme opératoire hybride qui comprend trois opérations o_4, o_5 et o_6 dont o_4 et o_5, présentant une gamme de base J, sont traitées avec une flexibilité totale suivie du traitement de l'opération o_6. J_3 consiste en deux opérations o_7 et o_8 traitées avec une flexibilité totale. La figure 35 montre la conception du FMS selon les 3 fonctions de base décrites précédemment.

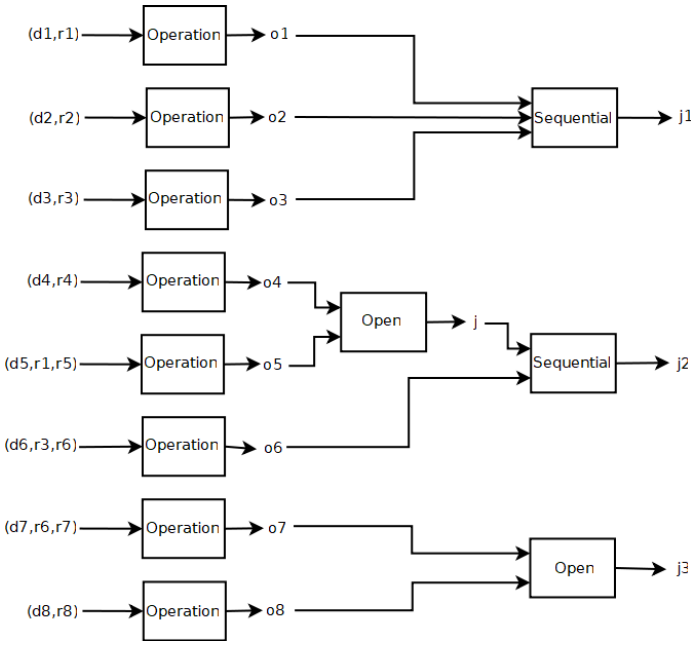


Figure 35 : Conception d'un FMS complexe

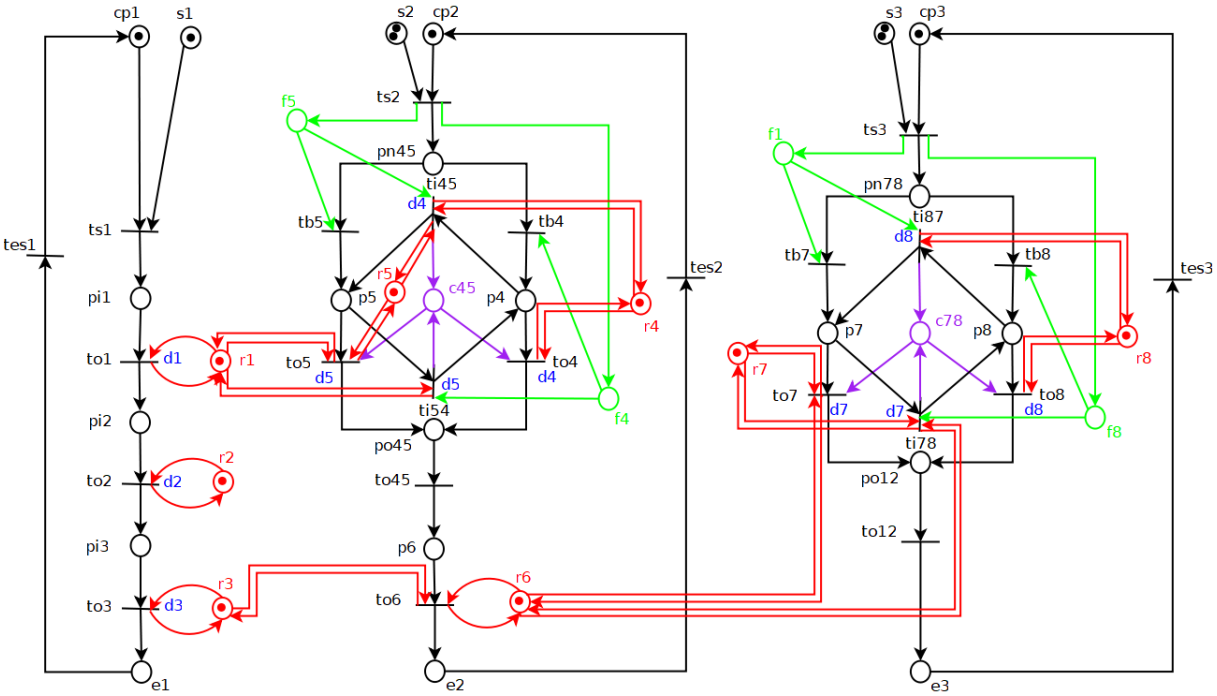


Figure 36 : FMS complexe

Dans cette partie, une modélisation d'un atelier constitué de différents types de gammes opératoires a été présentée. Cette modélisation est basée sur les fonctions de base introduites afin de créer l'organisation d'un atelier complexe. A partir d'une description synthétique du système, une technique systématique construit automatiquement les matrices d'incidence, le vecteur de marquage initial et le vecteur de paramètres temporels de l'atelier.

3.6. Conclusion

Dans ce chapitre, les PN ont été présentés pour la modélisation d'un SED. Par la suite, une modélisation par PN de différents types d'organisations usuelles d'ateliers a été introduite. En utilisant une méthode systématique de modélisation s'appuyant sur des fonctions de base, un nouveau type d'ateliers a été défini de telle sorte que les gammes opératoires respectent des contraintes de précédence partielle. La méthode proposée permet de modéliser une large varieté d'organisations d'ateliers à partir de la description synthétique du système. Suite à cette modélisation, la méthode approchée de recherche en faisceau sera utilisée pour obtenir un ordonnancement. Cette méthode basée sur l'exploration sélective de l'espace d'état du modèle PN sera détaillée dans le chapitre suivant.

Chapitre 4 : Recherche d'ordonnancement avec la recherche en faisceau

4.1.Introduction

Les PN, en association avec différentes méthodes d'optimisation, sont largement utilisés pour les problèmes d'ordonnancement des SED [Luo et al., 2015 ; Lee & DiCesare, 1994 ; Cassandras, 1993]. Afin de surmonter le problème d'explosion combinatoire liée aux méthodes exactes, où l'ensemble du graphe d'états est exploré [Lefebvre & Daoui, 2018 ; Lefebvre, 2018], une large littérature a été consacrée à la recherche de solutions approchées par des méthodes de faible complexité.

Les PN, associés aux algorithmes de recherche informés de graphes (IGS), sont très populaires pour résoudre les problèmes d'ordonnancement [Luo et al., 2015]. L'idée de ces méthodes est d'explorer sélectivement l'espace d'état du PN dans le but de trouver des chemins de coût minimal ou presque minimal depuis un ou plusieurs sommets initiaux vers un sommet cible.

Dans ce chapitre, sur la base de la stratégie de modélisation proposée dans le troisième chapitre, une nouvelle fonction coût est proposée afin de prendre en compte la modélisation itérée du système. Nous montrons que cette fonction est une borne inférieure de la durée réelle. Par la suite, une nouvelle variante de la méthode de recherche en faisceau, appelée GFBS (Generation Filtered Beam Search) [Cherif, Leclercq & Lefebvre, 2021a ; Cherif, Leclercq & Lefebvre, 2019a], est détaillée pour calculer des séquences de durée minimale ou presque minimale en explorant sélectivement l'espace d'état. L'approche doit également éviter les blocages et les branches mortes qui sont a priori inconnues pour le contrôleur.

Dans la deuxième section de ce chapitre, l'ordonnancement par recherche en faisceau est introduit. La troisième section présente la fonction coût utilisée afin d'évaluer les candidats à sélectionner. Les principales extensions de l'algorithme de recherche en faisceau (BS) sont présentées dans la section quatre. La cinquième section s'intéresse à la nouvelle variante de recherche en faisceau appelée Generation Filtered Beam Search (GFBS) et proposée dans ce travail. Dans la dernière section, la méthode GFBS est appliquée sur un exemple de la littérature et les résultats sont comparés avec ceux obtenus avec des méthodes connues.

4.2.Ordonnancement par recherche en faisceau

4.2.1 Algorithmes de recherche informés de graphes (IGS)

L'idée des IGS est de trouver une solution exacte ou approchée en explorant sélectivement l'espace d'état du modèle PN afin de faire face au problème de l'explosion combinatoire pour les systèmes de grande taille. Dans tous les algorithmes IGS, tous les sommets générés mais non développés sont

conservés sur une liste dynamique désignée génériquement comme la «frontière». Les sommets de cette liste sont appelés «sommets frontières». La manière dont les sommets frontières sont sélectionnés pour l'exploration constitue la principale différence entre tous les algorithmes IGS.

A* [Lee & DiCesare 1994 ; Ow & Morton, 1988 ; Hart, Nilsson & Raphael, 1968] est l'algorithme IGS le plus connu : cet algorithme explore les branches les plus prometteuses d'un arbre de recherche selon un critère établi avec la fonction heuristique $f(N) = g(N) + h(N)$. La figure 37 présente les coûts calculés pour un sommet N : $g(N)$ est le coût réel du chemin entre un sommet initial et le sommet N et $h(N)$ est une estimation du coût de chemin de coût minimal entre le sommet N et un sommet cible souhaité. Les sommets ayant des petites valeurs de $f(N)$ auront la priorité requise pour l'exploration.

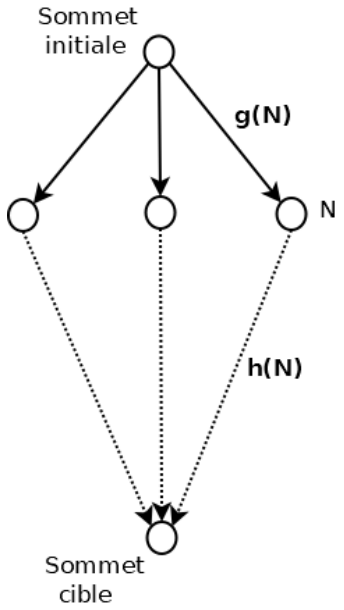


Figure 37 : Fonction heuristique f(N) = g(N) + h(N)

Les premiers articles combinant les PN et IGS ont été présentés par Lee et DiCesare (1994) qui ont introduit une version modifiée de l'algorithme A*. Dans sa version originelle, les auteurs ont tenté de forcer la recherche vers le sommet cible en attribuant plus de poids aux sommets plus profonds dans l'arbre de recherche pour éviter l'explosion de l'espace d'état. Cependant, la complexité demeurait non polynomiale [Russell & Norvig, 1995 ; Pearl, 1984], ce qui le rend uniquement approprié pour de très petites instanciations.

Plus tard, d'autres algorithmes similaires ont fourni de meilleures performances en termes de qualité et de vitesse. En général, la plupart des approches utilisent l'algorithme de base A* avec (i) des extensions qui limitent les sommets à explorer et (ii) de nouvelles fonctions heuristiques. Pour

améliorer l'exploration, les chercheurs ont développé plusieurs améliorations, par exemple, la stratégie best-first avec retour en arrière contrôlé [Lei et al., 2017 ; Lefebvre, 2016 ; Jeng & Chen, 1998 ; Huanxin Henry & MengChu, 1998], la programmation dynamique [Zhang, Freiheit & Yang, 2005], la recherche A* hybride pour assouplir le périmètre d'évaluation [Lee & Lee, 2010], l'algorithme de recherche DWS (dynamic window search) dans laquelle une politique est appliquée pour sélectionner les chemins les plus prometteurs [Luo et al., 2015 ; Reyes, Kelleher & Lloyd, 2002], l'algorithme A* avec la stratégie depth-first [Huang et al., 2009 ; Huang, Sun & Sun, 2008] qui approfondit la recherche dans le graphe d'accessibilité avec un mécanisme de contrôle, la recherche A* réactive [Kim, Suzuki & Narikiyo, 2007] qui utilise un superviseur basé sur des règles pour réduire l'espace de recherche, A* itératif avec retour en arrière [Baruwa, Piera, & Guasch, 2015 ; Baruwa & Piera, 2014], et d'autres heuristiques encore[Mejía & Niño, 2017 ; Huang, Jiang & Zhang, 2014 ; Liu et al., 2014 ; Moro, Yu & Kelleher, 2000].

Contrairement à l'algorithme A* où les candidats précédemment explorés lors de la recherche sont tous conservés, un autre IGS populaire appelé algorithme de recherche en faisceau (Beam Search (BS)) [Ow, & Morton, 1988], ne conserve que les β meilleurs candidats précédemment explorés. Cet algorithme sera détaillé dans la section suivante.

4.2.2 Algorithmes de recherche en faisceau

La recherche en faisceau [Ow, & Morton, 1988] est une méthode heuristique pour résoudre les problèmes d'optimisation. Il s'agit d'une adaptation de la méthode branch and bound [Baptiste & Le Pape, 1996 ; Le Pape, 1995], dans laquelle seuls quelques candidats sont évalués. L'algorithme de recherche en faisceau limite l'espace de recherche en développant uniquement les meilleurs candidats à chaque niveau du graphe d'états : à chaque niveau, seuls les β candidats les plus prometteurs sont conservés pour un branchement ultérieur et les candidats restants sont élagués de manière permanente. Le paramètre β est appelé «largeur de faisceau».

La figure 38 montre l'exploration d'un arbre de recherche avec l'algorithme BS définissant la largeur du faisceau à $\beta = 3$. A chaque niveau, l'algorithme sélectionne les β meilleurs candidats pour être explorés dans le niveau suivant.

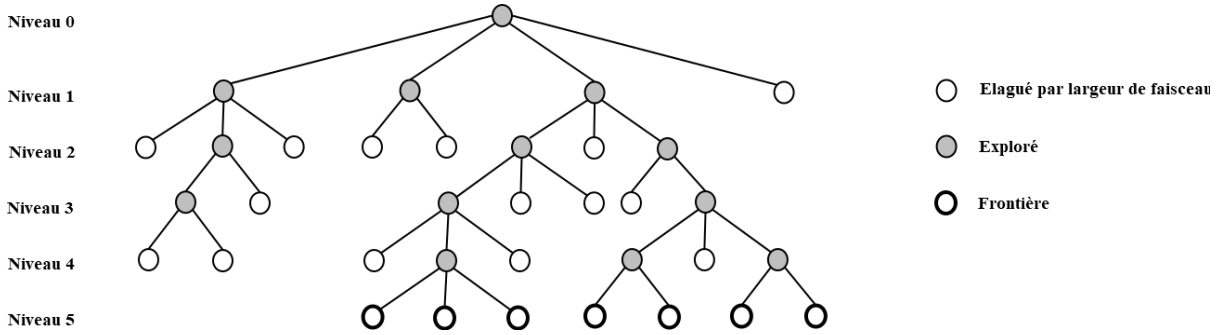


Figure 38 : Exploration de l'arbre de recherche avec l'algorithme BS et $\beta = 3$

Avec cette stratégie, la recherche progresse de manière contrôlée vers la cible. Puisqu'une grande partie de l'arbre de recherche est élaguée pour obtenir une solution, son temps d'exécution est polynomial en fonction de la taille du problème [Ow & Morton, 1988], ce qui rend cette stratégie efficace pour les grands systèmes lorsque l'espace mémoire est insuffisant pour le stockage de l'ensemble de l'arbre de recherche.

La sélection des β meilleurs candidats est effectuée selon un critère établi avec une fonction coût qui évalue le coût pour chaque candidat. Cette fonction coût permet de choisir les candidats qui doivent être conservés ainsi que ceux qui doivent disparaitre. La fonction coût est détaillée dans la section suivante.

4.3.Fonction coût

La fonction coût calcule une valeur utilisée pour sélectionner les candidats à explorer à chaque itération (niveau) de la recherche.

En termes de fonctions heuristiques, la littérature montre que des fonctions et combinaisons admissibles (une fonction qui retourne toujours une estimation inférieure au coût réel) et non admissibles ont été proposées : d'un côté, les fonctions admissibles sont généralement basées sur des caractéristiques structurelles du PN telles que la longueur de chemin minimal [Luo et al., 2015 ; Lei et al., 2014 ; Mejía & Montoya, 2009 ; Reyes, Kelleher & Lloyd, 2002], l'équations d'état [Lei et al.,2017 ; Lei et al., 2014 ; Lee & Lee, 2010 ; Jeng & Chen, 1998 ; Jeng, Chiou & Wen, 1998], la charge des ressources [Huang et al., 2008 ; Huanxin Henry & MengChu, 1998] et sur la méthode du chemin critique [Mejía et al., 2016]. D'un autre côté, les fonctions non admissibles utilisent des règles de répartition [Mejía & Odrey, 2005], sur les temps d'inactivité des ressources et les temps d'attente [Luo et al., 2015 ; Huang, Jiang & Zhang, 2014]. Des combinaisons ont également été

proposées : certains auteurs [par exemple Mejía et al., 2016] utilisent des fonctions admissibles avec des liens rompus avec des fonctions non admissibles. La littérature ne donne aucune conclusion définitive concernant les fonctions admissibles ou non et possédant les meilleures performances, le choix dépend du problème traité.

Dans cette section, une nouvelle stratégie de calcul de la fonction coût est définie afin de prendre en compte la modélisation itérée d'un FMS complexe qui consiste en des gammes opératoires hybrides dans lesquelles les opérations sont exécutées avec des contraintes de précédence partielle.

Pour chaque candidat d'une trajectoire donnée (σ, M_0), la trajectoire est divisée en deux parties autour d'un marquage intermédiaire M. La première partie présente la trajectoire déjà calculée (σ_1, M_0) de M_0 à M et la deuxième partie présente la trajectoire résiduelle (σ_2, M) avec une séquence σ_2 inconnue de M à M_{ref}. La fonction coût est formulée comme suit :

$$f(M_0, \sigma_1, M_{ref}) = g(\sigma_1, M_0) + h(M, M_{ref}) \quad (3)$$

où M est défini par $M_0 [\sigma_1 > M$.

Pour la trajectoire (σ_1, M_0) déjà exécutée, un algorithme systématique calcule la durée $g(\sigma_1, M_0)$ de (σ_1, M_0) en transformant la trajectoire non temporisée en une trajectoire temporisée sous la politique de tir au plus tôt (i.e. chaque transition t_j dans σ_1 est tirée dès que sa contrainte de temps $D(t_j)$ est satisfaite). L'algorithme, détaillé dans [Lefebvre, 2017a ; Lefebvre, 2017b] met à jour à chaque nouveau marquage M les durées restantes des transitions activées pour être tirés.

L'estimation du temps résiduel pour atteindre la référence $h(M, M_{ref})$ est simplement la somme des durées des opérations non effectuées sans prendre en compte la disponibilité des ressources. Pour estimer le temps résiduel avec la fonction $h(M,M_{ref})$, un modèle PN réduit $N_r = <P_r, T, Wr_{PR}, Wr_{PO}, D>$ est introduit où $P_r = P \setminus \{\boldsymbol{R}, CP\}$ est un ensemble de m_r places où les places ressources et capacités sont supprimées et T est l'ensemble des q transitions. $Wr_{PO} \in (\mathbf{N})^{mr \times q}$ et $Wr_{PR} \in (\mathbf{N})^{mr \times q}$ sont les matrices post et pré-incidence réduites, et $Wr = Wr_{PO} - Wr_{PR} \in (\mathbf{Z})^{mr \times q}$ est la matrice d'incidence réduite. Le marquage du PN réduit est noté M_r.

La stratégie de calcul de la fonction coût est améliorée et adaptée pour traiter la modélisation de la nouvelle organisation du FMS. L'estimation $h(M, M_{ref})$ est définie en fonction du type de la gamme opératoire.

4.3.1 Fonction estimation pour une séquence d'opérations

Pour une séquence d'opérations *J* (par exemple, une gamme opératoire *J* dans un problème de job shop), plusieurs fonctions d'estimation $h_J(M,M_{ref})$ de la durée résiduelle ont été proposées : dans [Lei et al., 2014], h_J s'appuie sur les équations d'états de PN ; dans [Luo et al., 2015], h_J s'appuie sur la recherche de ressources et de places opérationnelles ; dans [Lefebvre, 2017a], h_J est basée sur le vecteur de tir ; dans [Baruwa & Piera, 2016 ; Xiong & Zhou, 1998], h_J s'appuie sur la durée résiduelle des machines.

Dans ce travail, nous proposons une fonction d'estimation h_J basée sur la recherche des chemins les plus courts depuis les places marquées vers une place référence p_{ref} qui est ajoutée au modèle global du FMS [Lefebvre & Mejía, 2018 ; Mejía & Nino, 2017 ; Luo et al., 2015].

Pour un ensemble d'opérations avec contraintes de précédence totale, l'approximation $h_J(M, M_{ref})$ du coût de la partie inconnue de la trajectoire est donnée par (4) :

$$h_J(M, M_{ref}) = M_r(s_J) \times D(J) + \max\{\forall\, p_i \in P(M_r)\setminus\{s_J\},\ \chi*(p_i, e_J) - EFT(to_i) \text{ tel que } p_i \in PI_J \text{ and } to_i \in (p_i)\bullet\} \quad (4)$$

où $M_r = Réduire(M)$ est le marquage réduit obtenu à partir de *M* en supprimant les places de capacité et de ressources, s_J est la place de départ unique de la gamme opératoire *J*, e_J est la place de fin unique de la gamme opératoire *J* et $EFT(to_i)$ est un terme de correction qui correspond à la durée d'activation pour le tir de la première transition to_i du chemin de coût minimal entre p_i et e_J.

En termes simples, $h_J(M, M_{ref})$ correspond à la somme des durées d'opérations qui ne sont pas encore effectuées, corrigées selon l'opération en cours. La première partie de l'équation (4) se réfère au produit du temps nécessaire $D(J)$ pour effectuer la gamme opératoire par le nombre de jetons dans la place de départ $M_r(s_J)$ (qui doivent attendre la fin de l'exécution en cours de la gamme opératoire) ; la deuxième partie se réfère au coût maximum des chemins les plus courts de chaque place marquée p_i à la place de référence e_J. Tant que la capacité de la séquence d'opérations est égale à 1, il existe au plus une seule place $p_i \in PI_J$ telle que $M(p_i) = 1$. Par conséquent l'équation (4) peut être reformulée avec l'équation (5) :

$$h_J(M, M_{ref}) = M_r(s_J) \times D(J) + \chi^*(p_i, e_J) - EFT(to_i) \text{ tel que } to_i \in (p_i)\bullet \quad (5)$$

4.3.2 Fonction estimation pour un ensemble d'opérations avec une flexibilité totale

Utilisant un modèle S2OPR, la durée résiduelle pour une gamme opératoire composée d'un ensemble d'opérations avec une flexibilité totale, ne peut plus être calculée en fonction des chemins du modèle PN, mais dépend des durées des opérations. Une telle durée est obtenue comme la somme des durées d'opérations non encore exécutées. Une fonction heuristique est proposée dans [Mejia et al., 2017] afin de prendre en compte le modélisation S2OPR pour un ensemble d'opérations avec une flexibilité totale : cette fonction heuristique est basée sur les durées résiduelles de traitement des opérations restantes et des siphons critiques.

Dans cette section, nous proposons une fonction, inspirée de [Mejia et al., 2017], afin de calculer le temps résiduel restant pour une gamme opératoire composée d'un ensemble d'opérations avec une flexibilité totale. La fonction d'estimation $h_J(M, M_{ref})$ de la durée résiduelle pour une gamme opératoire composée d'un ensemble d'opérations avec une flexibilité totale est obtenue comme la somme corrigée des durées d'opérations non encore exécutées. Le statut de chaque opération o_i est défini par une place d'opération p_i et une place drapeau f_i.

Trois cas peuvent se produire : (i) la place drapeau f_i est marquée et la place de l'opération p_i n'est pas marquée, alors l'opération n'est pas encore effectuée ; (ii) la place d'opération p_i est marquée et la place drapeau f_i n'est pas marquée, alors l'opération est en cours d'exécution ; (iii) la place drapeau f_i et la place d'opération p_i sont toutes les deux non marquées, alors l'opération o_i est déjà exécutée.

La place drapeau et la place d'opération correspondant à une opération donnée ne peuvent pas être marquées ensemble (au contraire, les deux places peuvent être vides ensemble si l'opération a déjà été exécutée). Ainsi, l'approximation $h_J(M, M_{ref})$ de la durée de σ_2 est donnée par la somme des durées d'opérations non encore exécutées, et est égale au produit du marquage de la place drapeau $M(f_i)$ par la durée d'opération $D(to_i)$ à laquelle on ajoute le produit du marquage de la place d'opération $M(p_i)$ par la durée d'opération $D(to_i)$. Un terme de correction $EFT(to_i)$, correspond au temps d'activation pour le tir de la transition to_i, est ajouté pour l'opération en cours d'exécution. Le temps résiduel jusqu'à M_{ref} est estimé par l'équation (6) :

$$h_J(M, M_{ref}) = M(s_J) \times D(J) + \sum_{f_i \in \mathrm{PF}} (M(f_i) \times D(to_i) + M(p_i) \times D(to_i)) - EFT(to_i) \text{ tel que } to_i \in (p_i)\bullet \quad (6)$$

où $PF = \{f_i, o_i \in O_J\}$ est l'ensemble des places drapeau.

4.3.3 Fonction estimation pour une gamme opératoire hybride

Pour calculer la durée résiduelle d'une gamme opératoire hybride, nous utilisons l'organisation itérée présentée dans le chapitre précédent avec un ensemble de gammes opératoires de base et nous reformulons l'ensemble du modèle comme un PN en couches avec différents niveaux. L'idée, que nous proposons ici [Cherif, Leclercq & Lefebvre, 2021a ; Cherif, Leclercq & Lefebvre, 2019a], est d'introduire des transitions dynamiques (qui seront représentées par des rectangles dans les figures suivantes) et des places dynamiques (qui seront représentées par des doubles cercles dans les figures suivantes). Les transitions dynamiques ont des temps de déclenchement variables afin de propager les temps résiduels d'une couche à la suivante. Les places dynamiques reportent le marquage complet d'une gamme opératoire de base donnée d'une couche à une autre. L'utilisation combinée des transitions et places dynamiques intègre les modèles des niveaux 1,…, l-1 dans les transitions et places dynamiques du niveau l : le marquage et la durée résiduelle de chaque gamme opératoire de base au niveau 1,…, l- 1 sont propagés au niveau l pour changer respectivement le marquage des places dynamiques et le temps de tir des transitions dynamiques. Les autres places et transitions du modèle seront appelés places et transitions statiques.

Soit $J_l = \{J_{1l},..,J_{kl}\}$ l'ensemble des gammes opératoires de base au niveau l et$\{1,..,L\}$ l'ensemble des niveaux. Pour tout marquage donné M, l'estimation $h_J(M,M_{ref})$ est calculée comme suit :

Au niveau 1 : toutes les places et transitions sont statiques. Le marquage des gammes opératoires de base est une simple copie du marquage du PN et la durée de tir $D(to_i)$ d'une transition temporisée est égale à la durée d'exécution de l'opération correspondante o_i.

Au niveau l> 1 : le marquage des places statiques est une copie du marquage du modèle PN. La durée de tir $D(to_i)$ d'une transition statique est égale à la durée d'exécution de l'opération correspondante o_i. Au contraire, le marquage des places dynamiques reporte le nombre de jetons dans la gamme opératoire de base $J_{k'l'}$ au niveau $l' < l$, et la durée de tir $D(to_i)$ d'une transition dynamique est égale à la durée résiduelle $h_{Jk'l'}(M, Mref)$ de la gamme opératoire de base $J_{k'l'}$ au niveau $l'<l$. La gamme opératoire de base $J_{k'l'}$ est définie par l'organisation du FMS.

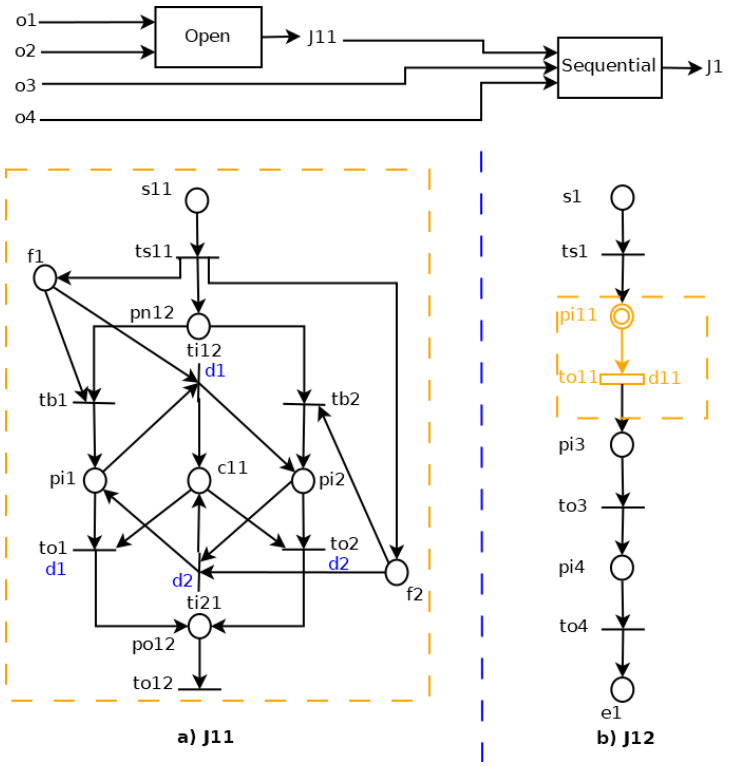


Figure 39 : Propagation du marquage et de la durée résiduelle pour une gamme opératoire hybride

Exemple : Prenons l'exemple de la gamme opératoire hybride J_1 illustrée à la figure 39 avec deux gammes opératoires de base $\{J_{11}, J_{12}\}$ où J_{11} est la seule gamme de base du premier niveau et J_{21} appartient au deuxième et dernier niveau. En commençant par le premier niveau et la gamme opératoire de base J_{11} (composée de deux opérations statiques avec une flexibilité totale $\{o_1, o_2\}$), l'estimation de la durée résiduelle est évaluée à l'aide de l'équation (6). Puis le paramètre de temps d_{11} de la transitions dynamique to_{11} de la gamme opératoire de base J_{12} au niveau 2 est mis à jour. Le marquage est également mis à jour pour la place dynamique pi_{11} de la gamme opératoire de base J_{12}. Par la suite, l'estimation de la durée résiduelle pour la gamme opératoire J_{12} du dernier niveau 2 est évaluée à l'aide de l'équation (5). La durée résiduelle de la gamme opératoire hybride est finalement obtenue $h_J(M, M_{ref}) = h_{J12}(M, M_{ref})$.

4.3.4 Fonction estimation d'un FMS complexe

La durée résiduelle pour le FMS complexe est obtenue et correspond à la valeur maximale des durées résiduelles sur toutes les gammes opératoires hybrides du FMS :

$$h(M, M_{ref}) = max\{h_J(M, M_{ref}) \; tel \; que \; J \in \boldsymbol{J}\} \qquad (7)$$

4.3.5 Fonction estimation : une borne inférieure de la durée réelle

Afin de garantir que l'algorithme peut trouver une solution optimale, $h(M, M_{ref})$ doit être admissible [Nilsson, 1980] pour tout marquage atteignable M, c'est-à-dire $h(M, M_{ref}) \leq h^*(M, M_{ref})$, où $h^*(M, M_{ref})$ est le temps minimal réel entre M et M_{ref} selon la politique de tir au plus tôt.

Proposition : Considérons un FMS hybride. L'estimation $h(M, M_{ref})$ calculée avec (7) est une borne inférieure de la durée réelle de la trajectoire résiduelle (σ_2, M) de M à M_{ref}.

Démonstration : Selon l'organisation itérée, précédemment introduite, chaque gamme opératoire du FMS est décrite avec une ou plusieurs gammes opératoires de base J_{kl} organisées en L niveaux et un ensemble de ressources qui sont partagées par les opérations. J_{kl} est soit une séquence d'opérations soit un ensemble d'opérations avec une flexibilité de totale.

Au niveau 1, pour une séquence d'opérations J_{k1} de capacité 1 et sans aucune ressource, l'estimation de la durée résiduelle est donnée par (5). Cette estimation est la durée réelle dans le cas où les ressources ne sont pas prises en compte. Lorsque les ressources sont prises en compte, cette durée augmente si certaines ressources deviennent indisponibles. Ainsi, la durée réelle est au moins égale à (5). Le raisonnement est similaire pour un ensemble d'opérations J_{k1} avec une flexibilité totale. La durée réelle est au moins égale à (6) car un temps d'attente supplémentaire peut être ajouté lorsque les ressources deviennent indisponibles.

Au niveau $l > 1$, le raisonnement est le même mais des séquences de gammes opératoires doivent être considérées au lieu de séquences d'opérations (resp. des ensembles de gammes opératoires avec une flexibilité totale au lieu d'ensembles d'opérations). À partir du marquage M du modèle T-TPN et du calcul de l'estimation $h_{k'l'}(M, M_{ref})$ de la durée résiduelle pour les gammes opératoires de base $J_{k'l'}$ à $l' < l$, les marquages des places dynamiques et les durées de tir des transitions dynamiques de la gamme opératoire de base J_{kl} sont d'abord miss à jour puis l'estimation $h_{kl}(M, M_{ref})$ de la durée résiduelle de la gamme opératoire de base J_{kl} est calculée. Cette estimation est égale à la durée réelle lorsque les ressources ne sont pas prises en compte. L'ajout de ressources partagées augmente la durée réelle mais ne change pas la valeur de $h_{kl}(M, M_{ref})$. La propagation des estimations $h_{kl}(M, M_{ref})$ jusqu'au niveau L conduit à l'estimation $h_{JL}(M, M_{ref})$ pour la gamme opératoire J qui est une borne inférieure de la durée réelle requise pour exécuter la gamme opératoire J. Finalement, (7) est une borne inférieure de la durée réelle requise pour exécuter tous les gammes opératoires dans le FMS.

4.4.Variantes de l'algorithme de recherche en faisceau

L'algorithme de recherche en faisceau a été efficace sur un certain nombre de problèmes combinatoires et sa complexité de temps d'exécution est polynomiale [Ow & Morton, 1988]. Le principal inconvénient de la recherche en faisceau est la sélection des sommets qui peut conduire à l'élimination de bons candidats dans l'arbre de recherche. L'avantage est un coût raisonnable afin de pouvoir résoudre des problèmes d'ordonnancement pour des systèmes de grande taille que les méthodes exactes n'arrivent pas à résoudre. Les extensions de l'algorithme BS sont (i) la recherche en faisceau filtrée [Ow & Morton, 1988] qui pré-évalue et filtre les sommets avec une fonction plus simple avant d'appliquer une fonction d'évaluation plus complexe aux sommets restants et (ii) la recherche en faisceau avec récupération (RBS) appliquant à chaque sommet un autre test qui vérifie si le sommet actuel est dominé par un autre sommet [Croce, Ghirardi & Tadei, 2004].

La combinaison de l'algorithme BS avec les PN a été étudiée dans [Lefebvre & Mejía, 2018 ; Mejía et al. 2016 ; Mejía & Montoya, 2009 ; Mejía & Odrey, 2005]. Plusieurs extensions ont été présentées afin d'améliorer principalement la sélection des sommets à explorer et la vitesse de la recherche. Ces extensions combinent la stratégie d'exploration de l'algorithme A* avec la méthode d'élagage de recherche en faisceau.

La première extension, appelée Beam A* search (BAS), a été introduite par Mejía & Odrey (2005). La stratégie consiste à explorer un nombre maximum de marquages à chaque niveau du graphe d'états. Ainsi, à une itération donnée, seuls les marquages dont le niveau n'a pas été entièrement développé restent sur une liste notée OPEN : tous les autres marquages sont supprimés. Un inconvénient de l'algorithme BAS est que, au fur et à mesure que la recherche progresse, la plupart des marquages explorés proviennent d'un marquage parent commun aux premières itérations de la recherche. A ce titre, la recherche explore uniquement les marquages d'une région limitée de l'espace d'état [Mejía & Odrey, 2005].

Afin de surmonter les limites de BAS, un autre algorithme appelé Filtered Beam Search (FBS) [Lefebvre & Mejía, 2018 ; Mejía & Niño, 2017 ; Ow & Morton, 1988] a été introduit. L'idée est d'explorer des marquages dans différentes branches afin de surmonter le problème d'exploration d'une région limitée de l'espace d'état avec BAS. Deux paramètres ont été introduits : un filtre global β_g qui définit le nombre maximum de marquages pouvant être développés à n'importe quel niveau de l'arbre de recherche et un filtre local β_l qui établit le nombre maximum de marquages successeurs qui atteignent la frontière à partir d'un sommet parent donné. Le choix des candidats à explorer en utilisant l'algorithme FBS présente la limite principale de la méthode. En effet, la sélection du candidat à

explorer se fait entre candidats de différents niveaux ce qui implique qui n'ont pas les mêmes chances pour être sélectionnés [Lefebvre & Mejía, 2018 ; Mejía & Niño, 2017]. Cela peut aboutir à une recherche en profondeur avec le risque de choix médiocre ce qui implique un retard de la recherche comme le montre l'exemple de la figure 40.

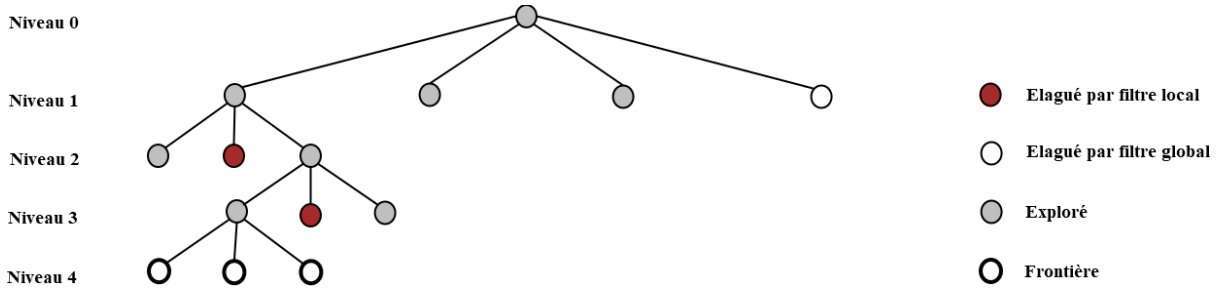


Figure 40 : Exploration de l'arbre de recherche avec l'algorithme FBS, $\beta_g = 3$ et $\beta_l = 2$

Afin de surmonter les limites de BAS et FBS, un autre algorithme appelé Hybrid Filtered Beam Search (HFBS) a été introduit [Mejía & Niño, 2017]. Cet algorithme combine les principes des algorithmes Beam Search [Sabuncuoglu & Bayiz, 1999], Filtered Beam Search (FBS) [Ow & Morton, 1988] et Beam A* Search [Mejía & Montoya, 2009]. L'idée est d'explorer des marquages issus de différentes branches à des niveaux peu profonds et des marquages issus seulement des meilleures branches sélectionnées à des niveaux plus profonds. L'algorithme utilise deux paramètres : un filtre global β_g et un filtre local β_l. L'algorithme de recherche explore les différentes branches de même profondeur aux premiers niveaux. Par contre, aux niveaux plus profonds, la recherche sera intensifiée sur les meilleures branches. L'algorithme HFBS a deux limites potentielles : premièrement, il n'est pas garanti de trouver une solution même s'il en existe une et deuxièmement, la sélection de la meilleure combinaison des paramètres β_g et β_l n'est pas facile à obtenir. En particulier il n'y a pas de propriété de monotonie des performances en fonction de β_g et β_l.

Même si la recherche explore, au début, des candidats de différentes branches, elle ne conserve pas cet équilibre pour les stades ultérieurs de la recherche : la recherche est intensifiée sur les meilleurs branches ce qui peut favoriser la recherche en profondeur avant de découvrir qu'on a fait le mauvais choix comme le montre l'exemple de la figure 41.

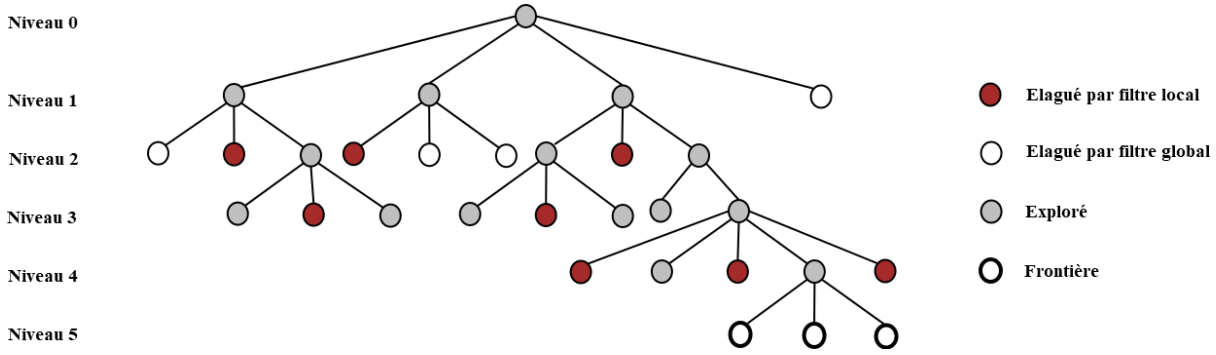


Figure 41 : Exploration de l'arbre de recherche avec l'algorithme HFBS, $\beta_g = 3$ et $\beta_l = 2$

Dans le but de surmonter ces limites, une nouvelle stratégie, appelée Iterated Hybrid Filtered Beam Search (IHFBS) [Mejía & Niño, 2017], a été proposée. Cette stratégie consiste à répéter l'algorithme HFBS, en commençant à chaque fois par une combinaison différente de β_g et β_l. Ceci est parfaitement raisonnable si le temps de calcul d'une seule exécution est court. Cette idée est inspirée de la métaheuristique Iterated Local Search (ILS) [Lourenço, Martin & Stützle, 2003] qui peut déclencher une recherche locale dans l'algorithme à tout moment [Baruwa et al., 2015]. Le nouvel algorithme utilise un mécanisme d'apprentissage utilisant la meilleure valeur de la fonction objectif comme borne supérieure (*UB*). Ainsi, chaque fois que l'algorithme trouve une meilleure solution, la borne supérieure est mise à jour. Dans la prochaine exécution, seuls les marquages avec des valeurs de $f(M) \leq UB$ sont conservés. L'utilisation de cette stratégie améliore la performance mais elle est très consommatrice en temps et en mémoire à cause de l'appel répété de l'algorithme de recherche.

Chacune des différentes méthodes proposées ont des points forts et des inconvénients. Afin de surmonter les limites et de bien exploiter les avantages des différentes méthodes, une nouvelle version de recherche en faisceau est proposée. Cet algorithme, détaillé dans la section suivante, est appelé Generation Filtered Beam Search (GFBS) [Cherif, Leclercq & Lefebvre, 2021a ; Cherif, Leclercq & Lefebvre, 2019a].

4.5.Algorithme de recherche Generation Filtered Beam Search (GFBS)

Dans cette section, un algorithme de recherche en faisceau modifié, basé sur la notion de génération, est proposé afin de donner une chance égale à tous les candidats issus de la même génération d'être sélectionnés et d'avoir ainsi un équilibre entre la largeur et la profondeur de la recherche. En utilisant FBS et HFBS, la sélection d'une population est faite à partir de candidats qui n'appartiennent pas à la même génération. Ce que nous appelons une génération est la population composée de tous les successeurs directs (candidats issus de tous les parents) d'une population donnée de parents. Avec

FBS, la comparaison est faite entre les successeurs d'un parent et tous les autres parents. Puisque la sélection des candidats est basée sur la fonction objectif composée du coût de la trajectoire déjà calculée et de l'estimation de la trajectoire résiduelle qui est une borne inférieure du coût réel, les parents ont plus de chance d'être sélectionnés. Avec GFBS, nous donnons des chances égales aux successeurs (puisque ceux-ci ne sont plus comparés avec les parents).

Pour chaque sommet parent, l'algorithme commence par explorer leurs successeurs, puis les trie en fonction du coût total f pour placer les β_l meilleurs candidats dans une liste temporaire. Une fois l'exploration de tous les sommets parents terminée, l'algorithme sélectionne les β_g meilleurs candidats dans la liste temporaire pour créer une nouvelle génération. L'algorithme résultant est itéré sur les générations successives. La caractéristique principale de l'algorithme proposé est que le tri global est effectué après expansion de tous les sommets parents ce qui n'est pas le cas pour le FBS habituel où le tri global est effectué après l'expansion du meilleur parent uniquement. Nous désignons cette variante de l'algorithme FBS sous le nom de Generation Filtered Beam Search (GFBS) [Cherif, Leclercq & Lefebvre, 2021a ; Cherif, Leclercq & Lefebvre, 2019a].

L'objectif est d'explorer sélectivement l'espace d'état du PN en fonction des générations successives de candidats afin d'atteindre le marquage de référence M_{ref} à partir d'un marquage initial M_0 avec une trajectoire de marquage minimisant la durée de l'exécution des gammes opératoires.

La figure 42 montre l'exploration d'un arbre de recherche avec l'algorithme GFBS pour $\beta_l = 2$ et $\beta_g = 3$. Pour chaque parent, l'algorithme sélectionne les β_l meilleurs candidats. A chaque niveau, l'algorithme sélectionne les β_g meilleurs candidats devant être explorés au niveau suivant. L'algorithme est basé sur un équilibre entre la profondeur et la largeur de la recherche avec une diversification des candidats sélectionnés pour ne pas explorer uniquement des candidats d'une région limitée de l'espace d'état.

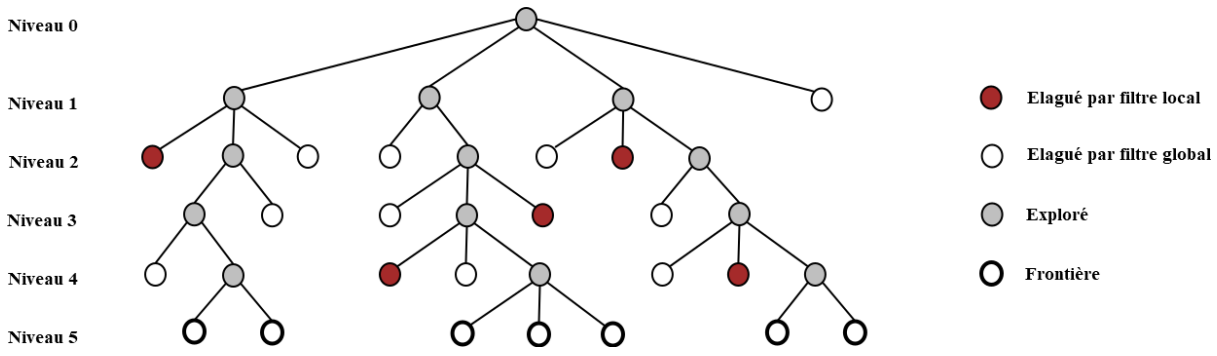


Figure 42 : Exploration de l'arbre de recherche avec l'algorithme GFBS, $\beta_g = 3$ et $\beta_l = 2$

Nous présentons ci-dessous le pseudo-code de l'algorithme GFBS. L'algorithme renvoie une séquence valide de transition *Seq* ainsi que son coût. L'algorithme utilise une liste, appelée OPEN, pour stocker les candidats d'une génération donnée qui doivent être développés. La fonction coût *f* détaillée à la section 4.3 est utilisée pour trier les candidats d'une génération à des fins de sélection. La liste OPEN est mise à jour après chaque itération (une itération correspond à l'exploration de tous les candidats d'une génération donnée).

Algorithme : GFBS Entrées : N, M_0, β_g, β_l, M_{ref} Sorties : *Seq*, C_{max}

1. Générer les successeurs de M_0 et placer les β_g meilleurs candidats dans OPEN.
2. Si OPEN est vide, sortir et retourner *Seq* = ∅ and $C_{max} = \infty$.
2. Sinon, pour chaque candidat *M* dans *OPEN* faire
   a. Générer les successeurs de *M*.
   b. Pour chaque successeur *M'* issu de *M*.
      i. Si le successeur *M'* est égale à M_{ref}, construire la séquence de transitions *Seq* de M_0 à M_{ref} et renvoyer *Seq* et C_{max} comme la durée de *Seq*.
      ii. Sinon, calculer la fonction de coût $f(M_0, M', M_{ref})$ et placer *M'* dans une liste temporaire TEMPLIST.
   c. Sélectionner jusqu'au β_l meilleurs candidats de TEMPLIST et les placer dans une liste temporaire GENERATION en respectant :
      i. Si *M'* est égal à un marquage dans GENERATION, vérifier si un nouveau meilleur chemin a été trouvé, mettre à jour le chemin de M_0 à *M* et le coût $f(M_0, M', M_{ref})$.
      ii. Sinon, placer *M'* dans GENERATION.
   d. Vider TEMPLIST.
3. Vider OPEN.
4. Sélectionner jusqu'aux β_g meilleurs candidats de GENERATION et les placer dans OPEN.
5. Vider GENERATION.
6. Retourner à l'étape 2.

Un avantage de l'algorithme proposé est d'éviter systématiquement les blocages qui sont a priori inconnus pour le contrôleur. En effet, un coût infini est attribué pour les candidats qui ne peuvent pas atteindre le marquage référence M_{ref} (c'est le cas des blocages). Ainsi, ces candidats ne sont pas sélectionnés pour la prochaine génération car ils possèdent un coût plus élevé par rapport aux autres candidats.

Exemple :

Un exemple, composé de deux gammes opératoires $\{J_1, J_2\}$ et trois ressources $\{r_1, r_2, r_3\}$, est présenté sur la figure 43. Chaque gamme opératoire est composée d'une séquence de deux opérations : J_1 est la séquence des opérations $\{o_{11}, o_{12}\}$ et J_2 est la séquence des opérations $\{o_{21}, o_{22}\}$. La ressource r_1 est partagée entre les deux gammes opératoires et plus précisément entre les deux opérations o_{11} et o_{12}.

Pour chaque opération une durée d_{Ji} est définie : $d_{11} = 14$; $d_{12} = 15$; $d_{11} = 13$; $d_{11} = 17$.

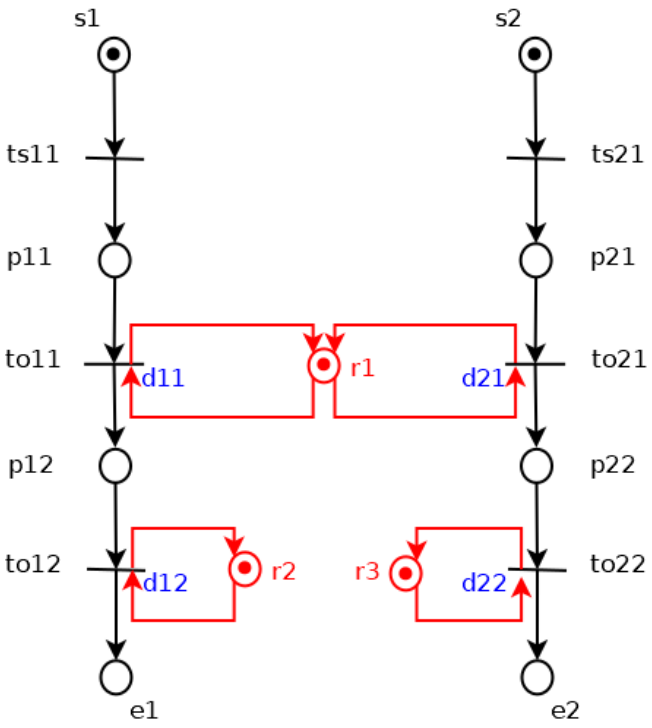


Figure 43 : Exemple d'un FMS

Pour cet exemple, les différentes générations sont présentées. En utilisant les paramètres $\beta_g = 3$ et un $\beta_l = 2$, les résultats obtenus sont illustrés dans la figure 44.

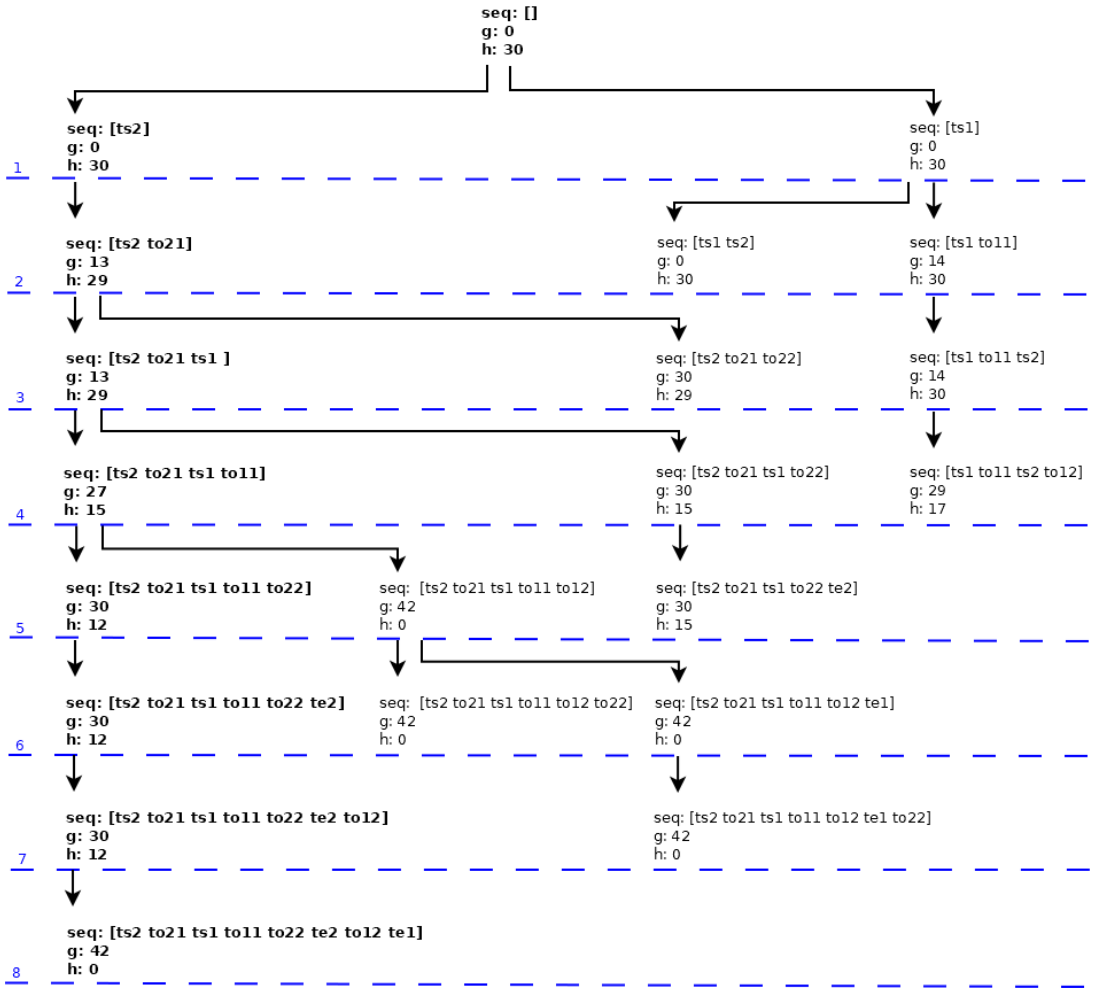


Figure 44 : Détails de l'exploration du FMS par l'algorithme GFBS, $\beta_g = 3$ et $\beta_l = 2$

Les détails de l'exploration sont donnés sur la figure 44 : les candidats sélectionnés à chaque itération (la nouvelle génération) sont représentés ainsi que la séquence déjà calculée (*seq*), la durée de la séquence déjà calculée (g) et l'estimation du coût résiduel (h). Les candidats dans chaque population sont générés avec des séquences de tirs qui ont un nombre égal de transitions et la sélection est faite sur la meilleure valeur de la fonction coût $g + h$. Tous les candidats d'une génération donnée ont des marquages différents (si deux candidats ont le même marquage, seul celui avec le coût minimal est sélectionné pour l'exploration). L'idée est d'améliorer la diversification en donnant la possibilité à d'autres candidats, avec des marquages différents, d'être sélectionnés. Afin d'atteindre la référence et d'obtenir le résultat optimal, 8 itérations successives ont été calculées comme le montre la figure 44. L'ordonnancement obtenu est optimal, cela a été validé en appliquant une méthode globale [Lefebvre & Daoui, 2018 ; Lefebvre, 2018].

4.6.Application

Cette section est consacrée à illustrer l'utilisation de l'algorithme GFBS. L'algorithme a été programmé sur un ordinateur 1,8 GHz avec 16 Go de mémoire RAM avec le logiciel Matlab. L'exemple présenté dans [Han et al., 2015] et utilisé dans [Mejia & Nino, 2017], se compose de 20 instanciations. Le FMS est composé de 2 gammes opératoires et 4 ressources. L'exemple est à l'origine modélisé comme un P-TPN et nous générons son T-TPN équivalent en remplaçant chaque place temporisée par deux places et une transition temporisée [Sifakis, 1979] comme le montre la figure 45.

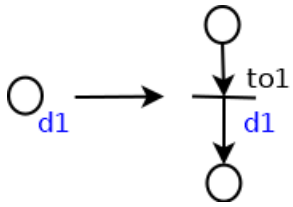
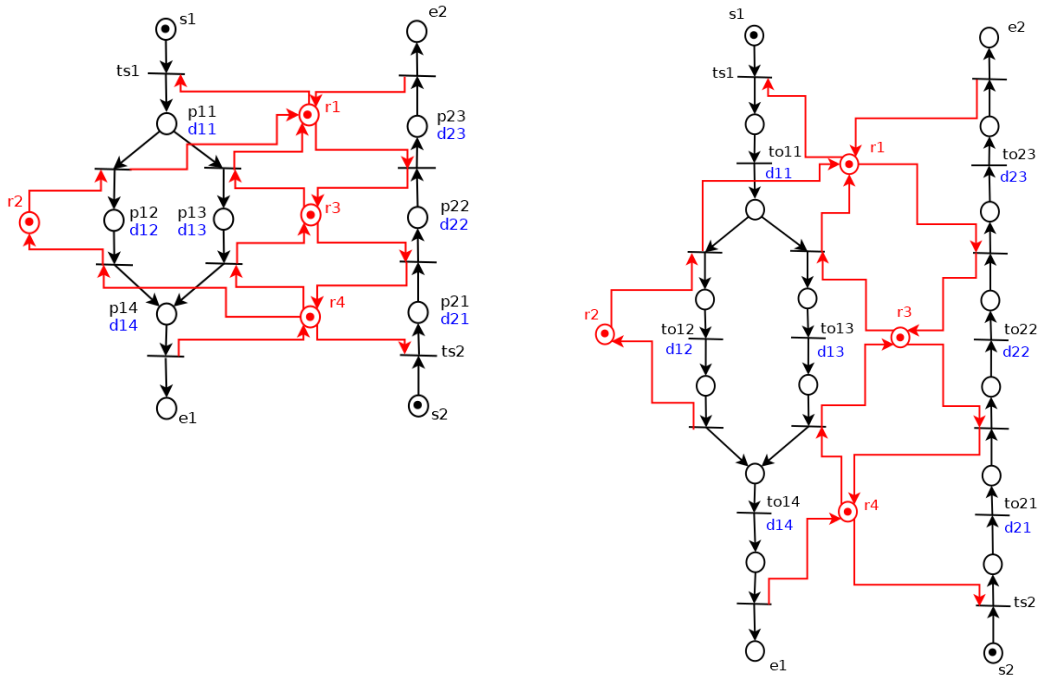


Figure 45 : Passage de P-TPN vers T-TPN

Le P-TPN (figure 46.a) et son équivalent T-TPN (figure 46.b) sont présentés sur la figure 46. Notez que la taille du T-TPN généré avec 22 places et 17 transitions est plus grande que son équivalent P-TPN avec 15 places et 10 transitions. Par conséquent, la taille du graphe d'accessibilité du modèle T-TPN est également plus grande que la taille du graphe d'accessibilité du modèle P-TPN. Cela pénalise notre méthode par rapport aux autres méthodes basées sur des modèles P-TPN. Les temps de traitement des opérations sont d_{11} = 25, d_{12} = 23, d_{13} = 27, d_{14} = 25, d_{21} = 25, d_{22} = 25 et d_{23} = 25. 20 instanciations, avec un nombre de ressources et de capacités d'exécution des gammes opératoires différentes, sont prises en compte.



a- Modèle P-TPN b- Modèle T-TPN

Figure 46 : Exemple d'un modèle FMS avec $M(s_J) = 1$ et $M(r_i) = 1$

Tableau 1 : Instanciations de FMS

Instanciations	$M(s_j)$	$M(r_i)$	HPSO	CPU(s)	IHFBS	CPU(s)	GFBS	CPU(s)
FMS1	5	1	**293**	500	**293**	40	**293**	**1**
FMS2	10	1	**557**	1000	**557**	80	**557**	**3**
FMS3	20	1	**1087**	2000	**1087**	160	**1087**	**10**
FMS4	30	1	**1617**	3000	**1617**	240	**1617**	**19**
FMS5	50	1	**2677**	5000	**2677**	800	**2677**	**31**
FMS6	5	2	**150**	500	**150**	80	**150**	**6**
FMS7	10	2	**273**	1000	**273**	160	**273**	**18**
FMS8	20	2	547	2000	**531**	320	**531**	**26**
FMS9	30	2	833	3000	**795**	480	**795**	**39**
FMS10	50	2	1461	5000	**1325**	800	**1325**	**53**
FMS11	5	3	**106**	500	**106**	120	**106**	**17**
FMS12	10	3	**185**	1000	**185**	240	**185**	**32**
FMS13	20	3	371	2000	**368**	480	**368**	**40**
FMS14	30	3	591	3000	**532**	720	534	**53**
FMS15	50	3	1092	5000	**897**	1200	921	**69**
FMS16	5	4	**99**	500	**99**	160	**99**	**22**
FMS17	10	4	150	1000	**149**	320	**149**	**39**
FMS18	20	4	287	2000	**274**	640	**274**	**51**
FMS19	30	4	445	3000	**401**	960	421	**62**
FMS20	50	4	877	5000	**664**	1600	687	**80**

Les instanciations sont désignées par FMS01-FMS20 comme indiqué dans le tableau 1. L'algorithme GFBS est exécuté sur ces instanciations avec différentes valeurs des paramètres β_g dans l'intervalle [20 : 100] et différentes valeurs des paramètres β_l dans l'intervalle [2 : 10] (comme cela est fait dans IHFBS). Nous avons utilisé trois couples de paramètres selon le nombre d'exécutions des gammes opératoires et le nombre de ressources. Pour les FMS avec un petit nombre d'exécution et/ou un nombre limité de ressources (FMS1, FMS2, FMS3, FMS4, FMS5, FMS6, FMS7), nous avons utilisé β_g =20 et β_l =2. Pour FMS8, FMS9, FMS11, FMS12 et FMS16, nous avons utilisé β_g =50 et β_l =10. Enfin, pour les instanciations restantes, avec un grand nombre d'exécutions et/ou de multiples ressources (FMS10, FMS13, FMS14, FMS15, FMS17, FMS18, FMS19 et FMS20), nous avons utilisé β_g =100 et un β_l =20. Les meilleurs résultats obtenus sont donnés dans le tableau 1. L'idée est d'augmenter le nombre de candidats explorés au fur et à mesure de l'augmentation de la taille de l'espace d'état de l'instanciation mais en maintenant une complexité raisonnable en mémoire et donc en temps de calcul (CPU).

Les résultats sont comparés à ceux obtenus avec l'algorithme HPSO (Hybrid Particle Swarm Optimization algorithm) utilisé dans [Han et al., 2015] et à ceux obtenus avec l'algorithme IHFBS (Iterated Hybrid Filtered Beam Search) utilisé dans [Mejia & Nino, 2017]. L'algorithme HPSO utilise les contrôleurs de blocage proposé par [Piroddi, Cordone & Fumagalli, 2008]. L'algorithme IHFBS est basé sur l'appel répété de l'algorithme de recherche où la valeur de la fonction objectif de l'appel en cours est utilisée comme borne supérieure pour l'appel suivant.

Nous avons rapporté les temps de calcul CPU mentionnés dans la littérature mais il convient de noter que les processeurs utilisés étaient différents de notre processeur : GFBS fonctionne sur ordinateur 1,8 GHz, HPSO fonctionne sur un ordinateur 2,6 GHz [Han et al., 2015] et IHFBS fonctionne sur un Ordinateur 3,2 GHz [Mejia & Nino, 2017]. Compte tenu des vitesses d'horloge et des différents types de configurations des ordinateurs, des conclusions définitives sur la comparaison de la complexité temporelle sont difficiles à tirer. La comparaison avec l'algorithme HPSO peut être faite du point de vue mémoire et temps de calcul : en termes de mémoire, l'espace de recherche du modèle T-TPN utilisé avec l'algorithme GFBS est beaucoup plus grand que ses équivalents P-TPN utilisés avec HPSO.

Contrairement à GFBS, l'algorithme HPSO utilise des contrôleurs de blocage qui réduisent encore davantage l'espace de recherche. De plus, pour HPSO, la recherche s'arrête uniquement si une durée de fonctionnement maximale prédéfinie est atteinte. En effet, même en utilisant un processeur aux performances supérieures (ordinateur 2,6 GHz avec mémoire RAM 4G), la durée de fonctionnement

de l'algorithme HPSO est comprise entre 500 secondes sur FMS1 et 5000 secondes sur FMS20. Pour l'algorithme GFBS, et malgré l'utilisation de différents couples de paramètres selon la taille de l'espace d'état de l'instanciation, le temps d'exécution est compris entre 1 seconde sur FMS1 et 80 secondes sur FMS20. Cette différence importante en temps de calcul, en faveur de GFBS, est due à l'appel récursif de l'algorithme HPSO ce qui n'est pas le cas pour GFBS. Malgré sa complexité en temps et en mémoire, l'algorithme GFBS a trouvé de meilleures solutions pour 10 instanciations sur 20 et correspond aux solutions des 10 instanciations restantes.

La comparaison avec l'algorithme IHFBS conduit aux remarques suivantes : en termes de mémoire, l'algorithme IHFBS utilise, ainsi que HPSO, le modèle P-TPN qui est plus petit que son équivalent T-TPN utilisé avec GFBS. De plus, contrairement à l'algorithme GFBS qui n'est exécuté qu'une seule fois, IHFBS est basé sur un appel répété de l'algorithme de recherche où la valeur de la fonction objectif de l'appel en cours est utilisée comme borne supérieure pour l'appel suivant. Ainsi, le nombre de candidats explorés est beaucoup plus important, ce qui influence le temps de calcul nécessaire pour exécuter l'algorithme. L'algorithme GFBS n'a pas besoin de plus d'une seconde pour obtenir le résultat de la plus petite instanciation FMS1, alors que le temps nécessaire pour la plus grande instanciation FMS20 était de 80 secondes.

Pour IHFBS, même en utilisant un processeur plus performant (ordinateur 3,2 GHz avec mémoire RAM 8G), les temps de fonctionnement allaient de 40 secondes sur FMS1 à 1600 secondes sur FMS20. Malgré sa complexité en temps et en mémoire, l'algorithme GFBS a trouvé les mêmes résultats pour 16 des 20 instanciations. Dans les 4 cas restants, les résultats obtenus avec IHFBS sont légèrement meilleurs. Cette légère amélioration est obtenue grâce à une grande complexité en temps et en mémoire permettant d'explorer une plus grande partie de l'espace d'état par rapport à celui exploré par GFBS.

Les valeurs reportées dans le tableau 1 illustrent que la complexité temporelle des approches développées dans [Han et al., 2015] et [Mejia & Nino, 2017] dépasse largement le temps requis par GFBS même si la fréquence du processeur utilisé était plus grande.

4.7.Conclusion

Une nouvelle stratégie d'ordonnancement, qui calcule de manière incrémentale les séquences de contrôle pour les FMS hybrides, a été présentée. La méthode utilise un modèle T-TPN, une organisation itérée des opérations pour les gammes opératoires hybrides et une nouvelle variante de méthodes de recherche en faisceau qui explore sélectivement l'espace d'état PN en fonction des

générations successives de candidats. La fonction coût utilisée par la méthode GFBS s'est avérée fournir une borne inférieure de la durée de la trajectoire. Une comparaison avec d'autres méthodes d'ordonnancement a été présentée afin d'illustrer les résultats de l'algorithme GFBS par rapport à d'autres algorithmes de la littérature. Jusqu'ici, nous avons considéré que l'ordonnancement est effectué dans un environnement sûr et sans prendre en compte les interruptions pouvant affecter le système. Le chapitre suivant est consacré à l'étude de l'ordonnancement dans un environnement incertain.

Chapitre 5 : Recherche d'ordonnancement dans un environnement incertain

5.1. Introduction

Les approches d'ordonnancement usuelles se concentrent sur les modèles utilisés pour des environnements sûrs et invariants [Cherif, Leclercq & Lefebvre, 2021a ; Cherif, Leclercq, & Lefebvre, 2019a ; Lefebvre, 2016 ; Baruwa, Piera & Guasch, 2015 ; Mejía, & Montoya, 2009], dans lesquels le temps de traitement de chaque opération est censé être connu et supposé rester constant lors de la réalisation de la gamme opératoire.

De plus, ces méthodes utilisent des valeurs exactes comme le temps d'exécution attendu et la vitesse de la machine afin de représenter l'état du système réel. Cependant, ces valeurs ne sont pas toujours connues exactement, et des difficultés surviennent lorsque certains délais de traitement des opérations (qui étaient supposés être connus et constants) varient en raison d'interruptions.

En réalité, il existe plusieurs types de risques qui peuvent perturber un plan de production et affecter le coût d'ordonnancement. En effet, des paramètres tels que la disponibilité des matières premières, la fiabilité des machines et les exigences du marché sont souvent sujets à des écarts inattendus.

Imaginons, à titre d'exemple, un atelier avec un bras robotisé manipulateur qui est utilisé par deux gammes opératoires telles que la production de l'atelier dépende de la vitesse du manipulateur. Le déplacement du bras à vitesse maximale présente l'avantage d'augmenter la production mais peut conduire à des erreurs de positionnement et de répétabilité qui aboutiront finalement à une interruption du fonctionnement. La diminution de la vitesse du bras ralentit la production mais réduit le risque d'interruption. En ce sens, ajouter une marge de temps à certaines opérations (celles qui nécessitent le manipulateur) diminuera le risque d'interruption et augmentera la robustesse du l'ordonnancement.

Par conséquent, il est important de développer des méthodes systématiques pour résoudre le problème de l'ordonnancement dans un environnement incertain, afin de créer des ordonnancements efficaces et fiables.

Dans la deuxième section de ce chapitre, le contrôle des incertitudes est abordé. La troisième section s'intéresse à la modélisation des incertitudes. Une nouvelle fonction coût qui prend en compte le risque est proposée dans la section quatre. Pour prendre en compte le risque, une mise à jour de l'algorithme GFBS est présentée dans la section cinq. La sixième section propose une nouvelle variante de recherche en faisceau appelée Generation Double Filtered Beam Search (GDFBS). Dans la dernière section, les méthodes GFBS et GDFBS sont appliquées sur des exemples de la littérature.

5.2. Contrôle des incertitudes

Bien qu'un effort important ait été consacré aux problèmes d'ordonnancement avec les PN, seuls quelques travaux ont introduit la résilience ou l'analyse de vulnérabilité. Les incertitudes, qui affectent le système, sont supposées résulter d'événements inattendus qui affectent les performances du système. On suppose que ces incertitudes se produisent en raison de pannes de fonctionnement et de ressources peu fiables. Pour faire face aux incertitudes, deux types d'approches ont été étudiées dans la littérature d'ordonnancement. Il s'agit de l'ordonnancement réactif [Leon, Wu & Storer, 1994] et l'ordonnancement robuste [Vieira, Herrman & Lin, 2003].

5.2.1 Ordonnancement réactif

Les approches du premier type réalisent un ordonnancement réactif (également connu sous le nom de re-ordonnancement ou ordonnancement dynamique) [Leon, Wu & Storer, 1994]. Cet ordonnancement est établi pour faire évoluer un système sous l'hypothèse que ce dernier peut retrouver un état normal après l'occurrence d'un état défectueux [Cabasino, Giua & Seatzu, 2009]. L'ordonnancement réactif fait référence aux modifications apportées à l'ordonnancement d'origine visant à compenser les perturbations qui se produisent dans l'atelier : ces approches sont utilisées pour maintenir la faisabilité de l'ordonnancement afin de tenir compte des perturbations après leur apparition. L'aspect réactif de ces approches permet donc de reconfigurer l'ordonnancement à partir du moment où la perturbation arrive dans l'atelier.

5.2.2 Ordonnancement robuste

Les approches du second type réalisent un ordonnancement robuste [Vieira, Herrman & Lin, 2003] qui garantit qu'un système fonctionne d'une manière prédéfinie malgré des changements inattendus : ces approches incluent l'insertion de temps d'inactivité [Mehta & Uzsoy, 1998], l'ajout des tampons [Jamili, 2016 ; Moradi & Shadrokh, 2018], l'ajout de sous-réseaux de récupération pour les ressources non fiables et les siphons minimaux [Liu et al., 2013], en recherchant les ordonnancements résilients à l'aide de l'optimisation multi-objectifs [Al-Hinai & Elmekkawy, 2011 ; Liu, Gu & Xi, 2007], en utilisant une approche modulaire basée sur le concept de plug and play permettent de représenter des éventuelles perturbations qui se produiront durant l'exécution de l'ordonnancement [Himmiche et al., 2019] et en combinant les objectifs d'ordonnancement et de risque [Cherif, Leclercq & Lefebvre, 2021b ; Cherif, Leclercq & Lefebvre, 2019b ; Mejía & Lefebvre, 2019, Lefebvre, 2017b ; Zafra-Cabeza, Ridao & Camacho, 2004]. Le but d'un ordonnancement robuste est de prendre en compte les éventuelles perturbations tout en

construisant l'ordonnancement d'origine qui permet de le rendre plus robuste. Un ordonnancement robuste est défini comme un ordonnancement insensible aux perturbations imprévues [Leon, Wu & Storer, 1994].

Chaque approche présente des avantages et des inconvénients. L'insertion de temps d'inactivité ou l'ajout de tampons peut être efficace pour les ordonnancements qui nécessitent peu d'ajustements, mais finiront par conduire à un temps d'inactivité excessif. Les méthodes multi-objectifs peuvent produire plusieurs ordonnancements sur une frontière de Pareto avec différents degrés de risque et de coût. Cela donne au décideur la possibilité de sélectionner l'ordonnancement qui correspond au mieux à ses objectifs. Cependant, (i) ce choix peut être difficile à faire parmi les nombreuses solutions et (ii) la génération de fronts de Pareto peut devenir rapidement compliquée [Collette et Siarry, 2004]. La dernière approche [Cherif, Leclercq & Lefebvre, 2021b ; Cherif, Leclercq & Lefebvre, 2019b ; Mejía & Lefebvre, 2019, Lefebvre, 2017b ; Zafra-Cabeza, Ridao & Camacho, 2004] propose un compromis entre les deux objectifs risque et coût : l'avantage est que les calculs des fonctions objectif sont des extensions du cas usuel (absence de risque).

Notre objectif est d'éviter des temps de calcul excessifs tout en faisant face aux incertitudes et en limitant le risque de défaillances dû aux pannes des opérations et ressources. Dans ce sens, les travaux les plus récents font appel aux superviseurs afin d'interdire les situations inacceptables résultant des perturbations [Liu, Lefebvre & Li, 2019 ; Liu et al., 2018]. La contribution proposée dans ce travail est différente en ce sens qu'aucun superviseur n'est ajouté au modèle original ; à la place, une évaluation des risques est intégrée dans la fonction coût [Cherif, Leclercq & Lefebvre, 2021b ; Cherif, Leclercq & Lefebvre, 2019b ; Mejía & Lefebvre, 2019]. En effet, la stratégie de modélisation est d'abord améliorée afin de prendre en compte l'occurrence des incertitudes qui présentent des défaillances et des événements imprévus. Ensuite, la fonction coût est mise à jour pour intégrer le risque dans l'évaluation de l'ordonnancement.

5.3. Modélisation des incertitudes avec PN

Les incertitudes ou les événements imprévus sont supposés se produire soit en raison d'une interruption des opérations, soit en raison de l'indisponibilité des ressources. Ces pannes ne sont pas définitives et les processus de reprise sont également pris en compte. Les événements inattendus sont modélisés par des transitions non commandables tandis que les processus de récupération sont modélisés avec des transitions commandables [Cherif, Leclercq & Lefebvre, 2021b ; Cherif, Leclercq, & Lefebvre, 2019b ; Lefebvre, 2014]. Afin de modéliser les incertitudes, un modèle T-TPN partiellement commandable est utilisé.

5.3.1 T-TPN partiellement contrôlable

Dans un T-TPN partiellement commandable, l'ensemble des transitions T est divisé en deux sous-ensembles disjoints T_C et T_{NC} tels que $T = T_C \cup T_{NC}$ où T_C est le sous-ensemble des transitions commandables et T_{NC} le sous-ensemble des transitions non commandables représentant les événements inattendus. L'ensemble des transitions commandables T_C est divisé en deux sous-ensembles disjoints $T_C = T_N \cup T_R$ où T_N est le sous-ensemble des transitions commandables normales, et T_R est le sous-ensemble des transitions commandables de récupération permettant de récupérer le système de son état défectueux et de revenir à un état normal. Les transitions commandables sont tirées selon la politique de tir au plus tôt avec des spécifications de temps similaires à celles utilisées avec les T-TPN [Ramchandani, 1974] où chaque transition $t \in T_C$ se déclenche dès que sa contrainte de temps $D(t)$ est satisfaite. La politique de mémoire (enabling memory policy) [Molloy, 1982] est utilisée pour les transitions commandables où, après le déclenchement d'une transition t et le passage d'un marquage M à un autre marquage M', le temps écoulé pour les transitions encore activées est conservé et décrémenté et le temps écoulé associé aux transitions désactivées est oublié. Enfin, Le choix des transitions commandables à tirer est fait par la commande (politique de présélection).

Dans ce travail, nous adoptons la politique de serveur unique où au plus une instance de déclenchement par transition dans chaque état est possible. Les transitions non commandables se déclenchent arbitrairement à tout moment à partir de leur date d'activation, selon un processus aléatoire piloté par une fonction de densité de probabilité exponentielle (pdfs) de paramètre $\mu > 0$. La politique de mémoire (resampling memory policy) est utilisée pour les transitions non commandables [Molloy, 1982]. Afin de résoudre les conflits qui incluent une transition non commandables une politique de choix par compétition est utilisée où les paramètres de temps sont utilisés pour calculer la probabilité de déclenchement des transitions.

Par exemple, sur la figure 47, la probabilité de déclencher la transition commandable t_1 est $e^{-\mu_2 d_1}$ et la probabilité de déclencher la transition non commandable t_2 est $1 - e^{-\mu_2 d_1}$. Si t_1 est sélectionné, son temps de tir est d_1, sinon t_2 est sélectionné et le temps de tir moyen est : $(1/\mu_2 - \exp(-\mu_2 \times d_1) \times (d_1 + 1/\mu_2))/(1 - \exp(-\mu_2 \times d_1))$.

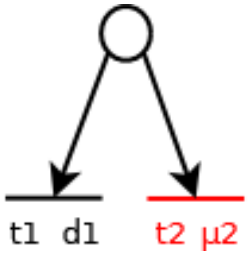


Figure 47 : Résolution des conflits avec une politique de choix par compétition

Dans ce qui suit, le T-TPN partiellement commandable est utilisé pour la modélisation des ateliers afin de pouvoir modéliser les événements inattendus qui affectent les opérations ainsi que les ressources.

5.3.2 Interruption d'une opération

La modélisation de l'interruption d'une opération est illustrée sur la figure 48. L'exemple consiste en une opération o_1, de durée d_1, effectuée sur la ressource r_1. L'opération est interrompue et retardée en raison de l'occurrence d'une panne représentée dans le modèle par le franchissement de tfo_1. L'interruption de l'opération est modélisée par un sous-réseau composé d'une place de récupération pro_1, d'une transition d'interruption tfo_1 et d'une transition de récupération tro_1.

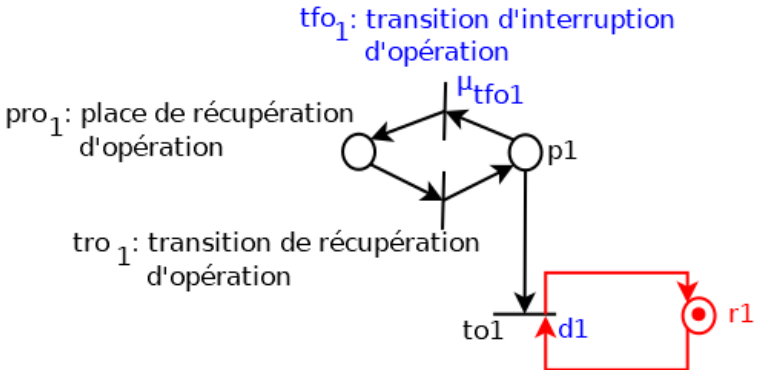


Figure 48 : Interruption d'une opération

Le risque est explicitement défini comme la somme du risque pouvant affecter certaines opérations (somme des durées des retards induits par les pannes) : chaque risque est le produit de la probabilité de s'écarter de l'échéancier attendu par la durée supplémentaire induite par cet écart.

Les pannes des opérations et des ressources sont modélisées comme des processus probabilistes indépendants qui n'interagissent pas. Ainsi, la probabilité de panne de chaque opération ou ressource est calculée séparément et n'est pas affectée par d'autres pannes. Lorsque la politique de choix par compétition est utilisée, le risque de panne de l'opération est défini par :

$$risk(to_1) = \left(1 - \exp\left(-\mu_{tfo_1} \times d_1\right)\right) \times \left(d_{tro_1} + d_{tfo_1}\right) \quad (8)$$

où d_{tfo_1} est la durée moyenne de tir de tfo_1 : $d_{tfo_1} = (1/\mu_{tfo_1} - \exp(-\mu_{tfo_1} \times d_1) \times (d_1 + 1/\mu_{tfo_1}))/(1 - \exp(-\mu_{tfo_1} \times d_1))$.

5.3.3 Indisponibilité des ressources

Les performances du système sont affectées non seulement en raison de l'interruption des opérations, mais également en raison de ressources peu fiables. En effet, la réalisation de la séquence de commande peut être retardée. Sur la figure 49, la ressource r_1 est requise pour effectuer l'opération o_1. Lorsque la transition tfr_1 est tirée, r_1 n'est plus disponible et redevient disponible seulement après le tir de la transition de récupération de ressource trr_1. Lorsque la politique de choix par compétition est utilisée, le risque peut être évalué par :

$$risk(to_1) = \left(1 - exp\left(-\mu_{tfr_1} \times d_1\right)\right) \times \left(d_{trr_1} + d_{tfr_1}\right) \quad (9)$$

où d_{tfr_1} est la durée moyenne de tir de tfr_1 : $d_{tfr_1} = (1/\mu_{tfr_1} - \exp(-\mu_{tfr_1} \times d_1) \times (d_1 + 1/\mu_{tfr_1}))/(1 - \exp(-\mu_{tfr_1} \times d_1))$.

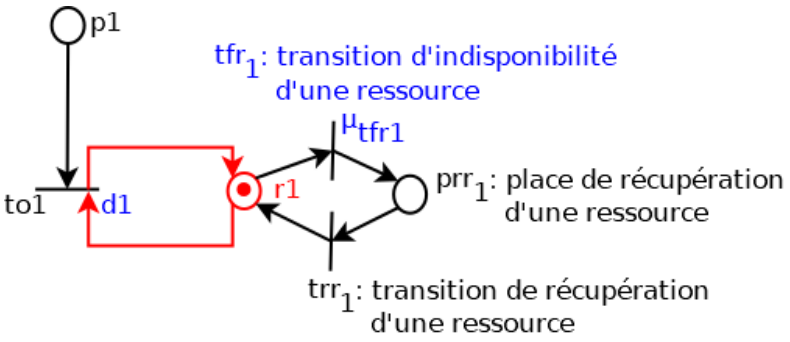


Figure 49 : Indisponibilité d'une ressource

Les transitions non commandables sont utilisées pour représenter soit les pannes des opérations, comme indiqué sur la figure 48, soit l'indisponibilité des ressources, comme illustré sur la figure 49. Le risque de retard pour une opération, résulte de la somme du risque d'interruption de l'opération et du risque de défaillance des ressources. Ce risque associé à la transition to_i et défini par :

$$risk(to_i) = \left(1 - exp\left(-\mu_{tfo_i} \times d_i\right)\right) \times \left(d_{tro_i} + d_{tfo_i}\right) + \sum_{r_j \in R_i} \left(1 - \exp\left(-\mu_{tfr_j} \times d_i\right)\right) \times \left(d_{trr_j} + d_{tfr_j}\right) \quad (10)$$

La première partie de l'équation (10) représente le risque issu de l'interruption de l'opération tandis que la seconde partie représente le risque résultant de la défaillance des ressources nécessaires pour le traitement de l'opération. Par extension, le risque associé à un chemin donné est défini par :

$$risk(Path) = \sum_{t \in Path} risk(t) \quad (11)$$

Dans la suite, le chemin de x_1 à x_K avec une durée minimale (resp. un risque minimal) est noté $Path_d^*(x_1, x_K)$ (resp. $Path_r^*(x_1, x_K)$). Nous notons $\chi_d^*(x_1, x_K)$ la durée de $Path_d^*(x_1, x_K)$ et $\chi_r^*(x_1, x_K)$ le risque de $Path_r^*(x_1, x_K)$.

Un T-TPN partiellement commandable est utilisé pour la modélisation d'un FMS complexe avec la présence d'événements inattendus. Le risque de déviation n'a pas été pris en compte dans la fonction coût proposée dans le chapitre 4. Afin de pouvoir prendre en compte ce risque dans la recherche d'ordonnancement, une modification est apportée à la fonction coût.

5.4. Fonction coût prenant en compte le risque

La fonction coût est modifiée de telle sorte à intégrer le risque de déviation dans l'évaluation et la sélection des candidats pendant la recherche de l'ordonnancement [Cherif, Leclercq & Lefebvre, 2021b ; Cherif, Leclercq & Lefebvre, 2019b ; Mejía & Lefebvre, 2019]. Dans cette partie, une nouvelle fonction coût est ainsi définie dans le but de quantifier non seulement la durée mais aussi le risque [Cherif, Leclercq & Lefebvre, 2021b ; Cherif, Leclercq & Lefebvre, 2019b]. Le risque est considéré comme un retard nécessaire pour réparer le système et le remettre dans un état de fonctionnement normal.

Pour chaque candidat avec un marquage intermédiaire *M* d'une trajectoire donnée (σ, M_0) devant atteindre le marquage de référence M_{ref}, la trajectoire est divisée en deux parties. La première partie présente la trajectoire déjà calculée (σ_1, M_0) de M_0 à *M* et la deuxième partie présente la trajectoire résiduelle (σ_2, *M*) avec une séquence inconnue σ_2 de *M* à M_{ref}. La durée et le risque de la trajectoire déjà calculée sont respectivement appelés $g_d(\sigma_1, M_0)$ et $g_r(\sigma_1, M_0)$ tandis que les fonctions $h_d(M, M_{ref})$ et $h_r(M, M_{ref})$ approchent respectivement la durée et le risque de la trajectoire résiduelle (σ_2, *M*). La fonction coût est définie par :

$$f_{d+r}(M_0, M, M_{ref}) = g_d(\sigma_1, M_0) + g_r(\sigma_1, M_0) + h_d(M, M_{ref}) + h_r(M, M_{ref}) \quad (12)$$

Le risque de la trajectoire déjà calculée entre M_0 et *M* est calculé en utilisant le même algorithme que celui utilisé pour le calcul de la durée $g_d(\sigma_1, M_0)$ de (σ_1, M_0) en transformant la trajectoire non temporisée en une trajectoire temporisée sous la politique de tir au plus tôt, i.e., chaque transition *t* de σ_1 est tirée dès que sa contrainte de temps *D(t)* est satisfaite. L'algorithme, détaillé dans [Lefebvre, 2017a ; Lefebvre, 2017b] met à jour, à chaque nouveau marquage *M*, les durées et les risques résiduels pour pouvoir tirer les transitions activées.

Pour un ensemble d'opérations avec contraintes de précédence totale dans une gamme opératoire *J*, l'approximation $h_{d_J}(M, M_{ref})$ de la durée de la partie inconnue de la trajectoire est donnée par l'équation (13). Elle correspond à l'approximation $h_J(M, M_{ref})$ calculée dans la partie 4.3.1 du chapitre 4 :

$$h_{d_J}(M, M_{ref}) = M(s_J) \times D(J) + \chi^*(p_i, e_J) - EFT_d(to_i) \text{ tel que } to_i \in (p_i)\bullet \quad (13)$$

où $EFT_d(to_i)$ est un terme de correction qui correspond à la durée d'activation pour le tir de la première transition to_i du chemin de coût minimal entre p_i et e_J.

Les incertitudes sont modélisées avec des transitions non commandables. En effet, à un marquage *M* donné, le risque de dévier de la trajectoire optimale est estimé en fonction de la probabilité qu'une transition non commandable *t'* se déclenche au lieu d'une transition commandable *t*. Le risque résiduel est donné par (14) :

$$h_{r_J}(M, M_{ref}) = M(s_J) \times risq(J) + \chi^*(p_i, e_J) - EFT_r(to_i) \text{ tel que } to_i \in (p_i)\bullet \quad (14)$$

où $EFT_r(to_i)$ est un terme de correction qui correspond au risque écoulé pour le tir de la première transition to_i du chemin de coût minimal entre p_i et e_J.

En termes simples, $h_{r_J}(M, M_{ref})$ correspond à la somme des risques attribués aux opérations qui ne sont pas encore effectuées, calculés avec l'équation (10) et corrigés selon l'opération en cours : la première partie de l'équation (14) se réfère au produit du risque de la gamme opératoire *risq(J)* par le nombre de jetons dans la place de départ $M_r(s_J)$ (qui devront attendre la fin de l'exécution en cours de la gamme opératoire) et la deuxième partie se réfère au risque maximum des chemins les plus courts de chaque place marquée p_i jusqu'à la place de référence e_J.

L'exemple d'un ensemble d'opérations avec contraintes de précédence totale avec des interruptions d'opérations et des pannes de ressources, est illustré sur la figure 50. La gamme opératoire réalise o_1 qui nécessite la ressource r_1, puis o_2 qui nécessite r_2. La durée minimale de la gamme opératoire est $D(J) = d_1 + d_2$ et le risque minimal de la gamme opératoire est $risq(J) = risq(to_1) + risq(to_2)$.

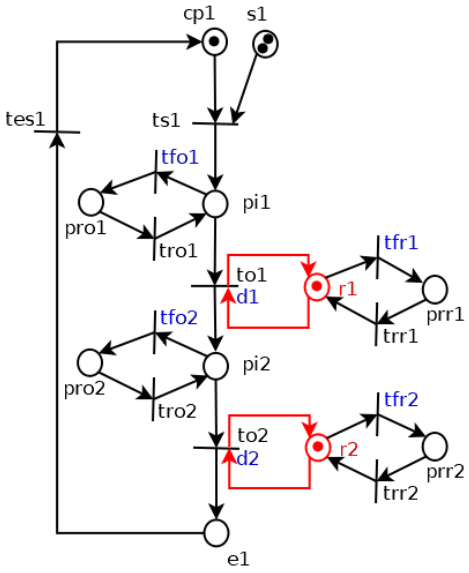


Figure 50 : Modèle T-TPN d'une séquence d'opérations

Pour un ensemble d'opérations avec une flexibilité totale, l'approximation $h_{d_J}(M, M_{ref})$ du coût de la partie inconnue de la trajectoire est donnée par l'équation (15). Elle correspond à l'approximation $h_J(M, M_{ref})$ calculée dans la partie 4.3.2 du chapitre 4 :

$$h_{d_J}(M, M_{ref}) = M(s_J) \times D(J) + \sum_{f_i \in PF} (M(f_i) \times D(to_i) + M(p_i) \times D(to_i)) - EFT_d(to_i) \text{ tel que } to_i \in (p_i)\bullet \quad (15)$$

L'approximation $h_{r_J}(M, M_{ref})$ du risque est donnée aussi par la somme des risques d'opérations non encore exécutées, et est égale au produit du marquage de la place $M(f_i)$ par le risque $risq(to_i)$ ajouté au produit du marquage de la place d'opération $M(p_i)$ par le risque d'opération $risq(to_i)$. Un terme de correction $EFT_r(to_i)$, qui correspond au risque écoulé pour le tir de la transition to_i, est ajouté pour l'opération en cours d'exécution. Le risque résiduel jusqu'à M_{ref} est estimé par (16) :

$$h_{r_J}(M, M_{ref}) = M(s_J) \times risq(J) + \sum_{f_i \in PF} (M(f_i) \times risq(to_i) + M(p_i) \times risq(to_i)) - EFT_r(to_i) \text{ tel que } to_i \in (p_i)\bullet \quad (16)$$

La figure 51 montre un exemple de deux opérations modélisées avec une flexibilité totale et en présence d'interruptions d'opérations et d'indisponibilités de ressources. La durée minimale pour traiter la gamme opératoire est $D(J) = d_1 + d_2$ et le risque de déviation pour la gamme opératoire est $risq(J) = risq(to_1) + risq(to_2)$.

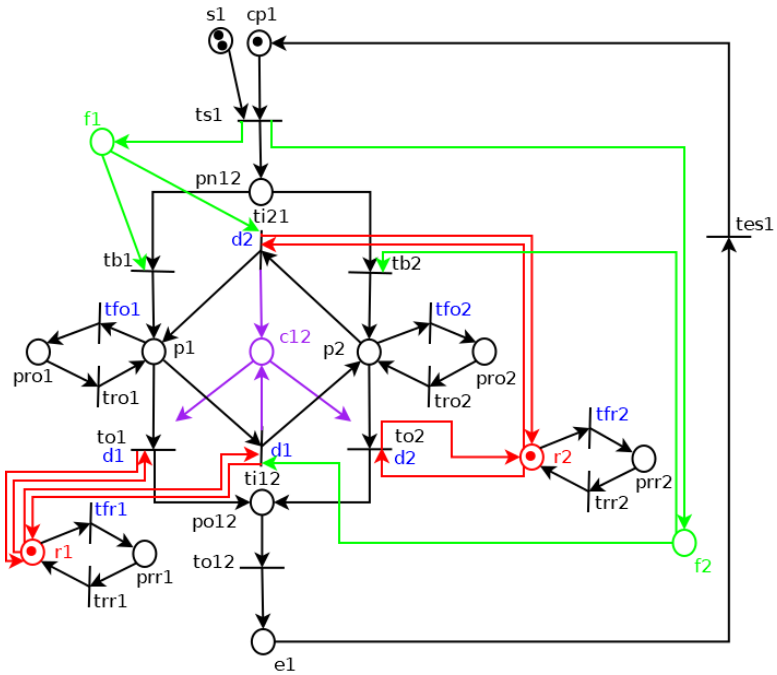


Figure 51 : Modèle T-TPN pour un ensemble d'opérations avec flexibilité totale

Pour un ensemble d'opérations avec une flexibilité partielle, l'approximation du risque de déviation $h_{r_J}(M, M_{ref})$ de la partie inconnue de la trajectoire est calculée selon la même stratégie que celle utilisée dans la section 4.3.3 du chapitre 4 pour calculer l'approximation de durée $h_{d_J}(M, M_{ref})$ en se basant sur l'organisation itérée de l'atelier [Cherif, Leclercq & Lefebvre, 2021a ; Cherif, Leclercq & Lefebvre, 2019b ; Cherif, Leclercq & Lefebvre, 2019a].

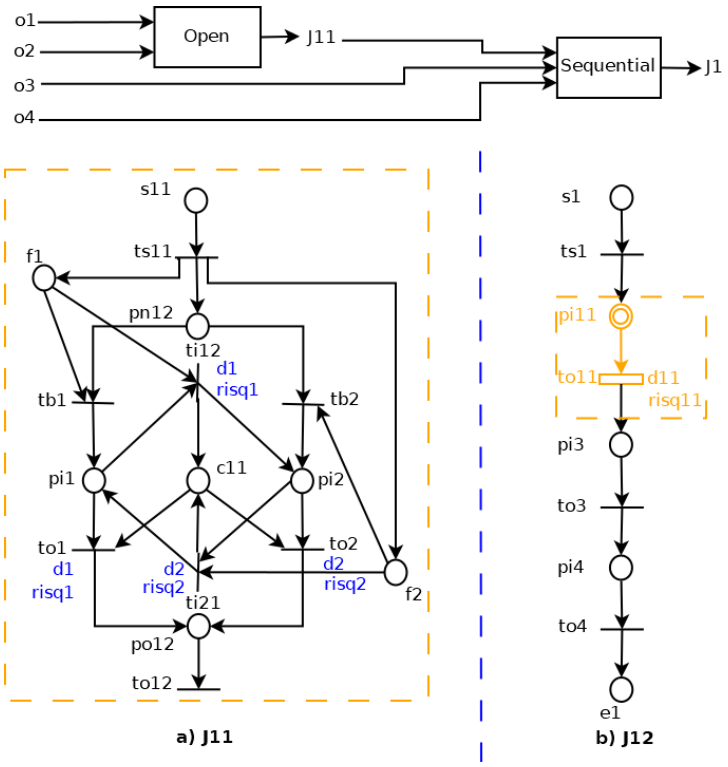


Figure 52 : Propagation du marquage, de la durée résiduelle et du risque résiduel pour une gamme opératoire hybride

Exemple : Prenons l'exemple de la gamme opératoire hybride J_1 illustré à la figure 52 avec deux gammes opératoires de base $\{J_{11}, J_{12}\}$ où J_{11} est la seule gamme de base du premier niveau et J_{21} appartient au deuxième et dernier niveau. En commençant par le premier niveau et la gamme opératoire de base J_{11} (composée de deux opérations avec une flexibilité totale $\{o_1, o_2\}$), l'estimation de la durée résiduelle est évaluée à l'aide de l'équation (15) et l'estimation du risque résiduel est évaluée à l'aide de l'équation (16). Puis les paramètres de temps d_{11} et de risque $risq_{11}$ de la transitions dynamique to_{11} de la gamme opératoire de base J_{12} au niveau 2 sont mis à jour. Le marquage est également mis à jour pour la place dynamique pi_{11} de la gamme opératoire de base J_{12}. Par la suite, les estimations de la durée résiduelle et du risque résiduel pour la gamme opératoire J_{12} du dernier niveau 2 sont évaluées à l'aide des équations (13) et (14). La durée résiduelle et le risque résiduel de

la gamme opératoire hybride sont finalement obtenus par $h_{d_J}(M, M_{ref}) = h_{d_{J12}}(M, M_{ref})$ et $h_{r_J}(M, M_{ref}) = h_{r_{J12}}(M, M_{ref})$

La durée résiduelle de l'ensemble du FMS est obtenue comme la valeur maximale des durées résiduelles sur toutes les gammes opératoires hybrides du FMS :

$$h_d(M, M_{ref}) = max\{h_{d_J}(M, M_{ref}) : J \in \boldsymbol{J}\} \quad (17)$$

De même, le risque résiduel de l'ensemble du FMS est obtenu comme la valeur maximale des risques résiduels sur toutes les gammes opératoires hybrides du FMS :

$$h_r(M, M_{ref}) = max\{h_{r_J}(M, M_{ref}) : J \in \boldsymbol{J}\} \quad (18)$$

Notez que $h_d(M, M_{ref})$ et $h_r(M, M_{ref})$ ne sont pas nécessairement issues de la même gamme opératoire.

5.5. Algorithme de recherche GFBS

Nous présentons ci-dessous le pseudo-code de l'algorithme GFBS [Cherif, Leclercq & Lefebvre, 2021b ; Cherif, Leclercq & Lefebvre, 2019b] qui a été modifié pour prendre en compte le risque de déviation dans le choix des candidats. L'algorithme renvoie une séquence valide de tir *Seq* ainsi que son coût C_{max}. L'algorithme utilise une liste, appelée OPEN, pour stocker les candidats à développer d'une génération donnée. La nouvelle fonction coût *f* détaillée à la section 5.4 est utilisée pour trier et sélectionner les candidats d'une génération. La liste OPEN est mise à jour après chaque itération (une itération correspond à l'exploration de tous les candidats d'une génération donnée).

Algorithme : GFBS Entrées : N, M_0, β_g, β_l, M_{ref} Sorties : *Seq*, C_{max}

1. Générer les successeurs de M_0 et placer les β_g meilleurs candidats dans OPEN.
2. Si OPEN est vide, terminer et retourner *Seq* = ∅ and $C_{max} = \infty$.

Sinon, pour chaque candidat *M* dans *OPEN* faire

a. Générer les successeurs de *M*.

b. Pour chaque successeur *M'* issu de *M*.

Si le successeur *M'* est égal à M_{ref}, construire la séquence de transitions *Seq* de M_0 à M_{ref} et renvoyer *Seq* et C_{max} comme la somme de durée et risque de *Seq*,

Sinon, calculer la fonction coût $f_{d+r}(M_0, M', M_{ref})$ et placer M' dans une liste temporaire TEMPLIST.

c. Sélectionner jusqu'au β_l meilleurs candidats de TEMPLIST et les placer dans une liste temporaire GENERATION en respectant :

Si M' est égal à un marquage déjà présent dans GENERATION, et si un meilleur chemin a été trouvé, mettre à jour le chemin de M_0 à M et le coût $f_{d+r}(M_0, M', M_{ref})$,

Sinon, placer M' dans GENERATION.

d. Vider TEMPLIST.

3. Vider OPEN.
4. Sélectionner les β_g meilleurs candidats de GENERATION et les placer dans OPEN.
5. Vider GENERATION.
6. Retourner à l'étape 2.

Un avantage de l'algorithme proposé est d'éviter systématiquement les blocages et autres candidats ne pouvant atteindre M_{ref} et qui sont a priori inconnus pour le contrôleur. En effet, un coût infini est attribué pour ces candidats.

Afin de faire la différence entre l'application de l'algorithme GFBS avec une fonction coût prenant en compte uniquement la durée pour la sélection des candidats et l'application de l'algorithme GFBS avec la nouvelle fonction coût prenant en compte la durée et le risque pour la sélection des candidats, nous appellerons GFBS-D la variante basée sur la durée et GFBS-DR pour celle basée sur la durée et le risque.

Exemple :

Reprenons l'exemple du chapitre précédant, illustré figure 43 et composé de deux gammes opératoires $\{J_1, J_2\}$ et de trois ressources $\{r_1, r_2, r_3\}$, auquel nous rajoutons des risques d'interruption d'opération et d'indisponibilité de ressource. L'exemple est présenté sur la figure 53. Chaque gamme opératoire est composée d'une séquence de deux opérations : J_1 est la séquence des opérations $\{o_{11}, o_{12}\}$ et J_2 est la séquence des opérations $\{o_{21}, o_{22}\}$. La ressource r_1 est partagée entre les deux gammes opératoires et plus précisament entre les deux opérations o_{11} et o_{12}.

Pour chaque opération, une durée d_{Ji} est définie : $d_{11} = 14$; $d_{12} = 15$; $d_{11} = 13$; $d_{11} = 17$, les paramètres des transitions sont donnés dans le tableau 2. Ainsi, $risk_{Ji}$ est calculé avec l'équation (8) et est défini pour chaque opération : $risk_{11} = 8$; $risk_{12} = 10$; $risk_{11} = 7$; $risk_{11} = 2$.

Tableau 2 : Paramètres des opérations d'un FMS en présence de risque

T	$D(t)$	T	$D(t)$	T	$D(t)$	T	μ	T	μ
to_{11}	14	tro_1	7	trr_1	1	tfo_1	2	tfr_1	6
to_{12}	15	tro_2	3	trr_2	7	tfo_2	8	tfr_2	1
to_{21}	13	tro_3	6	trr_3	1	tfo_3	2	tfr_3	8
to_{22}	17	tro_4	1			tfo_4	8		

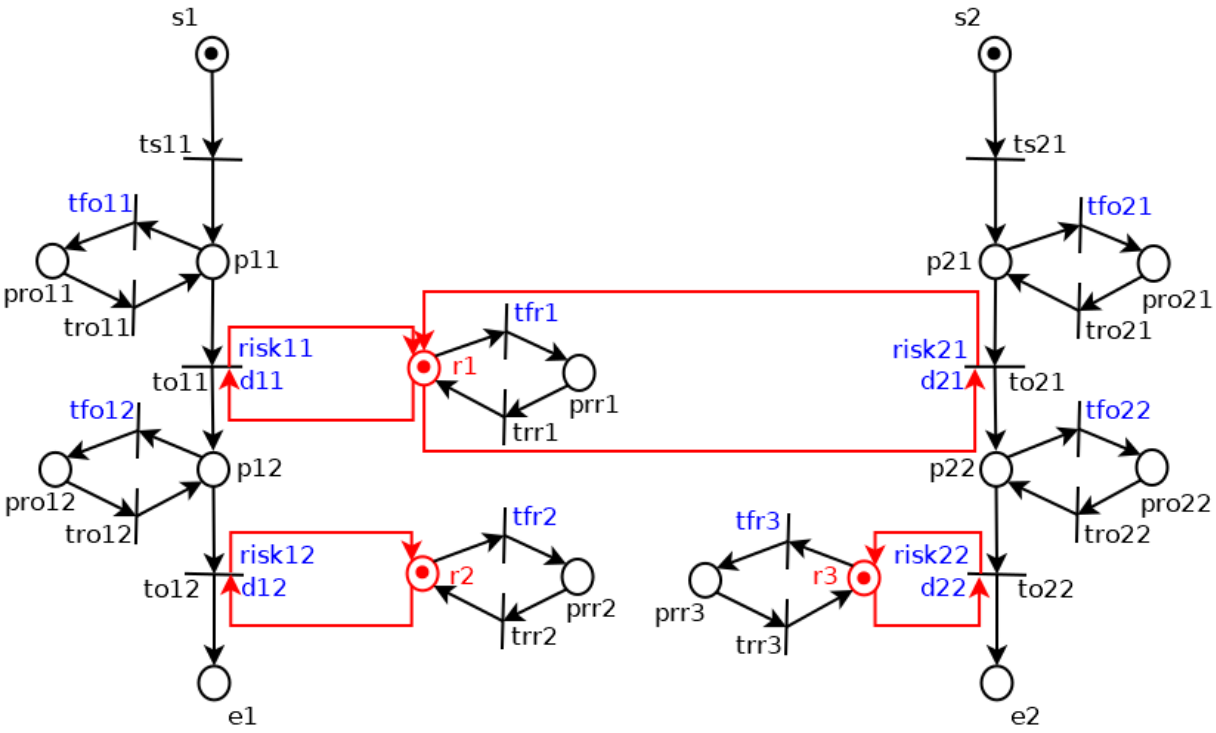


Figure 53 : Exemple d'un FMS en présence de risque d'interruptions

Pour cet exemple, les différentes générations sont présentées. En utilisant les paramètres $\beta_g = 3$ et $\beta_l = 2$, les résultats obtenus sont illustrés sur la figure 54.

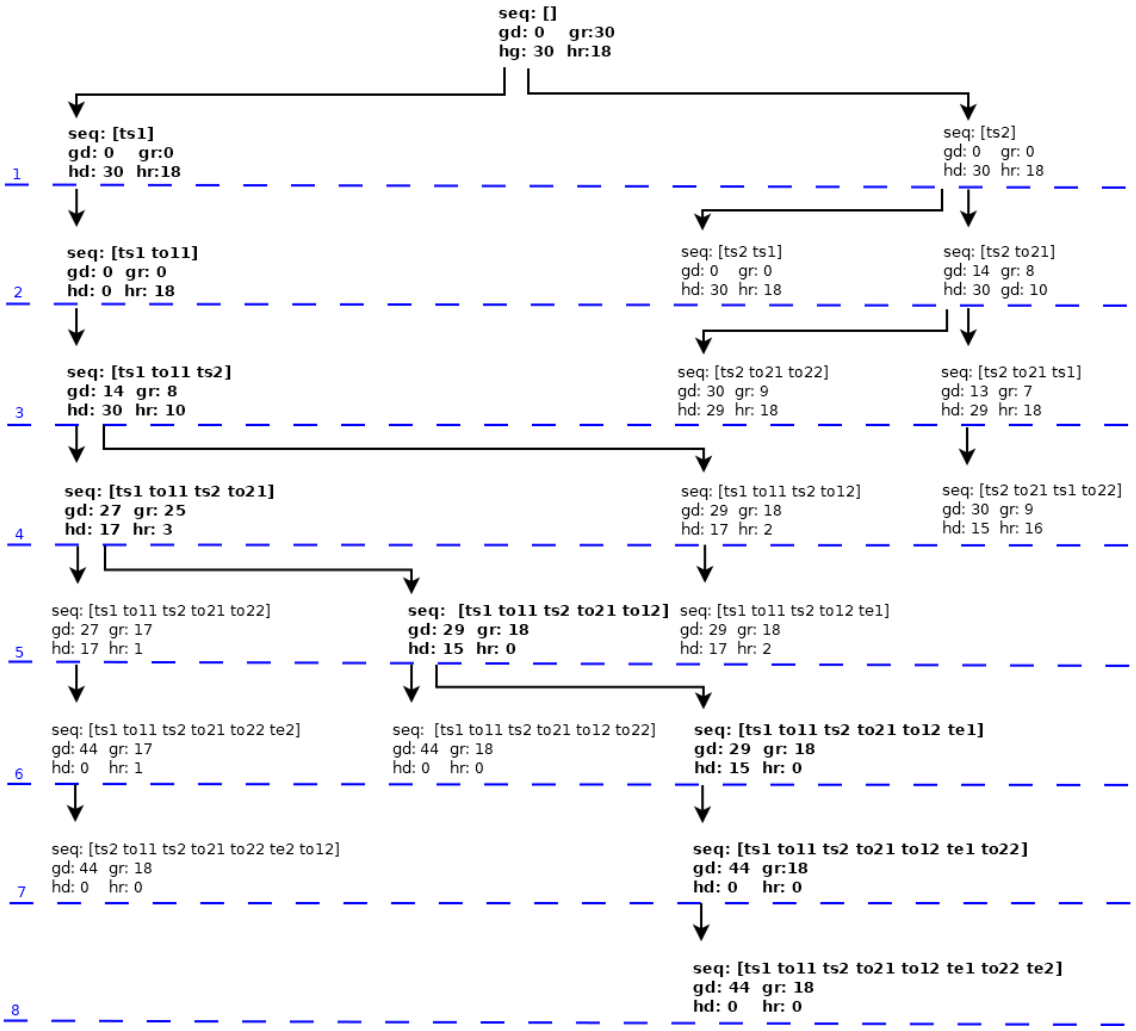


Figure 54 : Détails de l'exploration du FMS par l'algorithme GFBS-DR, pour $\beta g = 3$ et $\beta l = 2$

En utilisant les algorithmes GFBS-D et GFBS-DR, les ordonnancements obtenus sont les suivants :

GFBS-D : $o_{21} \rightarrow o_{11} \rightarrow o_{22} \rightarrow o_{12}$

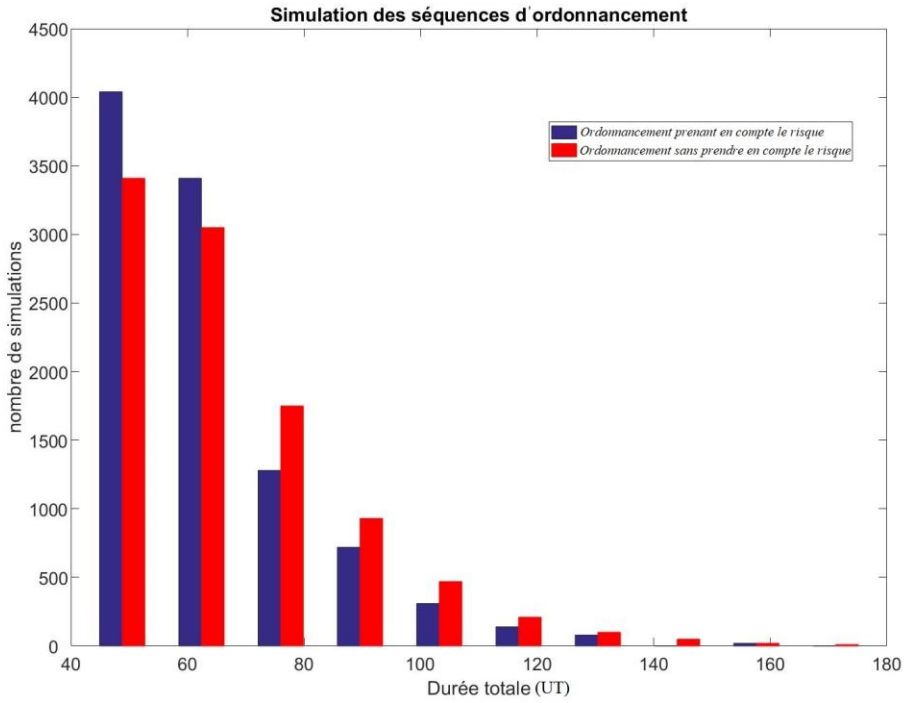
GFBS-DR : $o_{11} \rightarrow o_{21} \rightarrow o_{12} \rightarrow o_{22}$

En prenant en compte le risque pour la sélection des candidats et en utilisant l'algorithme GFBS-DR, le résultat est différent de celui obtenu sans prendre en compte le risque avec l'algorithme GFBS-D. La durée est légèrement dégradée de 5% lors de l'utilisation de l'algorithme GFBS-DR (44 UT) contre une durée de 42 UT avec l'algorithme GFBS-D. En revanche, le risque de l'ordonnancement obtenu avec l'algorithme GFBS-D dépasse largement le risque de l'ordonnancement obtenu avec l'algorithme GFBS-DR. Précisément, le risque est amélioré de 28% (18 UT avec l'algorithme GFBS-DR) contre 25 UT avec l'algorithme GFBS-D. Ainsi le coût total est amélioré de 8% en utilisant l'algorithme

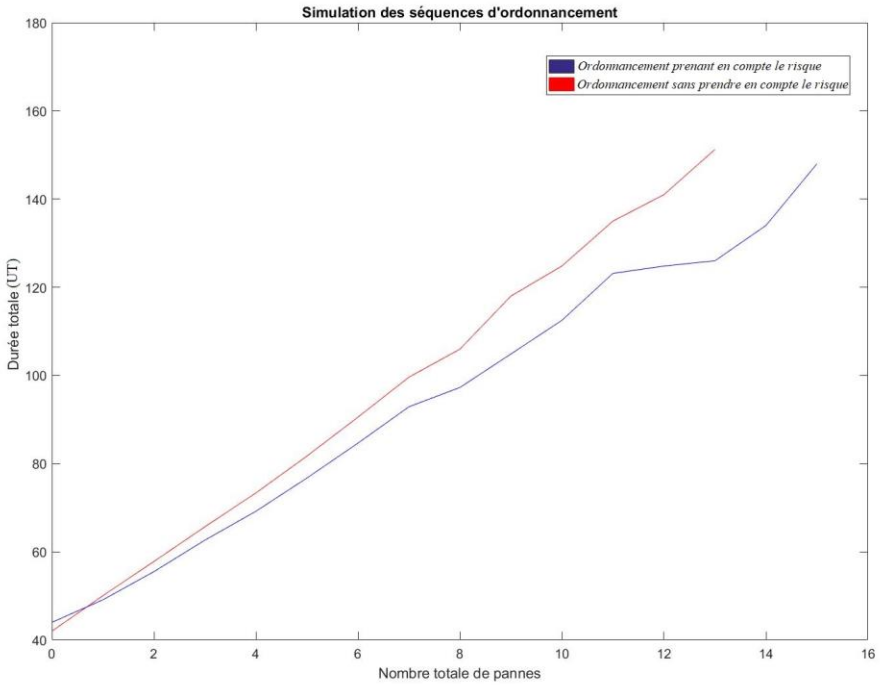
GFBS-DR par rapport à l'algorithme GFBS-D : l'algorithme GFBS-DR génère un ordonnancement avec un coût total égal à 62 UT tandis que l'algorithme GFBS-D génère un ordonnancement avec un coût total égal à 67 UT.

Pour conclure, la recherche d'un ordonnancement en prenant uniquement en compte la durée pour la sélection des candidats, permet de générer un ordonnancement avec une durée quasi optimale dans le cas idéal où aucun aléa de fonctionnement n'apparait alors que la prise en compte du risque de défaillance limite la dégradation des performances en présence d'aléa.

Dans ce contexte, une simulation est proposée afin d'illustrer les résultats obtenus. Les incertitudes sont simulées à l'aide d'un PN stochastique avec des transitions non commandables qui représentent les événements inattendus. Les transitions non commandables se déclenchent arbitrairement à tout moment à partir de leurs dates de validation selon des densités de probabilité exponentielles. Les simulations permettent de mesurer les écarts issus des pannes d'opération et de ressources en utilisant les ordonnancements obtenus avec et sans tenir compte du risque de déviation. En utilisant les séquences générées par chaque algorithme et les mêmes paramètres utilisés pour calculer les ordonnancements (Tableau 2), nous générons 10000 simulations pour chaque ordonnancement calculé puis comparons les durées totales moyennes. Les résultats de la simulation sont illustrés sur la figure 55.



(a) Histogramme des durées totales des scénarios



(b) Durée totale moyenne des scénarios en fonction du nombre de pannes

Figure 55 : Comparaison des ordonnancements obtenus avec et sans prise en compte des risques

L'ordonnancement obtenu à l'aide de l'algorithme GFBS-D, qui ne prend pas en compte le risque de défaillance est présenté en rouge sur la figure 55, alors que celui obtenu avec l'algorithme GFBS-DR qui prend en compte le risque de déviation est présenté en bleu.

On constate tout d'abord que les ordonnancements obtenus avec les différents algorithmes testés ne sont pas les mêmes et cela explique aussi pourquoi les performances (évaluées par simulation) sont différentes. La simulation montre que la durée totale moyenne des 10000 simulations obtenues en tenant compte du risque de pannes (algorithme GFBS-DR avec 62.6 UT) est améliorée de 6% par rapport à celle basée uniquement sur la durée (algorithme GFBS-D avec 66.3 UT). Ce gain est expliqué dans la figure 55.a dans laquelle on constate que pour une petite valeur de durée totale, le nombre de simulations obtenues avec l'ordonnancement généré par GFBS-DR est significativement plus élevé (pour les deux premières barres de l'histogramme). En contrepartie, pour une grande valeur de durée totale, le nombre de simulations obtenues avec l'ordonnancement généré par GFBS-D devient plus important (pour le reste des barres de l'histogramme).

La figure 55.b présente l'évolution du coût total par rapport au nombre de pannes pour les séquences obtenus avec les algorithmes GFBS-D (en rouge) et GFBS-DR (en bleu). En l'absence de pannes, le coût total est en faveur de la séquence obtenue avec l'algorithme GFBS-D avec un retard dû au risque égal à 0. La simulation montre qu'au fur et à mesure de l'augmentation du nombre de pannes, la différence en durée totale augmente en faveur de la séquence obtenue avec l'algorithme GFBS-DR. En effet, pour un même nombre de pannes, le retard dû aux perturbations est plus élevé pour la séquence obtenue avec GFBS-D par rapport à la séquence obtenue avec GFBS-DR. En conclusion, les résultats obtenus avec cette simulation confirment les conclusions obtenues par les calculs de risque.

5.6. Algorithme de recherche GDFBS

L'un des inconvénients de l'algorithme de recherche en faisceau est la difficulté à choisir la meilleure combinaison de paramètres (β_g et β_l) pour une instance donnée. Des valeurs plus élevées de β_g et β_l fournissent souvent de meilleures solutions mais avec un nombre de candidats plus important. Mais augmenter β_g et β_l n'améliore pas toujours la solution et peut même dans certains cas la dégrader. Afin d'améliorer la sélection des candidats à explorer, une nouvelle variante de l'algorithme GFBS est proposée.

L'objectif est d'obtenir un compromis entre la durée et le risque en sélectionnant des candidats sous des contraintes de durée et de risque. L'idée est de ne sélectionner que des candidats ayant à la fois une durée acceptable et un risque acceptable. Par conséquent, l'optimisation est réalisée sous les contraintes formalisées par l'utilisation de filtres. Ces filtres sont obtenus avec le seuil α_d appliqué sur la durée et α_r appliqué sur le risque. Dans le cas où il existe plus de candidats β_g ou β_l, l'algorithme ne sélectionnera que des candidats de durée inférieure à α_d ou de risque inférieur à α_r. Les candidats qui présentent à la fois une grande durée et un grand risque ne sont pas retenus afin d'améliorer la qualité des candidats sélectionnés et de converger plus efficacement vers un candidat avec à la fois une petite durée et un faible risque.

Dans ce travail, le seuil α_d (respectivement α_r) est fixé arbitrairement égal à la médiane des durées (respectivement risques) des candidats de la liste OPEN. Nous présentons ci-dessous le pseudo-code de l'algorithme GDFBS [Cherif, Leclercq & Lefebvre, 2021b ; Cherif, Leclercq & Lefebvre, 2019b].

Algorithme : GDFBS Entrées : N, M_0, β_g, β_l, M_{ref} Sorties : Seq, C_{max}

1. Initialiser OPEN avec M_0.
2. Pour chaque candidat M dans $OPEN$ faire
   a. Générer les successeurs de M.
   b. Pour chaque successeur M' issu de M.
      Si le successeur $M'est$ égal à M_{ref}, construire la séquence de transitions Seq de M_0 à M_{ref} et renvoyer Seq et C_{max} comme la somme du durée et risque de Seq.
      Sinon, calculer la fonction de coût f_{d+r} (M_0, M', M_{ref}) et placer M' dans une liste temporaire TEMPLIST.
   c. Si le nombre de candidats est supérieur à β_l, sélectionner ceux avec une durée inférieur à α_d ou un risque inférieur à α_r et les placer dans une liste temporaire GENERATION. Sinon, placer tous les candidats de TEMPLIST dans GENERATION en respectant :
      Si M' est égal à un marquage dans GENERATION, et si un meilleur chemin a été trouvé, mettre à jour le chemin de M_0 à M et le coût f_{d+r} (M_0, M', M_{ref}),
      Sinon, placer M' dans GENERATION.
   d. Vider TEMPLIST et retirer M de OPEN.
3. Vider OPEN.
4. Si le nombre de candidats est supérieur à β_g, sélectionner ceux avec une durée inférieure à α_d ou un risque inférieur à α_r et les placer dans OPEN. Sinon, placer tous les candidats de GENERATION dans OPEN.
5. Vider GENERATION.
6. Si OPEN est vide, terminer et retourner $Seq = \varnothing$ and $C_{max} = \infty$, sinon, retourner à l'étape 2.

En termes de complexité en mémoire, le nouveau mécanisme de filtrage réduit le nombre de candidats explorés. Ceci est dû à la stratégie utilisée pour sélectionner les candidats qui passent par deux filtres supplémentaires (α_d et α_r) par rapport aux filtres habituels (β_g et β_l) dans les algorithmes FBS et GFBS. L'algorithme GDFBS a une faible complexité en mémoire par rapport aux algorithmes FBS et GFBS et la convergence vers la référence est accélérée.

En termes de complexité en temps de calcul, notons que le nombre total de candidats explorés lors de la recherche de la trajectoire entre le marquage initial et celui de référence est également réduit par rapport aux algorithmes FBS et GFBS comme le temps nécessaire pour réaliser l'algorithme GDFBS.

5.7. Applications

Dans cette section, l'utilisation des algorithmes GFBS et GDFBS est illustrée. Ces algorithmes sont implémentés avec Matlab et fonctionnent sur un ordinateur de 1,8 GHz avec une mémoire RAM de 16 Go. Les algorithmes GFBS et GDFBS sont utilisés avec l'exemple utilisé pour la modélisation d'un FMS complexe, défini dans le chapitre 3 et reproduit sur la figure 56. Un autre FMS complexe tiré de la littérature [Giard, 2003] présenté sur la figure 57 est également introduit pour mettre en évidence l'amélioration des performances obtenues avec l'algorithme GDFBS. Notez que pour ces exemples, aucune comparaison avec la littérature n'a pu être proposée due à l'absence de travaux traitant le risque de la même manière.

Les paramètres des transitions de l'exemple défini sur la figure 56 sont donnés dans le tableau 3. Les défaillances et les réparations ne sont pas représentées sur la figure 56 pour plus de clarté mais nous supposons que les ressources r_1 à r_8 peuvent tomber en panne comme toutes les opérations du FMS. Les paramètres sont choisis comme suit : d'une part, une forte probabilité de défaillance est associée à une faible durée de récupération (défauts fréquents de faible gravité) par exemple la ressource r_4 avec $\mu_{tfr_4} = 9$ et $\mu_{trr_4} = 1$. En revanche, une faible probabilité de défaillance est associée à une longue durée de récupération (défaillances rares avec une gravité élevée) par exemple l'opération o_1 avec $\mu_{tfo_1} = 2$ et $\mu_{tro_1} = 9$. La capacité de toutes les gammes opératoires et ressources est égale à 1 : $M_0(r_k) = M_0(cp_J) = 1$, $J = 1, 2, 3$ et $k = 1, .., 8$. Le marquage initial est tel que $M_0(s_J) = 1$. Le marquage de référence est tel que $M_{ref}(s_J) = 0$ et $M_{ref}(cp_J) = M_{ref}(r_k) = 1$.

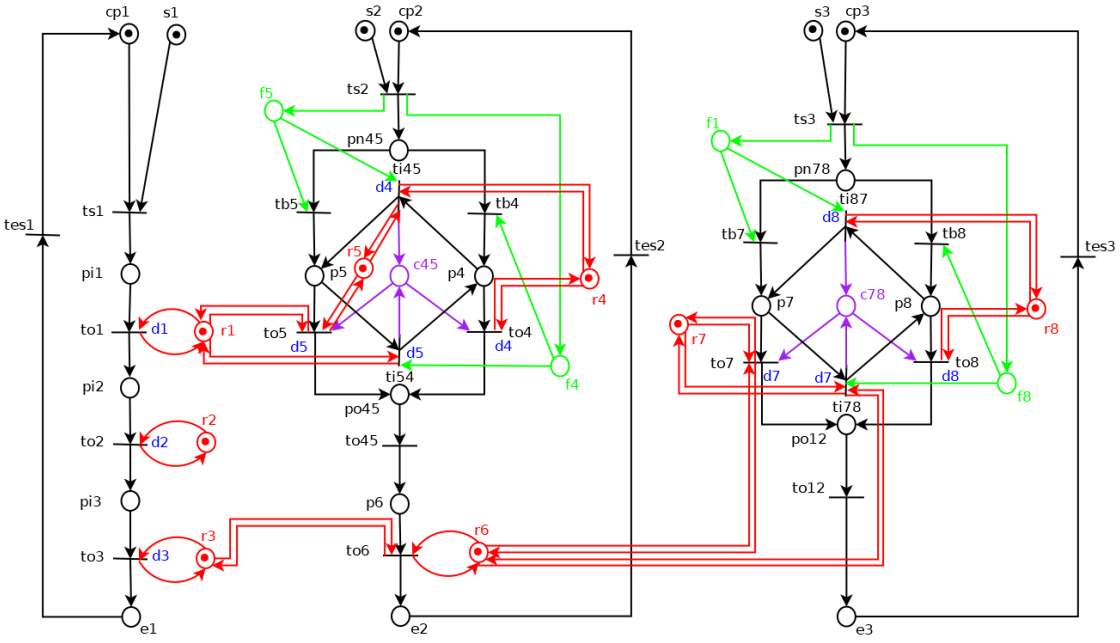


Figure 56 : Exemple d'un FMS complexe

Tableau 3 : Paramètres des opérations pour un FMS complexe

T	$D(t)$	T	$D(t)$	T	$D(t)$	T	μ	T	μ
to_1	10	tro_1	9	trr_1	10	tfo_1	2	tfr_1	3
to_2	3	tro_2	5	trr_2	8	tfo_2	8	tfr_2	2
to_3	7	tro_3	4	trr_3	7	tfo_3	9	tfr_3	1
to_4	7	tro_4	6	trr_4	1	tfo_4	1	tfr_4	9
to_5	2	tro_5	9	trr_5	9	tfo_5	2	tfr_5	3
to_6	11	tro_6	8	trr_6	7	tfo_6	2	tfr_8	2
to_7	5	tro_7	5	trr_7	2	tfo_7	10	tfr_8	8
to_8	7	tro_8	3	trr_8	4	tfo_8	7	tfr_8	10

Tableau 4 : Résultats d'ordonnancement d'un FMS complexe

Algorithme	Durée (UT)	Risque (UT)	Coût totale	Candidats générés	Temps de calcul CPU(s)	Durée moyenne de simulations
FBS-D*	31	32.1	63.1	589	20	64.3
FBS-DR**	33	28.4	61.4	582	20	62.9
GFBS-D*	27	38.0	65.0	465	16	68.2
GFBS-DR**	**30**	**24.7**	**54.7**	465	16	59.6
GDFBS	**31**	**21.9**	**52.9**	**339**	**12**	**57.1**

* D : exploration des candidats sur la base de durée.

** DR : exploration des candidats sur la base de durée + risque.

Les résultats sont obtenus en utilisant les paramètres $\beta g = 10$ et $\beta_l = 5$, pour tous les algorithmes et sont donnés dans le tableau 4. Dans le premier test utilisant FBS-D, les candidats de durée la plus faible sont explorés. Ensuite, le risque de la séquence obtenue est calculé. L'utilisation de la fonction de coût avec durée et risque (FBS-DR) améliore les résultats grâce à la sélection des candidats à explorer qui prend en compte le coût additionnel dû au risque. De même en comparant les algorithmes GFBS-D et GFBS-DR, malgré le même nombre de candidats explorés, le gain en durée est de 10% en faveur de l'algorithme GFBS-D (durée de 27 UT avec l'algorithme GFBS-D contre 30 UT pour l'algorithme GFBS-DR) mais le gain en risque est important (37%) en faveur de GFBS-DR (24.7 UT avec l'algorithme GFBS-DR contre 38.0 UT avec l'algorithme GFBS-D). En conséquence, le coût total est amélioré grâce à l'algorithme GFBS-DR par rapport à l'algorithme GFBS-D, le gain du coût total est de 17%.

En ne tenant compte que des algorithmes qui prennent en compte la durée et le risque pour la sélection des candidats, l'algorithme GDFBS renvoie une solution avec un meilleur coût et moins de candidats

explorés. La comparaison avec les algorithmes FBS-DR, GFBS-DR et GDFBS (dans lesquels le coût total comprend déjà le risque) pourrait être faite en fonction de la complexité en mémoire et en temps de calcul. En termes de complexité en mémoire, moins de candidats sont sélectionnés pour une exploration future à chaque itération de l'algorithme GDFBS par rapport aux algorithmes FBS-DR et GFBS puisque la sélection des candidats passe par les deux filtres supplémentaires α_d et α_r. En conséquence, la somme de tous les candidats conservés lors de la recherche avec l'algorithme GDFBS (339 candidats pour cet exemple) est plus petite par rapport aux algorithmes FBS-DR (589 candidats) et GFBS-DR (465 candidats). Le gain en complexité de mémoire pour cet exemple est important (28%) par rapport au GFBS-DR et encore plus important (43%) par rapport aux algorithmes FBS-DR.

Cette diminution du nombre total de candidats explorés lors de la recherche accélère la convergence vers la référence par rapport aux algorithmes FBS-DR et GFBS-DR. Elle se traduit par une diminution du temps de calcul avec l'algorithme GDFBS (12) de 25% par rapport à l'algorithme GFBS-DR (16 secondes) et de 40% par rapport à l'algorithme FBS-DR (20 secondes).

En conclusion, l'algorithme GDFBS renvoie une meilleure solution à moindre coût. Le gain en coût est limité pour cet exemple (4%) par rapport à l'algorithme GFBS-DR mais le gain est plus important (15%) par rapport à l'algorithme FBS-DR. Ce gain est dû à la stratégie de sélection : les candidats présentant à la fois une grande durée et un risque élevé ne sont pas conservés ce qui améliore la qualité des candidats sélectionnés et, par conséquent, l'ordonnancement obtenu. Cet exemple montre que les méthodes proposées conviennent aux problèmes d'ordonnancement dans les ateliers hybrides.

Les résultats de la simulation confirment ces observations. En effet, la simulation de l'ordonnancement généré par l'algorithme GDFBS retourne la meilleure durée totale moyenne des 10000 simulations obtenues en tenant compte du risque de pannes avec 57.1 UT. Ce résultat est amélioré de 5% par rapport à l'ordonnancement généré par GFBS-DR (durée totale moyenne égale à 59.6 UT), 16% par rapport à l'ordonnancement généré par GFBS-D (durée totale moyenne égale à 68.2 UT), 10% par rapport à l'ordonnancement généré par FBS-DR (durée totale moyenne égale à 62.9 UT) et 12% par rapport à l'ordonnancement généré par FBS-D (durée totale moyenne égale à 64.3 UT).

Un autre FMS, composé de 8 jobs J_1, J_2, J_3, J_4, J_5, J_6, J_7 et J_8, est considéré [Giard, 2003]. Le modèle T-TPN est représenté sur la figure 57. Les défaillances et réparations ne sont pas représentées pour plus de clarté mais nous supposons que les ressources r_1 à r_6 peuvent tomber en panne comme toutes

les opérations du FMS. Les détails sont présentés dans le tableau 5. Avec les paramètres β_g = 10 et β_l = 5, les résultats obtenus sont donnés dans le tableau 6.

Tableau 5 : Paramètres des opérations d'un FMS complexe (Giard, 2003)

D(t)	*T*	*μ*	*T*
1	to_{33}, tro_{33}, to_{71}, tro_{71}	1	tfo_{25}, tfo_{32}, tfo_{75}
2	to_{13}, tro_{13}, to_{22}, tro_{22}, to_{24}, tro_{24}, to_{64}, tro_{64}, to_{73}, tro_{73}, trr_2	2	tfo_{12}, tfo_{21}, tfo_{23}, tfo_{41}, tfo_{44}, tfo_{51}, tfo_{61}, tfo_{62}, tfo_{63}, tfo_{65}, tfo_{66}, tfo_{72}, tfo_{74}, tfo_{81}, tfo_{82}, tfo_{85}, tfr_1, tfr_2, tfr_3, tfr_5
3	to_{61}, tro_{61}, to_{62}, tro_{62}, to_{74}, tro_{74}, to_{84}, tro_{84}, trr_1	3	tfo_{11}, tfo_{31}, tfo_{52}, tfo_{53}, tfo_{67}, tfo_{83}
4	to_{31}, tro_{31}, to_{42}, tro_{42}, to_{43}, tro_{43}, to_{52}, tro_{52}, to_{65}, tro_{65}, to_{66}, tro_{66}, to_{72}, tro_{72}	4	tfo_{42}, tfo_{64}
5	to_{11}, tro_{11}, to_{53}, tro_{53}, to_{67}, tro_{67}, to_{81}, tro_{81}, to_{82}, tro_{82}, trr_5	5	tfo_{13}, tfo_{22}, tfo_{24}, tfo_{43}, tfo_{84}
6	to_{12}, tro_{12}, to_{21}, tro_{21}, to_{23}, tro_{23}, to_{51}, tro_{51}, to_{63}, tro_{63}, to_{83}, tro_{83}, to_{85}, tro_{85}	6	tfo_{73}
7	to_{25}, tro_{25}, to_{44}, tro_{44}	7	tfo_{33}, tfo_{71}
8	to_{75}, tro_{75}	8	tfr_4
10	to_{32}, tro_{32}, to_{41}, tro_{41}, trr_4		
20	trr_3		

Tableau 6 : Résultat d'ordonnancement d'un FMS complexe (Giard, 2003)

Algorithme	Durée (UT)	Risque (UT)	Coût totale	Candidats générés	Temps de calcul CPU(s)	Durée moyenne de simulations
FBS-D*	72	72.6	144.6	3419	27	158.3
FBS-DR**	72	67.9	139.9	3381	27	149.8
GFBS-D*	56	61.9	117.9	3028	19	129.5
GFBS-DR**	**58**	**55.9**	**113.9**	3030	19	124.5
GDFBS	**57**	**54.9**	**111.9**	**2830**	**15**	**119.2**

* D : exploration des candidats sur la base de durée.

** DR : exploration des candidats sur la base de durée + risque.

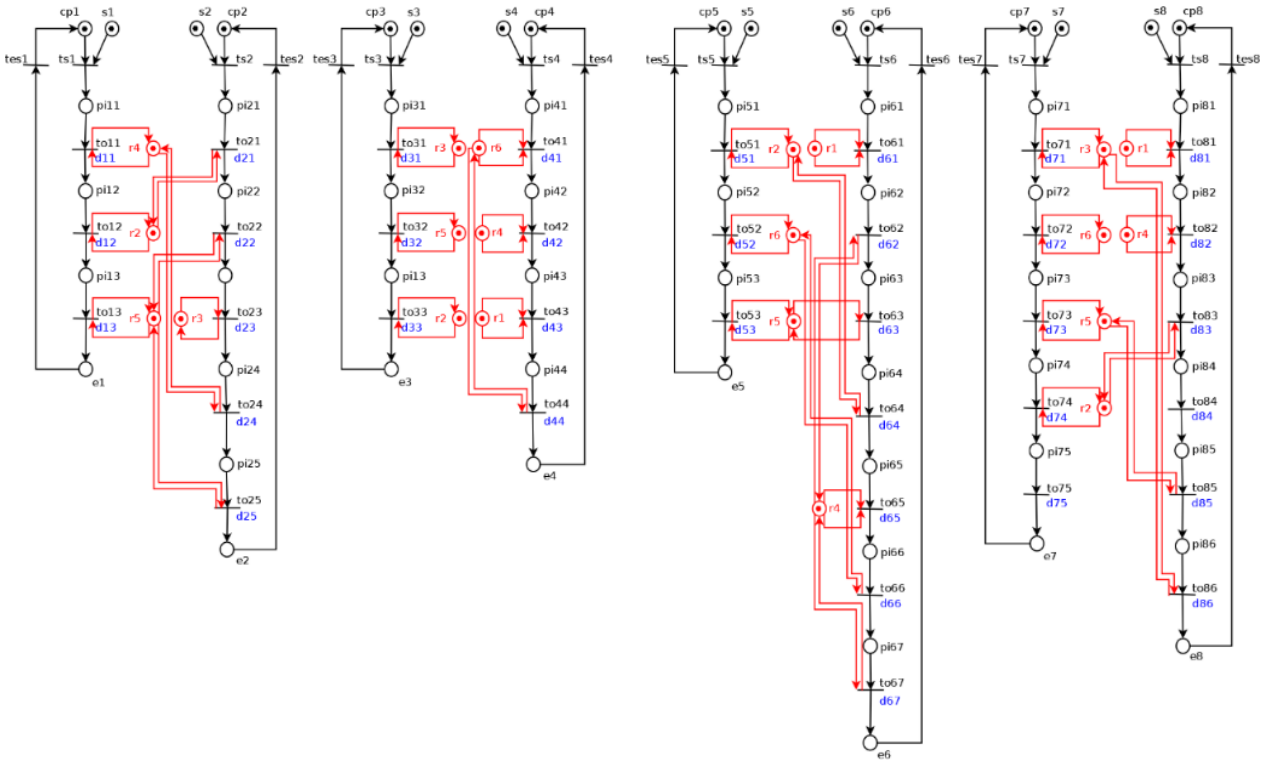


Figure 57 : Exemple d'un FMS complexe (Giard, 2003)

Pour cet exemple, nous commençons par la comparaison entre les algorithmes GFBS-D et GFBS-DR. Malgré le fait que le nombre de candidats explorés soit quasiment le même pour les deux algorithmes, la sélection des candidats avec l'algorithme GFBS-D, ne prenant en compte que la durée, conduit à un ordonnancement de meilleure durée théorique par rapport à l'algorithme GFBS-DR qui prend en compte la durée et le risque. La durée est légèrement dégradée de 4% lors de l'utilisation de l'algorithme GFBS-DR (58 UT) contre une durée de 56 UT avec l'algorithme GFBS-D. En revanche, malgré le gain en durée, le risque de l'ordonnancement obtenu avec l'algorithme GFBS-D dépasse largement le risque obtenu avec l'algorithme GFBS-DR. Le risque est significativement amélioré de 10% pour GFBS_DR (55,9 UT avec l'algorithme GFBS-DR) contre 61,9 UT avec l'algorithme GFBS-D. Ainsi le coût total est amélioré de 4% en utilisant l'algorithme GFBS-DR par rapport à l'algorithme GFBS-D : l'algorithme GFBS-DR génère un ordonnancement avec un coût total égal à 113,9 UT tandis que l'algorithme GFBS-D génère un ordonnancement avec un coût total égal à 117,9 UT.

En comparant l'algorithme GDFBS avec d'autres algorithmes prenant en compte la durée et le risque lors de la sélection des candidats, le gain en complexité de mémoire est en faveur de l'algorithme GDFBS, avec 2830 candidats explorés : le nombre de candidats est amélioré de 7% par rapport à

l'algorithme GFBS-DR (3030 candidats) et de 17% par rapport à l'algorithme FBS-DR (3381 candidats). Du fait de l'accélération de la convergence vers la référence, le temps de calcul est réduit de 22% par rapport à l'algorithme GFBS-DR (19 secondes) et de 45% par rapport à l'algorithme FBS-DR (27 secondes). De plus, on observe un léger gain en coût total (2%) par rapport au GFBS-DR, et un gain plus important (21%) par rapport à l'algorithme FBS-DR. Ce gain est dû non seulement à la réduction de la durée (2% par rapport au GFBS-DR et 21% par rapport aux algorithmes FBS-DR) mais aussi à la réduction du risque de déviation (2% par rapport au GFBS -DR et 19% par rapport aux algorithmes FBS-DR). La stratégie de sélection de l'algorithme GDFBS, basée sur des filtres sur la durée et le risque, améliore les performances de l'ordonnancement obtenu avec une plus petite durée et réduit le risque de déviation par rapport à l'ordonnancement attendu.

Les séquences générées par les différents algorithmes sont simulés (10000 simulations) en utilisant les mêmes paramètres utilisés pour calculer ces ordonnancements (Tableau 5). Les résultats montrent que la séquence générée par l'algorithme GDFBS génère la meilleure durée totale moyenne égale à 119.2 UT. En ce qui concerne la simulation des ordonnancements obtenus par les algorithmes GFBS, les résultats sont les suivants : une durée totale moyenne égale à 124.5 UT pour la simulation de la séquence générée par GFBS-DR (dégradation de 5% par rapport à l'algorithme GDFBS) suivie par une durée totale moyenne égale à 129.5 UT (dégradation plus importante égale à 8% par rapport à l'algorithme GDFBS). Pour les algorithmes FBS, la durée totale moyenne des simulations générées par FBS-DR est égale à 149.8 UT avec une importante dégradation qui atteint 21% et une durée totale moyenne égale à 158.3 UT pour l'algorithme FBS-D présentant la plus grande dégradation par rapport à l'algorithme GDFBS égale à 25%.

Cet exemple montre que les méthodes proposées sont adaptées à des problèmes d'ordonnancement sur les ateliers avec plusieurs gammes opératoires. L'algorithme GDFBS génère non seulement moins de candidats mais améliore également la sélection des candidats ce qui conduit à converger plus rapidement vers la cible et renvoie une meilleure solution à moindre coût.

5.8. Conclusion

Dans ce chapitre, les problèmes de modélisation et d'ordonnancement dans un environnement incertain pour un FMS complexe ont été étudiés. Tout d'abord, la méthodologie de modélisation systématique a été développé afin qui prendre en compte les incertitudes dans la modélisation. Par la suite, la fonction coût proposée, qui tient compte seulement de la durée, a été modifiée pour évaluer non seulement la durée mais aussi le risque dans la prise de décision de candidats à sélectionner. Enfin, une nouvelle méthode d'ordonnancement -Generation Double Filtered Beam Search- a été

présentée pour réaliser un compromis entre durée et risque, et pour accélérer la convergence de la méthode vers la référence. Des exemples ont été détaillés afin d'illustrer que l'algorithme GDFBS convient à une grande variété d'organisations d'ateliers flexibles, y compris les ateliers job shop, les ateliers open shop et les ateliers hybrides.

Chapitre 6 : Conclusion générale

6.1 Conclusions

La résolution des problèmes d'ordonnancement dans les ateliers complexes constitue la principale contribution des travaux de cette thèse. L'objectif de notre travail était, tout particulièrement, l'étude des problèmes d'ordonnancement pour un type d'ateliers complexes de l'industrie 4.0, dans lesquels les opérations sont traitées avec des contraintes de précédence partielles, et en présence d'incertitudes.

Dans le deuxième chapitre de cette thèse, nous avons présenté une formalisation des problèmes d'ordonnancement. Une description des tâches, des ressources, des contraintes et des critères, a été exposée. Les différents types d'ateliers usuels (flow shop, job shop et open shop) ont été présentés : un atelier est défini par l'organisation de ses tâches. Un problème important concerne la satisfaction des contraintes de précédence. Dans les ateliers flow shop et job shop ces contraintes empêchent toute flexibilité. Dans les open shop, à l'inverse, la flexibilité totale sur les opérations n'est pas très réaliste. Afin d'étendre les organisations usuelles d'ateliers, de surmonter les limites liées au contraintes de précédence et surtout de répondre aux attentes de la production intelligente de l'industrie 4.0, une nouvelle organisation d'ateliers a été proposée : l'idée a été de privilégier la flexibilité sur les opérations autant que possible et d'appliquer les contraintes de précédence là où elles sont nécessaires. L'objectif est de pouvoir profiter de cette flexibilité pour améliorer la gestion des ressources partagées. Cela permet d'avoir un impact positif sur le résultat de l'ordonnancement. Les ateliers munis de cette nouvelle organisation sont appelés ateliers complexes.

Par la suite, les méthodes de résolution, exactes et approchées sont introduites. Nous avons également discuté de leur complexité afin de montrer l'intérêt des chercheurs pour les problèmes d'ordonnancement et les différentes techniques disponibles à ce jour pour résoudre ces problèmes. Les approches exactes trouvent la solution optimale mais ne peuvent être utilisées uniquement pour des problèmes de petite taille. Par contre, même si elles ne garantissent pas l'optimalité de la solution, les approches approchées sont de plus en plus utilisées et appréciées puisqu'elles permettent de résoudre les problèmes d'ordonnancement de taille plus grande.

Le troisième chapitre a été consacré à la modélisation des problèmes d'ordonnancement. Cette modélisation a été élaborée en utilisant les réseaux de Petri. Dans ce travail, une nouvelle méthode a été proposée afin de permettre la modélisation des ateliers complexes. L'idée est de permettre la modélisation de l'organisation de l'atelier systématiquement sur la base de la structuration hiérarchique, à plusieurs niveaux, des opérations. Dans ce travail, le problème de modélisation a été considéré comme étant un ensemble de sous-problèmes : le modèle complet est obtenu à partir de sous modèles construits à différents niveaux. Pour un atelier de l'industrie 4.0, nous pouvons définir

pour chaque ensemble d'opérations les contraintes et les critères permettant de répondre à un besoin spécifique. Par la suite, ces sous modèles, qui répondent à un ensemble de besoins spécifiques, sont regroupés par niveau et les contraintes sont propagées à travers les niveaux pour répondre finalement à un besoin global représentant le produit personnalisé attendu.

Pour conclure, cette stratégie systématique de modélisation a permis non seulement de s'adapter à plusieurs problèmes d'atelier de production mais aussi de créer un système personnalisé permettant de répondre aux attentes des consommateurs du 21ème siècle.

Le quatrième chapitre a été consacré au calcul d'ordonnancements pour les ateliers complexes. Afin de résoudre ce problème, notre choix s'est porté sur l'utilisation d'une méthode approchée pour surmonter le problème d'explosion combinatoire lié aux méthodes exactes. Une nouvelle variante de recherche en faisceau appelée GFBS a été proposée pour améliorer la sélection des candidats à explorer : cette amélioration consiste en un équilibre entre la largeur et la profondeur de la recherche au lieu de favoriser uniquement la profondeur avec le risque de découvrir a posteriori que le choix effectué était médiocre (ce qui ce passe avec FBS). La recherche d'ordonnancement a été basée sur une nouvelle fonction coût définie pour permettre de prendre en compte la stratégie itérée de modélisation, selon plusieurs niveaux. Cette fonction coût a été prouvée être une borne inférieure de la durée réelle.

Le dernier chapitre a été consacré à l'étude des problèmes d'ordonnancement dans un environnement incertain. Dans le contexte de l'industrie 4.0 où les ateliers sont soumis à des contraintes de personnalisation de production, la prise en compte de l'impact des perturbations qui peuvent affecter le système est primordiale. Dans ce travail, nous avons introduit un paramètre permettant d'aider à la prise de décision : ce paramètre est une évaluation du risque de perturbations pouvant intervenir lors de la production. Cette approche permet au décideur d'avoir une idée de l'impact des éventuelles perturbations qui peuvent intervenir et donc prendre les décisions en tenant compte de ces perturbations.

Par la suite, nous avons proposé une approche permettant de résoudre le problème d'ordonnancement en présence des incertitudes par la recherche d'une solution avec un compromis entre la durée et le risque de façon à filtrer les candidats et d'éliminer ceux qui conduisent à une grande durée et à un grand risque. Cette approche utilise la nouvelle variante de recherche en faisceau GDFBS.

L'algorithme GDFBS a permis d'améliorer les performances par rapport à d'autres variantes de recherche en faisceau. En effet, nous avons montré que l'algorithme GDFBS réduit la complexité en mémoire comparativement aux algorithmes FBS et GFBS en explorant moins de candidats pour

trouver une solution. Par conséquent, la complexité en temps de calcul est réduite puisque le nombre de candidats explorés est inférieur à celui des algorithmes FBS et GFBS.

6.2 Perspectives

6.2.1 A court terme

Au terme des travaux de thèse, plusieurs pistes intéressantes peuvent être considérées à court terme.

Nous savons que les performances des méthodes heuristiques dépendent en premier lieu de la pertinence de la fonction objectif et tout particulièrement de la partie estimée de cette fonction. Une amélioration de la fonction coût devra donc être étudiée pour affiner l'estimation de la durée restante et du risque afin d'être plus proche du coût réel. En particulier, les retards dus à l'utilisation des ressources partagées devront être étudiés. Cela permettra d'améliorer la qualité des candidats sélectionnés et donc la solution obtenue.

Dans ce travail, les bornes sur la durée et le risque, utilisées pour la sélection des candidats à explorer avec l'algorithme GDFBS, sont définies arbitrairement. Une amélioration de la définition de ces bornes peut être envisagée dans nos travaux futurs, en particulier avec un appel récursif de l'algorithme GDFBS en utilisant le résultat du premier appel comme une borne supérieure, pour la durée ainsi que le risque, de l'appel suivant. Le principe d'une telle stratégie serait de garantir qu'aucun candidat avec un coût, qui dépasse la borne supérieure, ne soit sélectionné pour l'exploration. La borne supérieure sera flexible et dès qu'un meilleur résultat sera obtenu, elle sera mise à jour. La qualité des candidats sélectionnés sera donc améliorée au fur et à mesure de l'appel récursif. Trouver un compromis entre la condition d'arrêt et la complexité de l'algorithme sera également nécessaire. Une solution pourrait être de provoquer l'arrêt de la récursion si le résultat obtenu n'est pas amélioré par rapport au résultat de l'appel précédant.

Dans ce travail, nous avons évalué le risque pour le prendre en compte dans la prise de décision sans considérer explicitement la question de la robustesse. Cependant, la robustesse peut être quantifiée, comme étant une performance à évaluer afin de mesurer la capacité d'un ordonnancement donné pour faire face à l'impact des perturbations. Cette évaluation interviendra donc lors de la sélection des candidats pour poursuivre l'exploration.

Une autre amélioration liée au modèle PN peut être introduite en définissant des marquages dangereux, par exemple des marquages à partir desquels il n'existe aucune trajectoire commandable vers la référence (blocages) ou des marquages qui permettent le tir de transitions non commandables.

Cela nous permettrait d'évaluer la robustesse de la trajectoire résiduelle vers la référence et donc de privilégier les candidats plus robustes que d'autres.

Quantifier la qualité de solutions obtenues avec les méthodes approchées permettra de donner une idée sur la performance réelle de ces méthodes. Pour les algorithmes GFBS et GDFBS, nous pouvons qualifier l'algorithme en étudiant la dispersion des coûts des solutions obtenues avec différents couple de paramètres (β_g, β_l) et vérifier si l'ensemble des solutions conduit à des coûts voisins. Dans ce cas, nous pourrions conclure que l'algorithme est performant. A l'inverse, dans le cas où la dispersion des coûts est grande, nous pourrions conclure que l'algorithme n'est pas performant.

6.2.2 A plus long terme

Enfin, les travaux de recherches présentés dans cette thèse permettent de mettre en évidence des perspectives de recherche à long terme. L'hybridation de la recherche en faisceau avec d'autres méta-heuristiques peut être envisagée pour résoudre les problèmes d'ordonnancement. L'idée est de chercher un ordonnancement avec la recherche en faisceau et de l'utiliser comme solution de départ à améliorer.

Une autre extension de nos travaux consisterait à étudier le problème des incertitudes et de robustesse sur la base des types de défaillances qui peuvent affecter le système. En effet, les accidents sont souvent le résultat de multiples séquences de défaillances qui impliquent le matériel, les logiciels et les erreurs humaines. L'idée est d'évaluer le résultat d'une défaillance pouvant mettre en danger par exemple la sécurité des personnels ou des biens. L'objectif est de permettre au décideur de choisir par exemple une trajectoire avec des incertitudes qui ne présentent pas de réels dangers (mais un grand retard) au lieu de choisir une trajectoire avec des incertitudes associées à un grand danger (mais un petit retard).

Une dernière perspective à long terme serait d'intégrer notre approche dans un contexte réel d'industrie 4.0 afin de pouvoir créer un système personnalisé pour résoudre le problème d'ordonnancement et le paramétrage d'opérations et de ressources en tenant compte des perturbations.

Les travaux présentés dans cette thèse ont donné lieu à plusieurs publications et ont été conclus par l'acceptation d'un article de revue [Cherif, Leclercq & Lefebvre, 2021a], la soumission d'un deuxième article de revue [Cherif, Leclercq & Lefebvre, 2021b], ainsi que des communications à des conférences internationales [Cherif, Leclercq & Lefebvre, 2018 ; Cherif, Leclercq & Lefebvre, 2019a ; Cherif, Leclercq & Lefebvre, 2019b].

Bibliographie

[Aggoune, 2002] Aggoune, R., (2002), «Ordonnancement d'ateliers sous contraintes de disponibilité des machines», Thèse de Doctorat, Metz.

[Allaire, 2005] Allaire, G., (2005), «*Analyse numérique et optimisation : Une introduction à la modélisation mathématique et à la simulation numérique*», Mathématiques appliquées, École Polytechnique.

[Al-Hinai & Elmekkawy, 2011] Al-Hinai, N., & Elmekkawy, T. Y., (2011), «Robust and Stable Flexible job Shop Scheduling with Random Machine Breakdowns Using a Hybrid Genetic Algorithm», *International Journal of Production Economics*, vol. 132, no. 2, pp. 279–281.

[Ammour et al., 2015] Ammour, R., Leclercq, E., Sanlaville, E., & Lefebvre, D., (2015), «Estimation of the fault occurrence dates in DESs with partially observed stochastic Petri nets», 5th IFAC-Conference on Analysis and Design of Hybrid Systems (ADHS), Atlanta, USA, pp. 254–259.

[Andresen et al., 2008] Andresen, M., Bräsel, H., Mörig, M., Tusch, J., Werner, F., & Wil-lenius, P., (2008), «Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop», *Mathematical and Computer Modelling*, vol. 48, pp. 1279-1293.

[Asadzadeh & Zamanifar, 2010] Asadzadeh, L., & Zamanifar, K., (2010), «An agent-based parallel approach for the job shop scheduling problem with genetic algorithms», *Mathematical and Computer Modelling*, vol. 52, pp. 11-12.

[Baptiste & Le Pape, 1996] Baptiste, P., & Le Pape, C., (1996), «A constraint-Based Branch and Bound Algorithm for Preemptive Job-Shop Scheduling», 5th IEEE International Symposium on Assembly and Task Planning, pp. 263–287, Belgium.

[Barichard, 2003] Barichard, V., (2003), «Approches hybrides pour les problèmes multi-objectifs», Thèse de Doctorat, Université d'Angers.

[Barkaoui & Ben Abdallah, 1996] Barkaoui, K., & Ben Abdallah, I., (1996), «Analysis of a resource allocation problem in FMS using structure theory of Petri nets», Proceedings, First International Workshop on Manufacturing and Petri Nets, pp. 1-15, Osaka, Japan.

[Baruwa & Piera, 2014] Baruwa, O. T., & Piera, M. A., (2014), «Anytime heuristic search for scheduling flexible manufacturing systems: a timed colored Petri net approach». *International Journal of Advanced Manufacturing Technology*, vol. 75, no. 1-4, pp. 123-137.

[Baruwa, Piera & Guasch, 2015] Baruwa, O. T., Piera, M. A., & Guasch, A., (2015), «Deadlock-free scheduling method for flexible manufacturing systems based on timed colored Petri nets and anytime heuristic search», *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 5, pp. 831-846.

[Baruwa & Piera, 2016] Baruwa, O. T., & Piera, M. A., (2016), «A Coloured Petri net-Based Hybrid Heuristic Search Approach to Simultaneous Scheduling of Machines and Automated Guided Vehicles», *International Journal of Production Research*, vol. 54, no. 16, pp. 4773–4792.

[Bel, 2001] Bel, G., & Cavaillé, J. B., (2001), «*Ordonnancement de la production*», Editions Hermès, Paris.

[Bellman, 1986] Bellman, R. E., (1986), «*The Bellman continuum*», Editions Robert S. Roth, Philadelphia, USA.

[Berthomieu & Vernadat, 2003] Berthomieu, B., & Vernadat, F., (2003), «State class constructions for branching analysis of time Petri nets», *In Tools and Algorithms for the Construction and Analysis of Systems, Springer*, pp. 442–457.

[Blazewicz et al., 1996] Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Weglarz, J., (1996), «*Scheduling computer and manufacturing process*», Springer, Berlin.

[Blum et al., 2005] Blum, C., Roli, A., & Alba, E., (2005), «An introduction to metaheuristic techniques», *Parallel Metaheuristics : A New Class of Algorithms*, pp. 1-47.

[Bonnans et al., 2006] Bonnans, J. F., JGilbert, . Ch., Lemaréchal, C., & Sagastizábal, C., (2006), «*Numerical Optimization Theoretical and Practical Aspects*», Springer.

[Brucker, 2007] P. Brucker, (2007), « *Scheduling Algorithms* », Springer.

[Cabasino, Giua & Seatzu, 2009] Cabasino, M. P., Giua, A., & Seatzu, C., (2009), «Diagnosability of bounded Petri nets», in Proc. 48th IEEE Conf. Decision and Control and 28th Chinese Control Conference, pp. 1254-1260, Shanghai, China.

[Campos et al., 1989] Campos, J., Chiola, G., Colom, J. M., & Silva, M., (1989), «Tight Polynomial Bounds for Steady-State Performance of Marked Graphs», Workshop on Petri Nets and Performance Models PNPM, pp. 200-209, Kyoto, Japan.

[Camus, 1997] Camus, H., (1997) «Conduite des systèmes flexibles de production manufacturière par composition des régimes permanents cycliques : modélisation et évaluation des performances à l'aide des réseaux de Petri». Thèse de Doctorat, Université de Lille 1.

[Carlier & Rebaï, 1996] Carlier, J., & Rebaï, I., (1996), «Two branch and bound algorithms for the permutation flow shop problem», *European Journal of Operational Research*, vol. 90, no. 2, pp. 238-251.

[Carlier & Chrétienne, 1988] Carlier, J., & Chrétienne, P., (1988), «*Problèmes d'ordonnancement, Modélisation, Complexité, Algorithmes*», Edition Masson, Paris.

[Cassandras & Lafortune, 2009] Cassandras, C., & Lafortune, S., (2009), «*Introduction to discrete event systems*». Springer Science & Business Media.

[Cassandras, 1993] Cassandras, C., (1993), «*Discrete Event Systems: Modeling and Performances Analysis*», Homewood, IL, USA: Aksen Assoc. Inc. Pub.

[Cherif, Leclercq & Lefebvre, 2021a] Cherif, G., Leclercq, E., & Lefebvre, D. (2021). «Scheduling problems for a class of hybrid FMS using T-TPN and Beam Search», *Journal of Control, Automation and Electrical Systems (JCAE)*, Accepté (03/02/2021).

[Cherif, Leclercq & Lefebvre, 2021b] Cherif, G., Leclercq, E., & Lefebvre, D. (2021). «Hybrid FMS scheduling using transition-timed Petri nets and a new variant of Beam Search in an uncertain environment», *Journal of Intelligent Manufacturing (JIMS)*, Soumis (3ème révision).

[Cherif, Leclercq & Lefebvre, 2019a] Cherif, G., Leclercq, E., & Lefebvre, D. (2019a). «Generation Filtered Beam Search algorithm for the scheduling of hybrid FMS using T-TPN», IEEE 18th European Control Conference (ECC), pp. 3225-3230, Naples, Italy.

[Cherif, Leclercq & Lefebvre, 2019b] Cherif, G., Leclercq, E., & Lefebvre, D., (2019b), «Hybrid FMS scheduling using T-TPN and Beam Search in uncertain environments», IEEE 45th Annual Conference of the Industrial Electronics Society, pp. 405-410, Lisbon, Portugal.

[Cherif, Leclercq & Lefebvre, 2018] Cherif, G., Leclercq, E., & Lefebvre, D., (2018), «Modeling hybrid manufacturing systems using T-TPN with buffers», IEEE 23rd International Conference on Emerging Technologies and Factory Automation, vol. 1, pp. 480-587, Turin, Italy.

[Collette & Siarry, 2004] Collette, Y., & Siarry., P., (2004), «*Multiobjective Optimization: Principles and Case Studies*», Springer, Berlin.

[Collette & Siarry, 2002] Collette, Y., & Siarry, P., (2002), «*Optimisation Multiobjectif*», Editions Eyrolles, Paris.

[Colorni et al., 1992] Colorni, A., Dorigo, M., & Maniezzo, V. (1992), «*An Investigation of some Properties of an Ant Algorithm*», In Reinhard Männer & Bernard Manderick, ed., 'PPSN', Elsevier, pp. 515-526

[Colorni et al., 1991] Colorni, A., Dorigo, M., & Maniezzo, V. (1991), «Distributed Optimization by Ant Colonies», In: Varela, F. and Bourgine, P., Eds., *Proceedings of the European Conference on Artificial Life*, ECAL'91, Paris, Elsevier Publishing, pp. 134-142, Amsterdam.

[Cook, 1971] Cook, S.A., (1971), «The complexity of theorem proving procedures», Association of Computing Machinery, *Proceedings of the third annual ACM Symposium on the Theory of Computing*, pp. 151-158, United States.

[Cormen et al., 2001] Cormen, T., Leiserson, C., Rivest, R., & Stein, C., (2001), «*Introduction à l'algorithmique*», DUNOD, 3e éd. (1re éd. 1990).

[Croce, Ghirardi & Tadei, 2004] Croce, F. D., Ghirardi, M., & Tadei, R., (2004), «Recovering Beam Search: Enhancing the Beam Search Approach for Combinatorial Optimization Problems», *Journal of Heuristics*, vol. 10, no. 1, pp. 89-104.

[Dantzig et al., 1954] Dantzig, G., Fulkerson, R., & Selmer, (1954), «Solution of a large-scale traveling-salesman problem», *Journal of the operations research society of America*, vol. 2, no. 4, pp. 393–410.

[David & Alla, 2010] David, R., & Alla, H., (2010), «*Discrete, Continuous, and Hybrid Petri nets*», Springer Science & Business Media.

[David & Alla, 1992] David, R., & Alla, H., (1992), «*Petri Nets and Grafcet : tools for modeling discrete event systems*», Prentice-Hall, London.

[David & Alla, 2001] David, R., & Alla, H., (2001), «On hybrid Petri Nets», *Discrete event systems: Theory and applications*, vol. 11, pp. 9-40.

[Dezani et al., 2014] Dezani, H., Bassi, R. D. S., Marranghello, N., Gomes, L., Damiani, F., & Nunes da Silva, I., (2014), «Optimizing urban traffic flow using Genetic Algorithm with Petri net analysis as fitness function», *Neurocomputing*, vol. 124, pp. 162–167.

[Diaz, 2003] Diaz, M., (2003), «*Vérification et mise en œuvre des réseaux de Petri*», TraitéIC2. Hermès Science, Paris.

[Eksioglu et al., 2008] Eksioglu, B., Duni Eksioglu, S., & Jain, P., (2008), «A tabu search algorithm for the flowshop scheduling problem with changing neighborhoods», *Computers & Industrial Engineering*, vol. 54, pp. 1-11.

[Ezpeleta et al., 1995] Ezpeleta, J., Colom, J.M., & Martinez, J., (1995), «A Petri Net based deadlock prevention policy for Flexible Manufacturing Systems», *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 173-184.

[Gagnié, 2001] Gagnié, C., Gravel, M., & Price, W.L., (2001). «Optimisation par colonies de fourmis pour un problème d'ordonnancement industriel avec temps de réglage dépendant de la séquence», 3ème Conférence Francophone de MOdélisation et,de SIMulation, MOSIM'01, France.

[Gargouri, 2003] Gargouri, E., (2003), «Ordonnancement coopératif en industries agroalimentaires», Thèse de Doctorat, Université des Sciences et Technologies de Lille 1.

[Giard, 2003] Giard, V., (2003), «*Gestion de la production et des flux*», Economica.

[Glover, 1989] Glover, F., (1989), «Tabu search, part I. ORSA», *Journal of Computing*, vol. 1, pp. 190-206.

[Glover, 1990] Glover, F., (1990), «Tabu search, part II. ORSA», *Journal of Computing*, vol. 2, pp. 4-32.

[Gotha, 1993] Gotha, (1993), «Les problèmes d'ordonnancements», *RAIRO-Recherche Opérationnelle*, vol. 27, pp. 77-150.

[Graham et al., 1979] Graham, R. L., Lawler, E. L., Lentra, J. K., & Rinno oy-Kan, A. H. G., (1979), «Optimization and approximation in deterministic sequencing and scheduling: A survey», *Annals of Discrete Mathematics*, vol. 5, pp. 287-326.

[Han et al., 2015] Han, L., Xing, K., Chen, X., & Xiong, F., (2015), «A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems», *Journal of Intelligent Manufacturing*, vol. 29, pp. 1083–1096.

[Hart, Nilsson & Raphael, 1968] Hart, P. E., Nilsson, N. J., & Raphael, B., (1968), «A Formal Basis for the Heuristic Determination of Minimum Cost Paths», *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107.

[Hervé Hillion, 1989] Hervé Hillion, M., (1989), «Modélisation et analyse des systèmes de production discrets par par les réseaux de Petri». Thèse de Doctorat, Université Pierre et Marie Curie Paris VI.

[Himmiche et al., 2018] Himmiche, S., Marangé, P., Aubry, A., & Pétin, J.-F. (2018), "Robust production scheduling under machine failures - A DES based evaluation approach", *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 271-276.

[Holland, 1975] Holland, J.H., (1975), «Adaptation in natural and artificial systems», Thèse de Doctorat, Thesis Michigan Press University, Ann Arbor, Michigan.

[Holmström & Aavikko, 1994] Holmström, J., & Aavikko, P., (1994), «Achieving a Management Breakthrough in Inbound Logistics by Improving the Efficacy of Operational Decisions», *Production and Inventory Management Journal*, (Third Quarter), pp. 1-8.

[Huang, Jiang & Zhang, 2014] Huang, B., Jiang, R., & Zhang, G., (2014), «Search strategy for scheduling flexible manufacturing systems simultaneously using admissible heuristic functions and nonadmissible heuristic functions», *Computers & Industrial Engineering*, vol. 71, pp. 21–26.

[Huang et al., 2009] Huang, B., Sun, Y., Sun, Y. M., & Zhao, C.-X., (2009), «A hybrid heuristic search algorithm for scheduling FMS based on Petri net model», *The International Journal of Advanced Manufacturing Technology*, vol. 48, no.9, pp. 925-933.

[Huang, Sun & Sun, 2008] Huang, B., Sun, Y., & Sun, Y.M., (2008), «Scheduling of flexible manufacturing systems based on Petri nets and hybrid heuristic search», *International Journal of Production Research*, vol. 46, pp. 4553–4565.

[Huang, Jeng & Chung, 2006] Huang, Y. S., Jeng, M. X., & Chung, D. H. (2006), «Siphon-based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 36, no. 6, pp. 1248-1256.

[Huanxin Henry & MengChu, 1998] Huanxin Henry, X., & MengChu, Z., (1998), «Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search», *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, no.3, pp. 384-393.

[Jamili, 2016] Jamili, A., (2016), «Robust job Shop Scheduling Problem: Mathematical Models, Exact and Heuristic Algorithms», *Expert Systems with Applications*, vol.55, pp. 341–350.

[Jarboui et al., 2008] Jarboui, B., Ibrahim, S., Siarry, P., & Rebai., A., (2008) «A combinatorial particle swarm optimisation for solving permutation flowshop problems», *Computers &Industrial Engineering*, vol. 54, no. 3, pp. 526-538.

[Jeng & Chen, 1998] Jeng, M. D., & Chen, S. C., (1998), « Heuristic search approach using approximate solutions to Petri net state equations for scheduling flexible manufacturing systems », *International Journal of Flexible Manufacturing Systems*, vol. 10, no. 2, pp. 139-162.

[Jeng, Chiou & Wen, 1998] Jeng, M. D., Chiou, W. D., & Wen, Y. L. (1998), « Deadlock-free scheduling of flexible manufacturing systems based on heuristic search and Petri net structures. Paper presented at the Proceedings of the 1998 », IEEE International Conference on Systems, Man, and Cybernetics. Part 1 (of 5), Piscataway, NJ, United States. San Diego, CA, USA.

[Kacem, 2003] Kacem, I., (2003), « Ordonnancement multicritère des job-shops flexibles: formulation, bornes inférieures et approche évolutionniste cooperative », Thèse de Doctorat, Université de Lille 1.

[Kammarti et al., 2005] Kammarti, R., Hammadi, S., Borne, P., & Ksouri, M., (2005), «Special Tabu Search in an Hybrid Evolutionary Approach for the PDPTW», SCI IEEE International Conference, Orlando (USA).

[Karray et al., 2008] Karray, A., Laabidi, K., & Ksouri, M., (2008), «Métaheuristiques pour la commande des systèmes non linéaires», CIFA Conférence International Francophone d'Automatique, Roumanie.

[Karp & Miller, 1969] Karp, R. M., Miller, R. E., (1969), «Parallel program schemata», *Journal of Computer and System Sciences*, vol. 3, no. 2, pp. 147-195.

[Kennedy, 2011] Kennedy J. (2011), « Particle Swarm Optimization», In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA.

[Kim, Suzuki & Narikiyo, 2007] Kim, Y. W., Suzuki, T., & Narikiyo, T., (2007), « FMS scheduling based on timed Petri Net model and reactive graph search », *Applied Mathematical Modelling*, vol. 31, no. 6, pp. 955-970.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P., (1983), «Optimization by simulated annealing», *science*, vol. 220, no. 4598, pp. 671-680.

[Koné et al., 2013] Koné, O., Artigues, C., López, P., & Mongeau, M., (2013), «Comparison of mixed integer linear programming model for the resource constrained project scheculing problem with consumption and production of resources», *Flexible Services and Manufacturing*, vol. 25, pp. 25-47.

[Koné et al., 2011] Koné, O., Artigues, C., López, P., & Mongeau, M., (2011), «Event-based milp model for resource-constrained project scheduling problems», *Computers & Operations Research*, vol. 38, pp. 3-13.

[Lee & Lee, 2010] Lee, J., & Lee, J.S., (2010), «Heuristic search for scheduling flexible manufacturing systems using lower bound reachability matrix», *Computers & Industrial Engineering*, vol. 59, no. 4, pp. 799–806.

[Lee & DiCesare, 1994] Lee, D.Y., & DiCesare, F., (1994), «Scheduling flexible manufacturing systems using Petri nets and heuristic search», *IEEE Transactions on Robotics and Automation*, vol. 10, no. 2, pp. 123–132.

[Lefebvre et al., 2018] Lefebvre, D., Rachidi, S., Leclercq, E., & Pigné, Y., (2018), «Diagnosis of structural and temporal faults for k bounded non-markovian stochasticPetri nets», *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 9, pp. 3369-3381.

[Lefebvre, 2018] Lefebvre, D., (2018), «Approximated Timed Reachability Graphs for performance evaluation and control of DES», *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 224-229.

[Lefebvre & Mejía, 2018] Lefebvre, D., & Mejía, G. (2018), « Robust scheduling in uncertain environment with Petri nets and beam search », *IFAC-PapersOnLine*, vol 51, no. 11, pp 1077-1082.

[Lefebvre & Daoui, 2018] Lefebvre, D., & Daoui, C., (2018), «Control Design for Bounded Partially Controlled TPNs Using Timed Extended Reachability Graphs and MDP», *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 6, pp. 2273-2283.

[Lefebvre, 2017a] Lefebvre, D., (2017a), «Evaluating the robustness of scheduling in uncertain environment with Petri nets», Valuetools Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools, pp. 170-177, Venice, Italy.

[Lefebvre, 2017b] Lefebvre, D., (2017b), «Dynamical Scheduling and Robust Control in Uncertain Environments with Petri Nets for DESs», *MDPI Processes*, vol. 5, no. 4, pp. 54-69.

[Lefebvre, 2016] Lefebvre, D., (2016), «Deadlock-free scheduling for timed Petri net models combined with MPC and backtracking», 13th International Workshop on Discrete Event Systems (WODES), pp. 466-471, Xi'an, China.

[Lefebvre, 2014] Lefebvre, D., (2014), «Fault diagnosis and prognosis with partially observed petri nets», *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 10, pp. 1413–1424.

[Lei et al., 2017] Lei, H., Xing, K., Han, L., & Gao, Z., (2017), «Hybrid heuristic search approach for deadlock-free scheduling of flexible manufacturing systems using Petri nets», *Applied Soft Computing*, vol. 55, pp. 413-423.

[Lei et al., 2014] Lei, H., Xing, K., Han, L., Xiong, F., & Ge, Z. (2014), «Deadlock-free scheduling for flexible manufacturing systems using Petri nets and heuristic search», *Computers and Industrial Engineering*, vol. 72, no. 1, pp. 297-305.

[Lemamou, 2009] Lemamou, E.A., (2009), «Ordonnancement de projet sous contraintes de ressources à l'aide d'un algorithme génétique à croisement hybride de type OEP», Université du Québec à Chicoutimi.

[Leon, Wu & Storer, 1994] Leon, J., Wu, S., & Storer, R. H., (1994), «Robustness measures and robust scheduling for job shops», *IIE Transactions*, vol. 26, no. 5, pp. 32–43.

[Le Pape, 1995] Le Pape, C., (1995), «Three mechanisms for managing resource constraints in the library for constraint-based scheduling», INRIA/IEEE Conference on Emerging Technologies and Factory Automation, pp. 980-995, Paris, France.

[Liaw, 1999] Liaw, C. F., (1999), «A tabu search algorithm for the open shop scheduling problem», *Computers & Operations Research*, vol. 26, pp. 109-126.

[Liouane, 1998] Liouane, N., (1998), «Contribution à l'élaboration d'un outil d'aide à la décision pour l'ordonnancement de production en environnement incertain», Thèse de Doctorat, Université des Sciences et Technologies de Lille1.

[List & Cetin, 2004] List, G. F., & Cetin, M., (2004), «Modeling Traffic Signal Control Using Petri Nets», *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 3, pp.177–187.

[Liu, Lefebvre & Li, 2019] Liu, G. Y., Lefebvre, D., & Li., Z. W., (2019), «Robust Deadlock-free Scheduling for FMS with Liveness-enforcing Supervisor Combined with Beam Search Controller», IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 1825-1830, Bary, Italy.

[Liu et al., 2018] Liu, G., Li, P., Li, Z., & Wu, N., (2018), «Robust Deadlock Control for Automated Manufacturing Systems with Unreliable Resources Based on Petri Net Reachability Graphs», *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 7, pp. 1371-1385.

[Liu & Barkaoui, 2016] Liu, G., & Barkaoui, K., (2016), «A survey of siphons in Petri nets», *Information Sciences*, vol. 363, pp. 198-220.

[Liu et al., 2014] Liu, H., Xing, K., Zhou, M., Han, L., & Wang, F., (2014), «Transition cover-based design of Petri net controllers for automated manufacturing systems», *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 2, pp. 196–208.

[Liu et al., 2013] Liu, G. Y., Li, Z. W., Barkaoui, K., & Al-Ahmari, A., (2013), « Robustness of deadlock control for a class of Petri nets with unreliable resources », *Information sciences*, vol. 235, no. 6, pp. 259-279.

[Liu, Gu & Xi, 2007] Liu, L., Gu, H., & Xi, Y., (2007), «Robust and Stable Scheduling of a Single Machine with Random Machine Breakdowns», *The International Journal of Advanced Manufacturing Technology*, vol.31, no. 7–8, pp. 645–654.

[Liu, 2000] Liu, J.W., (2000), «*In Real-Time Systems*», Prentice Hall 70.

[Lopez & Roubellat, 2001] Lopez, P., & Roubellat, F., (2001), «*Ordonnancement de la production*», Hermès Sciences, IC2 Productique.

[Lopez & Esquirol, 1999] Lopez, P., & Esquirol, P., (1999), «*L'ordonnancement*», Éditions Économia.

[Lourenço, Martin & Stützle, 2003] Lourenço, H.R., Martin, O., & Stützle, T., (2003), «Iterated Local Search », *In: Glover F., Kochenberger G.A. (eds) Handbook of Metaheuristics, International Series in Operations Research & Management Science*, vol. 57, pp. 321–353, Springer, Boston, MA.

[Low, 2005] Low, C. Y., (2005), «Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines», *Computers and Operations Research*, vol. 32, no. 8, pp. 2013-2025.

[Lowerre, 1976] Lowerre, B., (1976), «The Harpy Speech Recognition System», Thèse de Doctorat, Carnegie Mellon University.

[Luo et al., 2015] Luo, J.C., Xing, K., Zhou, M.C., Li, X.L., & Wang, X.N., (2015), «Deadlock-Free Scheduling of Automated Manufacturing Systems Using Petri Nets and Hybrid Heuristic Searc », *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol 45, pp. 530-541.

[Mazdeh & Rostami, 2014] Mazdeh, M. M., & Rostami, M., (2014), «A branch-and-bound algorithm for two-machine flow-shop scheduling problems with batch delivery costs», *International Journal of Systems Science: Operations & Logistics*, vol. 1, no. 2, pp. 94–104.

[Mehta & Uzsoy, 1998] Mehta, S. V, & Uzsoy, R. M., (1998), «Predictable scheduling of a job shop subject to breakdowns», *IEEE Transactions on Robotics and Automation*, vol. 14, no.3, pp. 365–378.

[Mejía & Lefebvre, 2019] Mejía, .G, & Lefebvre, .D., (2019), «Robust scheduling of flexible manufacturing systems with unreliable operations and resources», *International Journal of Production Research*, vol. 58, pp. 6474-6492.

[Mejía & Niño, 2017] Mejía, G., & Niño, K., (2017), «A new Hybrid Filtered Beam Search algorithm for deadlock-free scheduling of flexible manufacturing systems using Petri Nets», *Computers & Industrial Engineering*, vol 108, pp. 165-176.

[Mejía et al., 2016] Mejía, G., Niño, K., Montoya, C., Sánchez, M. A., Palacios, J., & Amodeo, L., (2016), «A Petri Net-based framework for realistic project management and scheduling: An application in animation and videogames», *Computers & Operations Research*, vol. 66, pp.190-198.

[Mejía & Montoya, 2009] Mejía, G., & Montoya, C., (2009), «Scheduling manufacturing systems with blocking: A Petri net approach», *International Journal of Production Research*, vol. 47, no.22, pp. 6261-6277.

[Mejía & Odrey, 2005] Mejía, G., & Odrey, N. G., (2005), «An approach using Petri nets and improved heuristic search for manufacturing system scheduling», *Journal of Manufacturing Systems*, vol. 24, no. 2, pp. 462-476.

[Mellouli et al., 2004] Mellouli, K., El Kamel, A., & Borne, P., (2004), «*Programmation linéaire et applications. Eléments de courset exercices résolus*», Editions Technip.

[Mesghouni, 1999] K. Mesghouni, (1999), «Application des algorithmes évolutionnistes dans les problèmes d'optimisation en ordonnancement de la production», Thèse de Doctorat, Université des Sciences et Technologies de Lille 1

[Merlin, 1974] Merlin, P. M. (1974), «A study of the recoverability of computing systems», Thèse de Doctorat, Department of Information and Computer Science, University of California, Irvine, CA.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E., (1953), «Equation of state calculations by fast computing machines», *The journal of chemical physics*, vol. 21, no. 6, pp. 1087-1092.

[Mok, 1983] Mok, A., (1983), «Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment», Thèse de Doctorat, MIT Laboratory for Computer Sciences.

[Molloy, 1982] Molloy, M. K., (1982), «Performance analysis using stochastic Petri nets», *IEEE Transactions on Computers*, vol. 31, no.9, pp. 913–917.

[Moradi & Shadrokh, 2018] Moradi, H., & Shadrokh, S., (2018), «A robust scheduling for the multi-mode project scheduling problem with a given deadline under uncertainty of activity duration», *International Journal of Production Research*, vol. 57, no. 3, pp. 1–30.

[Moro & Kelleher, 2002] Moro, A. R., Yu, H., & Kelleher, G., (2002), «Hybrid heuristic search for the scheduling of flexible manufacturing systems using Petri nets», *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 240–245.

[Moro, Yu & Kelleher, 2000] Moro, A. R., Yu, H., & Kelleher, G., (2000), «Advanced scheduling methodologies for flexible manufacturing systems using Petri nets and heuristic search», *IEEE International Conference on Robotics and Automation*, pp. 2398–2403, San Francisco, CA, USA.

[Murata, 1989] Murata, T., (1989), «Petri nets: Properties, analysis and applications», *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541-580.

[Nilsson, 1980] Nilsson, N., (1980), «*Principles of Artificial Intelligence*», Palo Alto, CA: Tioga.

[Osman & Potts, 1989] Osman, I. H., & Potts, C. N., (1989), «Simulated annealing for permutation flow-shop scheduling», *Omega*, vol. 17, no. 6, pp. 551-557.

[Ow & Morton, 1988] Ow, P. S., & Morton, T. E., (1988), «Filtered beam search in scheduling», *International Journal of Production Research*, vol. 26, no. 1, pp. 35–62.

[Pearl, 1984] Pearl, J., (1984), «*Heuristics: Intelligent Search Strategies for Computer Problem Solving*», *MA*, USA: Addison-Wesley.

[Petri, 1962] Petri, C.A., (1962), «Communicating with Automata», Thèse de Doctorat, Hamburg.

[Pinedo, 1955] Pinedo, M., (1955), «*Scheduling : Theory, Algorithms and systems*», Prentice-Hall, Englewood Clis, New Jersey.

[Piroddi, Cordone & Fumagalli, 2008] Piroddi, L., Cordone, R., & Fumagalli, I., (2008), «Selective Siphon Control for Deadlock Prevention in Petri Nets», *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 6, pp. 1337-1348.

[Rabin, 2000] Rabin, S., (2000), «*A* Speed Optimizations* », In Mark Deloura Game Programming Gems, pp. 272-287.

[Rachidi et al., 2017] Rachidi, S., Leclercq, E., Pigné, Y., & Lefebvre, D., (2017), «PN modeling of discrete event systems with temporal constraints», 21th International Conference on System Theory, Control and Computing (ICSTCC), pp. 70-75, Sinaia, Romania.

[Rahman et al., 2015] Rahman, H. F., Sarker, R., & Essam, D., (2015), «A genetic algorithm for permutation flow shop scheduling under make to stock production system», *Computers & Industrial Engineering*, vol. 90, pp. 12-24.

[Ramadge & Wonham, 1989] Ramadge, P. J. G., & Wonham, W. M., (1989), «The control of discrete event systems», *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98.

[Ramchandani, 1974] Ramchandani, C., (1974), «Analysis of asynchronous concurrent systems by timed Petri nets», Thèse de Doctorat, Massachusetts Institute of Technology, Cambridge, MA, Project MAC Report MAC-TR-120.

[Reeves, 1995] Reeves, C. R., (1995), «A genetic algorithm for flowshop sequencing», *Computers & operations research*, vol. 22, no. 1, pp. 5-13.

[Reyes, Kelleher & Lloyd, 2002] Reyes, A., Yu, H., Kelleher, G., & Lloyd, S., (2002), «Integrating Petri nets and hybrid heuristic search for the scheduling of FMS», *Computers in Industry*, vol. 47, no. 1, pp. 123–138.

[Rodammer & Preston White, 1999] Rodammer, F. A., & Preston White, K., (1999), «A recent survey of production Scheduling», *IEEE Transaction on Systems, Man and Cybernetics*, vol. 18, no. 6, pp. 841-851.

[Roy & Bouysson, 1993] Roy, B., & Bouysson, D., (1993), «Aide multi-critères à la décision : Méthodes et cas», *Collection Gestion Série : Production et technologie quantitatives appliquées à la gestion,* Edition Economica, Paris.

[Russell & Norvig, 1995] Russell, S., & Norvig, P., (1995), «*Artificial Intelligence: A Modern Approach*», Upper Saddle River, NJ: Prentice Hall.

[Sabuncuoglu & Bayiz, 1999] Sabuncuoglu, I., & Bayiz, M., (1999), «Job shop scheduling with beam search», *European Journal of Operational Research*, vol. 118, no.2, pp. 390–412.

[Sakarovitch, 1984] Sakarovitch, M., (1984), «*Graphes et Programmation Linéaire*», Edition Hermann, Paris.

[Schwiegelshohn, 2004] Schwiegelshohn, U., (2004), «Preemptive Weighted Completion Time Scheduling of Parallel Jobs», *SIAM Journal on Computing*, vol. 33, no. 6, pp. 1280-1308.

[Sifakis, 1979] Sifakis, J. (1979), «Performance Evaluation of Systems Using Nets, Net Theory and Applications», *Lecture Notes in Computer Science*, Springer Verlag, pp. 307-319.

[Silva, 2013] Silva, M., (2013), «Half a century after Carl adam Petri's Phd thesis: A perspective on the field», *Annual Review in Controls*, vol. 37, no. 2, pp. 191-192.

[Singh & Baghel, 2006] Singh, A., & Baghel, A. S., (2008). «A new grouping genetic algorithm approach to the multiple traveling salesperson problem», *Soft Computing*, vol. 13, no. 1, pp. 95–101.

[Taillard et al., 2001] Taillard, E. D., Gambardella, L. M., Gendreau, M., & Potvin, J. Y., (2001), «Adaptive memory programming: a unified view of metaheuristics», *European Journal of Operational Research*, vol. 135, no. 1, pp. 1-16.

[Tasgetiren et al., 2007] Tasgetiren, M. F., Liang, Y., Sevkli, M., & Gencyilmaz, G., (2007), «A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem», *European journal of operational research*, vol. 177, no. 3, pp. 1930-1947.

[Tillmann & Ney, 2018] Tillmann, C., & Ney, H., (2018), «Word Reordering and a Dynamic Programming Beam Search Algorithm for Statistical Machine Translation», *Computational Linguistics*, vol. 29, no. 1, pp. 97-133.

[Vieira, Herrman & Lin, 2003] Vieira, G., Herrman, J., & Lin, E., (2003), «Rescheduling Manufacturing Systems: a Framework of Strategies, Policies and Methods», *Journal of Scheduling*, vol. 6, no. 1, pp. 39–62.

[Wang & Kwei-Jay, 1999] Wang, Y. C., & Kwei-Jay, L., (1999), «Implementing a general real-time scheduling framework in the RED-Linux real-time kernel», *IEEE Real-Time Systems Symposium*, pp. 246-255, Phoenix, USA.

[Xing et al., 1996] Xing, K. Y., Xing, K. L., & Li, J. M., (1996), «Deadlock avoidance controller for a class of manufacturing systems, Proceedings», IEEE International Conference on Robotics and Automation, pp. 200-204, Minneapolis, USA.

[Xiong & Zhou, 1998] Xiong, H. H., & Zhou, M. (1998), «Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search», *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, no. 3, pp. 384–393.

[Yalaoui, 2006] Yalaoui, F., (2006), «Optimisation et gestion de la pro duction: problème de conception et problème d'ordonnancement», Habilitation à diriger les recherches, Université de Technologie de Compiègne.

[Yu et al., 2003a] Yu, H., Reyes, A., Cang, S., & Lloyd, S., (2003), «Combined Petri net modelling and AI-based heuristic hybrid search for flexible manufacturing systems—part II. Heuristic hybrid search», *Computers & Industrial Engineering*, vol. 44, no. 4, pp. 545–566.

[Yu et al., 2003b] Yu, H., Reyes, A., Cang, S., & Lloyd, S., (2003), «Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems—part 1. Petri net modelling and heuristic search», *Computers & Industrial Engineering*, vol. 44, no. 4, pp. 527–543.

[Zafra-Cabeza, Ridao & Camacho, 2004] Zafra-Cabeza, A., Ridao, M. A., & Camacho, E. F., (2004), «Chance constrained project scheduling under risk», Conference Proceedings IEEE International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1789–1794.

[Zhang & Wu, 2010] Zhang, R., & Wu, C., (2010), «A hybrid immune simulated annealing algorithm for the job shop scheduling problem», *Applied Soft Computing*, vol. 10, vol. 1, pp. 79-89.

[Zhang, Freiheit & Yang, 2005] Zhang, W., Freiheit, T., & Yang, H., (2005), «Dynamic scheduling in flexible assembly system based on timed Petri nets model», *Robotics and Computer Integrated Manufacturing*, vol. 21, pp. 550–558.

Ordonnancement des ateliers hybrides en environnement incertain

Résumé. Cette thèse concerne les problèmes d'ordonnancement pour une classe de systèmes manufacturiers flexibles (FMS) en environnement incertain. Les systèmes considérés répondent aux exigences de l'industrie 4.0 qui requièrent des organisations plus agiles permettant de proposer des produits personnalisés. Plus précisément, les systèmes proposés dans ce travail sont appelés FMS hybrides et traitent les opérations avec des contraintes de précédence partielles entre les opérations. La première contribution est une approche systématique de modélisation des FMS hybrides avec les réseaux de Petri temporisés sur les transitions (T-TPN) et un formalisme multi-niveaux basé sur la structuration hiérarchique des opérations. La deuxième contribution concerne la recherche d'ordonnancements pour les FMS hybrides. Nous utilisons la recherche en faisceau et proposons une fonction coût basée sur la modélisation multi-niveaux. Nous proposons également une nouvelle variante de la recherche en faisceau, appelée Generation Filtered Beam Search (GFBS). Cette nouvelle variante améliore la qualité des candidats sélectionnés par rapport à d'autres variantes par un meilleur équilibre entre la largeur et la profondeur de la recherche. La troisième contribution concerne l'étude des aléas pouvant affecter le système. Un risque d'interruption, dû à l'occurrence d'événements incontrôlables, est intégré, et une nouvelle fonction coût qui intègre ce risque est définie. Enfin, un algorithme de recherche en faisceau modifié, appelé Generation Double Filtered Beam Search (GDFBS), qui accélère la convergence de la méthode, est proposé. Ce nouvel algorithme est basé sur un mécanisme de filtrage qui utilise la fonction coût pour explorer de manière sélective l'espace d'état du T-TPN afin de trouver une séquence de contrôle qui réalise un compromis entre performance et risque.

Mots clés : Ordonnancement, système manufacturier flexible, réseau de Petri temporisé, FMS hybride, recherche en faisceau, performance, risque.

Scheduling of hybrid workshops in uncertain environment

Abstract. This thesis concerns the scheduling problems for a class of flexible manufacturing systems (FMS) in uncertain environment. The considered systems meet the requirements of Industry 4.0, which require more flexibility and modularity to offer personalized products. More precisely, the proposed systems in this work are called hybrid FMS and deal with operations with partial precedence constraints. The first contribution is a systematic approach used for the modeling of hybrid FMS with timed Petri nets (T-TPN) and a multi-level formalism based on the hierarchical organization of operations. The second contribution concerns the scheduling of hybrid FMS. We use the beam search and propose a cost function based on the multi-level modeling. We also offer a new variant of beam search called Generation Filtered Beam Search (GFBS). This new variant improves the quality of the selected candidates compared to other variants of beam search by a better balance between the width and the depth of the search. The third contribution concerns the study of interruptions that can affect the system. A risk of interruption, due to the occurrence of uncontrollable events, is integrated, and a new cost function that incorporates this risk is defined. Finally, a modified beam search algorithm, called Generation Double Filtered Beam Search (GDFBS), which accelerates the convergence of the method, is proposed. The new algorithm is based on a filtering mechanism that uses the cost function to selectively explore the state space of the TPN in order to find a control sequence that achieves a tradeoff between performance and risk.

Keywords: Scheduling, flexible manufacturing system, timed Petri net, hybrid FMS, beam search, performance, risk.