AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

Collegium Sciences et Technologie
Pôle AM2I
École Doctorale IAEM Lorraine
Comission de Mention Automatique

Ordonnancement sous perturbations : cadre d'étude et approche d'évaluation de la robustesse par automates stochastiques

THÈSE

présentée et soutenue publiquement le 07 décembre 2020

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention Automatique, Traitement du Signal et des Images, Génie Informatique)

par

Sara Himmiche

Composition du jury

Président :	Sébastien Lahaye, Professeur des Universités, LARIS, Université d'Angers
Rapporteurs :	Mireille Jacomino, Professeur des Universités, G-SCOP, Institut Polytechnique de Grenoble Dimitri Lefebvre, Professeur des Universités, GREAH, Université Le Havre Normandie
Examinateurs :	David Lemoine, Professeur Assistant HDR, LS2N, IMT-Atlantique Nantes Marie Duflot-Kremer, Maître de conférences, LORIA, Université de Lorraine Alexis Aubry, Maître de conférences HDR, CRAN, Université de Lorraine (Co-Directeur de thèse) Pascale Marangé, Maître de conférences, CRAN, Université de Lorraine (Encadrante de thèse) Jean-Fançois Pétin, Professeur des Universités, CRAN, Université de Lorraine (Directeur de thèse)

Centre de Recherche en Automatique de Nancy — UMR 7039
Université de Lorraine — CNRS

Mis en page avec la classe thesul.

À mon mari,
Pour ton appuie, ton encouragement et surtout ton amour,
Merci

Remerciements

L'aboutissement de ce projet de thèse n'aurait pas vu le jour sans la présence et le soutien de plusieurs personnes que j'ai pu rencontrer tout au long de cette aventure. Je leur dédie donc ces quelques mots pour leur exprimer ma profonde gratitude.

Pour cela, je commencerai par remercier ceux qui ont embarqué avec moi dans cette aventure, qui m'ont fait confiance et m'ont guider, ceux que j'appelle désormais ma famille de coeur. Mes directeurs de thèse, Jean-François Pétin, Alexis Aubry et mon encadrante Pascale Marangé. Merci à Jean-François pour avoir accepté de diriger cette thèse, pour tous ses précieux conseils et le temps qu'il m'a consacré. Merci à Alexis, de m'avoir pousser à dépasser mes limites et mes peurs. Merci pour son encouragement et son soutien dans les moments de doute. Toutes ses heures passées à débattre avec lui et Pascale n'ont pas été vaines. Je remercie Pascale, qui m'a offert l'opportunité de suivre la voie de la recherche, pour cela je serai toujours reconnaissante, alors un grand merci pour avoir cru en moi. Ses conseils tout au long de cette thèse, et même pendant mon Master, m'ont permis de m'initier à l'univers de la recherche et de l'enseignement, alors merci d'avoir été un pilier pour moi.

Je veux remercier également les membres du Jury, qui ont accepté d'évaluer mon travail. Merci à Sébastien Lahaye pour m'avoir fait l'honneur de présider la soutenance. Je remercie chaleureusement Mireille Jacomino et Dimitri Lefebvre pour avoir accepté d'être rapporteurs de ce travail, ainsi que pour la qualité des échanges que nous avons pu avoir et leurs commentaires constructifs. Je remercie également David Lemoine et Marie Duflot-Kremer pour leurs questions et remarques, lors de la soutenance.

Je remercie les merveilleuses personnes que j'ai rencontrées au sein du CRAN, les permanents, l'équipe administratives, les doctorants et les post-doctorants pour leur présence et leur accueil. Merci à Benoît Iung et Hind El-Haouzi, les directeurs du département ISET pour leur accueil chaleureux au sein de l'équipe. Merci à Eric Levrat pour m'avoir soutenue dans ma décision de faire une thèse. Je remercie David Gouyon pour ses conseils en enseignement. Je remercie également Rémi Pannequin qui a été d'une grande aide pour la réalisation de mon expérimentation. Je remercie tout particulièrement Laëtitia, Margaux, Yassamin, Chiara, Florent, Concetta, Hang et Fabian pour les moments inoubliables que nous avons pu partager.

Je tiens à exprimer ma reconnaissance et mon amour aux membres de ma famille . À mes parents, ma soeur et mon frère, merci pour le soutien infaillible, et ce malgré les kilomètres qui nous séparent. Enfin, merci à mon autre moitié, pour sa présence et sa patience.

Sommaire

Table des figures **ix**

Liste des tableaux **xi**

Introduction générale **1**

1
Ordonnancement sous perturbations : Contexte et état de l'art **5**

1.1 Introduction 5
1.2 Ordonnancement de la production : fondements 6
1.2.1 Problème d'ordonnancement 6
1.2.2 Solution d'ordonnancement 7
1.3 Problème d'ordonnancement et Industrie 4.0 8
1.3.1 Challenges de l'Industrie 4.0 8
1.3.2 Contexte général de l'ordonnancement sous perturbations 9
1.4 Ordonnancement sous perturbations 10
1.4.1 Classification des perturbations 10
1.4.2 Stratégies d'ordonnancement sous perturbations 11
1.4.3 Modélisation des perturbations 12
1.5 Ordonnancement proactif soumis à des perturbations stochastiques 17
1.5.1 Performances d'un ordonnancement sous perturbations 18
1.5.2 Modèles pour le problème d'ordonnancement sous perturbations 20
1.5.3 Résolution du problème d'ordonnancement sous perturbations 22
1.5.4 Synthèse de l'état de l'art 26
1.6 Hypothèses et questions de recherche 27
1.7 Conclusion 27

2
Cadre d'étude pour l'ordonnancement robuste **29**

2.1 Introduction 29
2.2 Performance de robustesse : spécification 30
2.2.1 À propos de robustesse 30
2.3 Formalisation des problèmes de robustesse 33
2.3.1 Problèmes d'évaluation 33
2.3.2 Problèmes d'optimisation 39
2.4 Conclusion 40

3
Évaluation du niveau de service par automates stochastiques 45

3.1 Introduction 46
3.2 Présentation du processus d'évaluation 46
3.2.1 Approche proposée 46
3.2.2 Présentation de l'étude de cas 47
3.3 Modélisation du problème : ordonnancement et perturbations 48
3.3.1 Modèles d'automates temporisés stochastiques 48
3.3.2 Structure de modélisation 53
3.3.3 Modélisation de l'ordonnancement déterministe 54
3.3.4 Modélisation des perturbations 57
3.3.5 Simulation de l'évolution des modèles 67
3.4 Évaluation du niveau de service 69
3.4.1 Modélisation de la propriété d'analyse du niveau de service 69
3.4.2 Model-checking probabiliste 73
3.5 Implémentation du processus d'évaluation 76
3.5.1 Implémentation du processus d'évaluation dans UppAal SMC 76
3.6 Conclusion 80

4
Analyse des performances de l'approche d'évaluation 81

4.1 Introduction 82
4.2 Généricité du processus d'évaluation 82
4.2.1 Généricité vis-à-vis de l'atelier de production 82
4.2.2 Généricité vis-à-vis des perturbations 84
4.3 Sensibilité du processus d'évaluation 87
4.3.1 Méthodologie 88
4.3.2 Sensibilité aux paramètres du Model-Checker 91
4.3.3 Sensibilité du processus aux paramètres d'entrée 93

4.4 Passage à l'échelle 100
4.4.1 Construction du plan d'expériences 101
4.4.2 Exécution du plan d'expérience 101
4.4.3 Analyse des résultats du passage à l'échelle 101
4.4.4 Passage à l'échelle : synthèse 103
4.5 Conclusion 104

5
Du processus d'évaluation vers l'aide à la décision 105

5.1 Introduction 105
5.2 Ordonnancement robuste sur une nouvelle ligne de production 106
5.2.1 Choix de l'ordonnancement permettant d'absorber la plus grande déviation sur les durées opératoires 108
5.2.2 Détermination de l'ordonnancement le plus robuste 108
5.2.3 Détermination d'une date de livraison malgré les perturbations 109
5.2.4 Identification de la ressource critique à surveiller 110
5.3 Optimisation robuste pour l'atelier d'emballage 110
5.4 Conclusion 114

Conclusions et perspectives 117

Bibliographie 121

Annexes

A
Formulation d'un problème d'ordonnancement

B
UppAal SMC

C
Niveau de service VS L_λ-robustesse

C.1 La L_λ-robustesse 133
C.2 Comparaison avec le niveau de service 133

D
Front de Pareto

D.1 Définition d'un front de Pareto 135

Table des figures

1 Questions autour du processus d'évaluation 2

1.1 Exemple d'ordonnancement 8
1.2 Problématique d'ordonnancement dans le contexte de l'industrie 4.0 10
1.3 Classification des modèles de perturbations 12
1.4 Fonction d'appartenance de D 14
1.5 Exemple de fonctions d'appartenance de P 15
1.6 Fonction de répartition uniforme discrète 16
1.7 Fonction de répartition uniforme continue 16
1.8 Fonction de répartition exponentielle continue 17
1.9 Méthode de traitement de la problématique 18
1.10 Synthèse des performances de l'ordonnancement sous perturbations 19
1.11 Synthèse des approches de modélisation du problème 20
1.12 Synthèse des approches de résolution des modèles 23

3.1 Processus d'évaluation de robustesse 47
3.2 Atelier de production (Étude de cas de [Giard, 2003]) 48
3.3 Diagramme de Gantt de s pour l'étude de cas de [Giard, 2003] 48
3.4 Exemple d'un automate temporisé stochastique 51
3.5 Évolution de l'événement e_1 (Exemple 6) 53
3.6 Structure de modélisation 54
3.7 Évolution des modèles proposés 55
3.8 Patrons de l'ordonnancement 57
3.9 Exemple d'instanciation des patrons d'ordonnancement 58
3.10 Scénario de perturbation de l'étude de cas 60
3.11 Patron de l'aléa 61
3.12 Instanciation du patron de l'aléa pour l'opération o_{12} 61
3.13 Patron d'incertitude 62
3.14 Instanciation du patron d'incertitude u_1^{ex} pour o_{12} 65
3.15 Intégration des perturbations dans le patron d'opération 66
3.16 Prise en compte des perturbations pour o_{12} 67
3.17 Simulation de l'évolution des modèles proposés 68
3.18 Graphe d'états des patrons : Aléa(H), Opération(Op), Ressource(R) 68
3.19 Graphe d'états patrons : Incertitude(U), Opération(Op), Ressource(R) 69
3.20 Hiérarchie des logiques temporelles 70
3.21 Opérateur **X** 71
3.22 Opérateur **F** 71

3.23 Opérateur **G** . . . 71
3.24 Opérateur **U** . . . 72
3.25 Expression *EF* . . . 72
3.26 Expression *AF* . . . 72
3.27 Principe du Model Cheking . . . 74
3.28 Implémentation des patrons d'ordonnancements dans UppAal SMC . . . 77
3.29 Implémentation des patrons de perturbation dans UppAal SMC . . . 78
3.30 Instanciation des patrons sur UppAal SMC . . . 78

4.1 Scénario de combinaison de plusieurs perturbations . . . 87
4.2 Paramètres de l'expérimentation . . . 88
4.3 Processus d'expérimentation . . . 90
4.4 Exemple du graphique des effets principaux (Source : Minitab) . . . 91
4.5 Effets des paramètres du Model-Checker sur les performances du processus . . . 92
4.6 Analyses de la variance en fonction des paramètres ϵ et α . . . 93
4.7 Exemple d'ordonnancement s_1 : règle *capa_ max* . . . 95
4.8 Exemple d'ordonnancement s_2 : règle *capa_ restante_ max* . . . 96
4.9 Exemple d'ordonnancement s_3 : règle *default_ sched* . . . 96
4.10 Impact de la règle d'ordonnancement sur $Time$ . . . 96
4.11 Effets de $Pert_i$ sur la performance $Time$ . . . 97
4.12 Analyse de la variance en fonction des combinaisons de perturbations . . . 97
4.13 Effets principaux des paramètres de perturbations sur $Time$ . . . 98
4.14 Analyse de la variance en fonction des paramètres de perturbations . . . 98
4.15 Impact de $p(l)$ sur $Time$ . . . 99
4.16 Impact de $p(h, o_{jk})$ sur $Time$ . . . 99
4.17 Effets principaux pour les paramètres de la deadline . . . 99
4.18 Analyse de la variance en fonction de la deadline . . . 100
4.19 Répartition du temps de résolution . . . 102
4.20 Effets de la taille de l'atelier . . . 102
4.21 Analyse de la variance en fonction de la taille de l'atelier . . . 102
4.22 Effets du ratio $NbOp/NbR$ sur $Time$ . . . 103
4.23 Analyse de la variance pour $NbOp/NbR$ . . . 103

5.1 Illustration du scénario industriel . . . 106
5.2 Ordonnancements déterministes générés à partir de $F2||C_{max}$ . . . 107
5.3 Méthode hybride pour trouver la solution robuste [Marangé et al., 2020] . . . 113

1 Synthèse des contributions de la thèse . . . 118

B.1 Outil UppAal SMC : présentation . . . 130
B.2 Exemple de déclaration dans UppAal SMC . . . 130
B.3 Exemple de vérification d'une propriété sur UppAal . . . 131

D.1 Exemple d'un front de Pareto . . . 135
D.2 Exemple d'une surface de Pareto . . . 136

Liste des tableaux

1.1 Données pour l'exemple 1 . . . 7
1.2 Classification des perturbations . . . 11
1.3 Exemple de la modélisation par approche ensembliste . . . 13
1.4 Approche ensembliste : Construction de scénarios . . . 13

2.1 Problèmes d'évaluation de la robustesse . . . 42
2.2 Problèmes d'optimisation de la robustesse . . . 43

3.1 Données de l'atelier de production pour s . . . 49
3.2 Données des patrons d'ordonnancement . . . 56
3.3 Données de perturbations . . . 59
3.4 Illustration du calcul de $p(l)$ . . . 64
3.5 Model-Checking : NMC Versus SMC . . . 74
3.6 Résultats de l'évaluation de la robustesse . . . 80

4.1 Règles d'instanciation des prédicats suivant le type d'atelier . . . 83
4.2 Scénarios de perturbations interprétés du benchmark de [Trentesaux et al., 2013] 85
4.3 Niveaux des paramètres du Model Checker . . . 92
4.4 Facteur lié à la règle d'ordonnancement s_i . . . 94
4.5 Facteur lié à la combinaison de perturbations de perturbations $Pert$ . . . 94
4.6 Facteurs liés à $Pert$ . . . 94
4.7 Facteur lié à $\tilde{d}$ . . . 95
4.8 Plan d'expériences pour l'analyse de sensibilité . . . 95
4.9 Synthèse des effets des facteurs . . . 100
4.10 Facteurs liés à la taille de l'atelier de production . . . 101

5.1 Données de l'atelier Flow Shop . . . 106
5.2 Analyse de la stabilité des ordonnancements . . . 108
5.3 Résultats de l'évaluation du niveau de service pour 6 ordonnancements . . . 109
5.4 Identification de la ressource critique . . . 110
5.5 Caractéristiques des jobs pour $R2||C_{max}$ . . . 114
5.6 Utilisations du processus d'évaluation dans un contexte industriel . . . 116

A.1 Notations des ateliers production . . . 128
A.2 Contraintes des ateliers de productions . . . 128

Introduction générale

L'ordonnancement de tâches consiste à définir une séquence caractérisée par l'affectation d'une tâche sur une ressource ayant les capacités de la réaliser ainsi que par les dates de début et de fin. Dans cette thèse, nous nous somme placé dans le contexte manufacturier, c'est à dire que l'ordonnancement de tâches permet de planifier des opérations de production (ou de maintenance) sur des ressources (machines ou opérateurs) à partir d'un ensemble de données connues (durée des opérations, capacités des machines, gamme de fabrication, quantités de produits à réaliser...) et selon des objectifs fixés (optimisation des durées de fabrication, optimisation des taux d'utilisation des ressources...).

Classiquement, les approches permettant d'obtenir un ordonnancement optimal font objet d'une littérature scientifique abondante, notamment dans la communauté de recherches opérationnelles (RO) [Pinedo, 2012]. Elles proposent, généralement, des solutions d'ordonnancement prévisionnelles en considérant un environnement de production stable et certain. L'ordonnancement optimal résultant est alors basé sur l'hypothèse optimiste que les données de l'atelier de production sont certaines et qu'aucun événement imprévu ne peut se produire. En réalité, l'ordonnancement optimal risque de voir ses performances détériorées quand il est mis en œuvre dans la mesure où un système de production évolue, dans la pratique, dans un environnement dynamique et doit faire face à de nombreuses perturbations. Celles-ci peuvent en effet avoir un impact l'ordonnancement prévisionnel en modifiant les durées réelles d'exécution des opérations ou encore la disponibilité des ressources. L'importance de cet impact a évolué au fil des années en fonction de la nature de l'atelier de production et des exigences des clients. En particulier, les évolutions récentes vers l'Industrie 4.0 s'accompagnent d'actions de ré-organisation, d'optimisation et de modernisation des ateliers de production qui se traduisent par des exigences toujours croissantes en termes de personnalisation des produits, de flexibilité voire d'agilité des processus de production.

Dans ce contexte, une perturbation peut avoir un impact considérable sur l'ordonnancement de la production en provoquant des phénomènes de cascades ou d'effet papillon dégradant fortement les performances de l'atelier de production. La prise en compte des perturbations en ordonnancement devient alors primordiale pour garantir un niveau de performance élevée.

L'ordonnancement sous perturbations n'est pas une nouvelle problématique de recherche. En effet, cette question a été abordée en profondeur dans le domaine de la Recherche Opérationnelle [Kouvelis and Yu, 1997] [Billaut et al., 2010]. Les travaux existants proposent de construire des ordonnancements robustes qui permettent de garantir certaines performances et ce malgré la présence de perturbations. Ils couvrent deux problématiques complémentaires : celle de la recherche d'une solution optimale et celle de l'évaluation des performances de l'ordonnancement en présence de perturbations. Les approches de résolution proposées sont souvent dédiées à un problème d'ordonnancement spécifique car dépendantes du type de l'atelier de production, des

perturbations considérées et des objectifs de performance poursuivis.

Cette thèse explore une voie alternative et complémentaire à ces approches classiques ayant pour objectif d'accroître le degré de généricité des techniques de résolution dans le contexte de l'ordonnancement sous perturbations. Ces dernières années, les méthodes et les langages basés sur la théorie des Systèmes à Évènements Discrets (SED) ont démontré leur efficacité pour la résolution de problèmes d'ordonnancement dans des premiers travaux [Panek et al., 2006] [Marangé et al., 2011] [Abdeddaïm and Masson, 2012]. La nature dynamique de ces modèles en font d'excellents candidats pour la prise en compte des phénomènes dynamiques induits par les perturbations. Dans ce contexte, ce mémoire propose un processus d'évaluation d'un ordonnancement sous perturbations basé sur une classe de modèles SED : les automates stochastiques. Il est organisé selon les différentes questions présentées en dans la Figure 1.

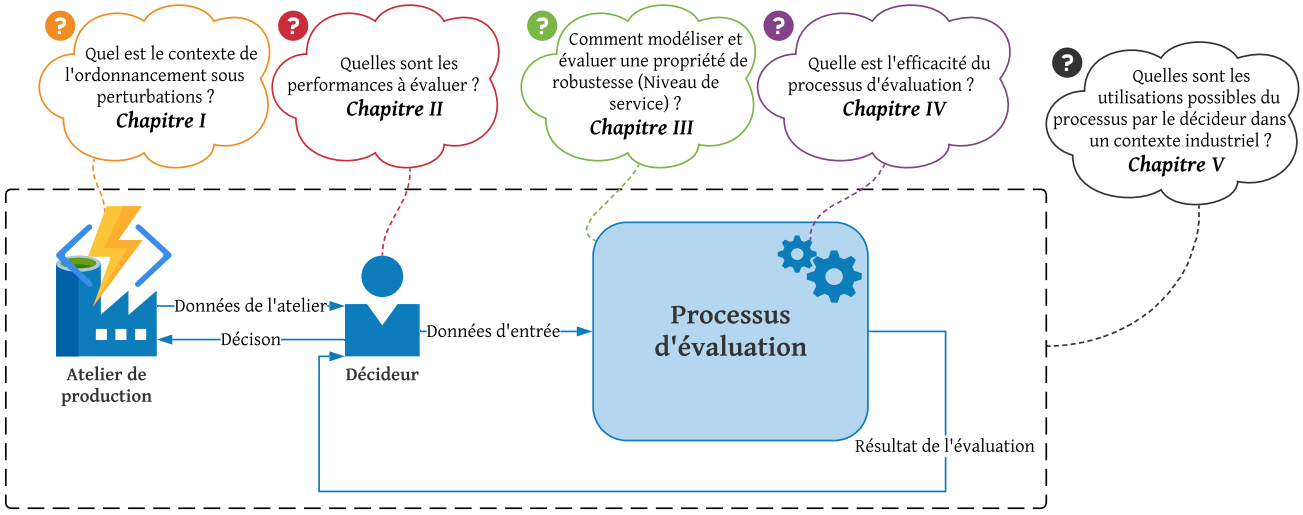


FIGURE 1 – Questions autour du processus d'évaluation

Le chapitre 1 définit le contexte scientifique de la problématique de l'ordonnancement sous perturbations. Dans une première partie, les notions d'ordonnancement déterministe sont introduits et l'importance de la prise en compte des perturbations dans le contexte d'un atelier de production de l'industrie 4.0 est mise en avant. La deuxième partie du chapitre présente les concepts généraux relatifs à l'ordonnancement sous perturbations. Une classification de ces perturbations est proposée puis différentes stratégies d'ordonnancement face à ces perturbations sont identifiées. Enfin, parmi celles-ci, l'ordonnancement proactif est retenu et requiert une modélisation préalable des perturbations qui est présentée à la fin de cette partie. Le contexte favorable de l'Industrie 4.0 qui met à disposition de nombreuses données historiques sur les performances des ateliers plaide en faveur de modèles stochastiques pour représenter les perturbations. La dernière partie du chapitre présente donc un état de l'art des approches proactives permettant de modéliser et résoudre le problème d'ordonnancement sous perturbations décrites avec des données stochastiques. La synthèse de l'état de l'art permet de catégoriser les approches de perturbations selon leur performances et méthodes de modélisation et de résolution. De cette synthèse, en découlent les hypothèses de recherche adoptées dans la suite du manuscrit ainsi que les questions auxquelles il ambitionne de répondre.

Le chapitre 2 aborde la notion de robustesse d'un ordonnancement vis-à-vis de perturbations. Dans une première partie, la robustesse est présentée comme un ensemble de propriétés (niveau de service [Dauzère-Pérès et al., 2010], sensibilité, stabilité...) en s'appuyant sur les définitions de la littérature. Dans un second temps, nous proposons une formalisation de ces propriétés faisant apparaître les différentes relations de dépendances ou d'inclusions existant entre elles. Le chapitre montre en particulier que les performances utiles à un décideur peuvent s'exprimer à partir du niveau de service associé à une mesure de probabilité. En dernier lieu, un cadre d'étude pour l'ordonnancement robuste est présenté. Deux types de problèmes de robustesse sont identifiés : des problèmes d'optimisation et des problèmes d'évaluation. Dans la mesure où l'évaluation est l'élément central indispensable à tous processus d'optimisation, le mémoire de thèse se focalise sur ce problème.

Ainsi, le chapitre 3 propose un processus d'évaluation des propriétés de robustesse reposant sur l'évaluation du niveau de service. Le processus est basée sur deux étapes : modélisation et évaluation. L'approche proposée de modélisation est basée sur des modèles SED stochastiques et plus précisément, nous proposons une structure de modèles d'automates temporisés stochastiques communicants qui permet de modéliser le comportement dynamique d'un ordonnancement et ses interactions avec les perturbations. L'évaluation du niveau de service est ensuite effectuée en utilisant le Model-Checking statistique. L'évaluation permet d'obtenir le niveau de service de l'ordonnancement. Tout au long du chapitre, le processus est illustré sur un cas d'étude d'un atelier Job Shop Flexible perturbé. En fin de chapitre, une implémentation des modèles développés est proposée sur l'outil UppAal SMC.

Le chapitre 4 permet d'identifier et d'étudier les performances du processus d'évaluation proposé. Dans la première partie du chapitre, la généricité du processus vis-à-vis de différents types d'ateliers de production, différents types de distributions de probabilité, différents scénarios de perturbations est évaluée en s'appuyant sur un Benchmark de la littérature. La seconde partie du chapitre s'intéresse à la sensibilité du processus mis en œuvre vis à vis des paramètres et caractéristiques de l'outil de Model-Checking statistique, notamment en ce qui concerne les temps de résolution et le nombre de simulations sur la base d'un sur un plan d'expérience et une analyse statistique. Dans la troisième partie, un plan d'expérience est utilisé pour évaluer la sensibilité de notre approche vis à vis des paramètres d'entrées. Enfin, la dernière partie du chapitre est consacrée à la mise en œuvre d'un plan d'expérience visant à évaluer les capacités de passage à l'échelle du processus proposé.

Le chapitre 5 présente une application du processus dans le cadre d'un scénario industriel. Celui-ci est représentatif de différentes propriétés de robustesse dont le décideur peut avoir besoin pour mettre en œuvre sa stratégie d'ordonnancement.

Le présent manuscrit se conclue par une synthèse des contributions principales de ces travaux de thèse. Ensuite, plusieurs travaux futurs et perspectives de recherche sont proposées.

Chapitre 1

Ordonnancement sous perturbations : Contexte et état de l'art

Sommaire

1.1 Introduction 5
1.2 Ordonnancement de la production : fondements 6
1.2.1 Problème d'ordonnancement 6
1.2.2 Solution d'ordonnancement 7
1.3 Problème d'ordonnancement et Industrie 4.0 8
1.3.1 Challenges de l'Industrie 4.0 8
1.3.2 Contexte général de l'ordonnancement sous perturbations 9
1.4 Ordonnancement sous perturbations 10
1.4.1 Classification des perturbations 10
1.4.2 Stratégies d'ordonnancement sous perturbations 11
1.4.3 Modélisation des perturbations 12
1.5 Ordonnancement proactif soumis à des perturbations stochastiques 17
1.5.1 Performances d'un ordonnancement sous perturbations 18
1.5.2 Modèles pour le problème d'ordonnancement sous perturbations 20
1.5.3 Résolution du problème d'ordonnancement sous perturbations 22
1.5.4 Synthèse de l'état de l'art 26
1.6 Hypothèses et questions de recherche 27
1.7 Conclusion 27

1.1 Introduction

L'ordonnancement de la production est un problème classique, abondamment traité dans la littérature et ce depuis plusieurs décennies. Pour prendre en compte les contraintes de la réalité de l'atelier de production et son environnement, les perturbations sont des éléments à considérer dans la résolution de ce problème. En effet, l'occurrence des perturbations peut impacter fortement la performance de l'ordonnancement implémenté et ainsi engendré des retards au niveau des lignes de production et des pénalités au niveau des commandes clients. Ce constat est d'autant plus vrai si on prend en plus en compte l'évolution des systèmes de production et des stratégies de pilotage de la production qui sont de plus en plus complexes, émanant de l'industrie 4.0.

Dans ce chapitre, le contexte de l'ordonnancement sous perturbations est détaillé. Au delà du contexte de l'industrie 4.0 exprimé dans l'introduction générale, il est important de positionner cette problématique dans le contexte scientifique. Plusieurs travaux, dans la littérature, proposent de traiter l'ordonnancement sous perturbations. L'objectif de l'état de l'art proposé est d'aborder la problématique des perturbations dans différentes communautés et l'évolution des approches de résolution au fil des années.

Avant d'établir l'état de l'art, nous présentons les notions et concepts importants pour bien définir la problématique scientifique abordée dans les travaux de thèse. Enfin, la dernière partie du présent chapitre identifie l'hypothèse de recherche et les questions qui en découle.

1.2 Ordonnancement de la production : fondements

L'ordonnancement et la planification sont deux éléments importants dans l'organisation de la production ou encore dans l'organisation des services [Pinedo, 2012]. La question d'ordonnancement de la production est également une problématique importante dans le cadre de l'aide à la décision. En effet, un processus d'aide à la décision permet de décider quelle est la meilleure solution d'ordonnancement pour l'atelier de production. Les processus d'aide à la décision permettent d'exploiter la connaissance de l'atelier et les outils d'analyses pour offrir au décideur un ensemble d'informations sur l'ordonnancement et ainsi garantir les performances du système de production. Le décideur peut ainsi choisir un ordonnancement adapté à l'atelier de production et aux différentes contraintes qui le caractérisent.

Cette section présente les fondements nécessaires à la définition du problème d'ordonnancement en général et de la prise en compte des perturbations dans ce cadre.

1.2.1 Problème d'ordonnancement

L'ordonnancement de la production peut être défini comme un problème d'optimisation qui vise allouer dans le temps les ressources disponibles aux jobs à réaliser de façon à satisfaire des contraintes (atelier de production, modes d'exécution, ...) et dans l'objectif d'optimiser un critère (durée totale, retard moyen, ...) [Pinedo, 2012].

Un atelier de production est défini par un ensemble d'éléments qui permettent de formuler le problème d'ordonnancement (Définition 1).

Définition 1. *Un atelier de production est défini par :*

- *Un ensemble J de NbJ jobs j à réaliser dans l'atelier.*
- *Chaque job j est associé à un ensemble O_j^J d'opérations o_{jk} définissant une gamme opératoire nécessaire à la réalisation du job. Le nombre d'opérations pour réaliser le job j est noté $NbOp^j$.*
- *Un ensemble R de NbR ressources r, permettant l'exécution des opérations.*
- *Afin d'exécuter l'opération o_{jk} sur la ressource r, une durée d'exécution d_{jkr} est définie.*

La configuration des ressources et des jobs dans un atelier de production permettent de représenter plusieurs types d'ateliers. Quatre types d'atelier sont classiquement définis dans la littérature.

- Atelier Flow Shop : permet basiquement de définir une ligne de production (de plusieurs ressources) imposant un ordre de passage pour chaque job. Ainsi les opérations de chaque job sont soumises à des contraintes de précédence.

— Atelier Job Shop : permet de définir un atelier flexible de production (de plusieurs ressources). Dans ce type d'atelier, chaque job à sa propre gamme opératoire avec des contraintes de précédence entre opérations.
— Atelier Open Shop : n'impose aucun ordre d'exécution, il nécessite néanmoins une contrainte de non-chevauchement des opérations d'un même job afin d'éviter l'exécution de plusieurs opérations du même job en même temps.
— Atelier Hybrid Shop : permet de combiner les caractéristiques de plusieurs type d'atelier (Par exemple un Job Shop suivi d'un Open Shop).

Exemple 1. *Considérons un atelier de production de type Job Shop Flexible. L'atelier est constitué de trois ressources ($R = \{r_1, r_2, r_3\}$). Ces ressources sont qualifiées pour l'exécution de chaque opération, c'est-à-dire 9 opérations. Il y a trois jobs à réaliser ($J = \{j_1, j_2, j_3\}$). Chaque job est associé à une gamme opératoire O_j^J. Pour le job j_1, $O_{j_1}^J = \{o_{11}, o_{12}, o_{13}\}$. Les opérations d'un même job sont soumises à des contraintes de précédence ($o_{11} \rightarrow o_{12} \rightarrow o_{13}$). Les contraintes de précédence et les durées d'exécution des opérations d_{jkr} sont fournies dans la Table 1.1.*

Job	Gamme opératoire		Durée opératoire		
	Opération O_{jk}	Op. précédente	d_{jkr_1}	d_{jkr_2}	d_{jkr_3}
j_1	o_{11}	$\emptyset$	5	5	10
	o_{12}	o_{11}	10	5	6
	o_{13}	o_{12}	3	4	4
j_2	o_{21}	$\emptyset$	10	10	11
	o_{22}	o_{21}	3	4	4
	o_{23}	o_{22}	4	6	4
j_3	o_{31}	$\emptyset$	6	7	5
	o_{32}	o_{31}	6	4	4
	o_{33}	o_{32}	3	4	4

TABLE 1.1 – Données pour l'exemple 1

Un problème d'ordonnancement répond également à un critère d'optimisation. Par exemple, le critère le plus courant, appelé souvent "makespan" et noté C_{max}, permet de minimiser la durée totale de l'ordonnancement. Pour formuler un problème d'ordonnancement, [Graham et al., 1979] propose une classification des problèmes d'ordonnancement en trois champs $\alpha|\beta|\gamma$ représentant respectivement le type d'atelier considéré, les contraintes particulières à prendre en compte, et le critère d'optimisation (Annexe A).

1.2.2 Solution d'ordonnancement

À partir d'un problème d'ordonnancement, des méthodes de résolution sont utilisées afin de rechercher une solution répondant au mieux aux contraintes de l'atelier au critère d'optimisation. Une solution d'ordonnancement est souvent représentée sous forme d'un diagramme de Gantt [Schmidt, 1992] (Figure 1.1). Une solution déterministe d'ordonnancement s considère que les données de l'atelier de production sont certaines et que les ressources sont toujours disponibles. Les informations pouvant être déduites de la solution sont :
— L'affectation des ressources aux opérations et la séquence des opérations sur la ressource ;
— La durée totale prévisionnelle de l'ordonnancement déterministe s notée $C_{max}^{ref}(s)$.

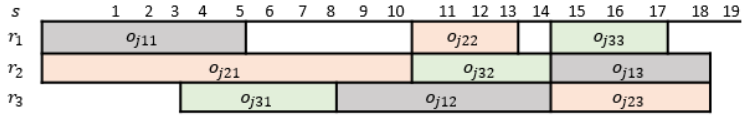


FIGURE 1.1 – Exemple d'ordonnancement

Exemple 2. *A partir de l'atelier job shop flexible de l'exemple 1, un ordonnancement s peut être généré sous forme d'un diagramme de Gantt. À partir de cet ordonnancement (Figure 1.1), les informations suivante sont déduites :*

- *La durée totale de l'ordonnancement* $C_{max}^{ref} = 18UT$ *(UT= unités de temps) ;*
- *Les séquences d'opérations sur les ressources. Par exemple, pour la ressource* r_1 *on a* $o_{j_1 1} \rightarrow o_{j_2 2} \rightarrow o_{j_3 3}$,
- *Les contraintes de précédence de la gamme d'opérations doivent également être vérifiées. Par exemple nous pouvons remarquer que l'opération* $o_{j_2 2}$ *est bien lancée sur la ressource* r_1 *après que l'opération* $o_{j_2 1}$ *ait fini d'être exécutée sur la ressource* r_2.

Une solution d'ordonnancement générée dans un contexte certain ne permet pas de prendre en considération les perturbations pouvant se produire dans l'atelier de production. La prise en compte des perturbations dans le problème d'ordonnancement est donc devenue une question de recherche largement traitée dans la littérature [Mula et al., 2006, Rossit et al., 2019].

1.3 Problème d'ordonnancement et Industrie 4.0

1.3.1 Challenges de l'Industrie 4.0

Avec l'évolution du monde manufacturier vers l'industrie 4.0, les ateliers de production ont connus une transformation progressive allant vers des ateliers à la pointe de la technologie et de plus en plus autonome. En effet, la compétitivité des industriels a permis la mise en points de nouveaux capteurs performants.

Cette évolution de technologies a permis aux systèmes cyber-physiques (CPS) de prendre une place importante dans la mise en place d'ateliers 4.0. Plus exactement, des systèmes cyber-physiques de production (CPPS) permettent d'intégrer ces nouvelles technologies et une nouvelle architecture aux ateliers de productions et ainsi répondre à différentes exigences du concept de l'Industrie 4.0. D'après [Monostori et al., 2016], un système cyber-physique de production se compose d'entités et de sous-systèmes autonomes et coopératifs qui s'interconnectent en fonction du contexte, au sein et au travers de tous les niveaux de production, de la machine jusqu'aux réseaux logistiques. Ce type de systèmes se caractérise une degré accru d'autonomie, de flexibilité et d'agilité et de communications avec l'environnement dans lequel il évolue autorisant une plus grande personnalisation des produits. De plus, l'instrumentation au plus près des équipements (IoT) doit permettre un suivi temps réel des processus beaucoup plus fin et le recueil d'historiques de données relatives au suivi de l'atelier et des performances. [Monostori et al., 2016].

Dans ce contexte, de nombreux challenges se posent à la communauté scientifique. Parmi ces challenges, [Monostori et al., 2016] en a recensé six :

Challenge 1 : Systèmes autonomes et adaptables aux changements pouvant se produire dans l'environnement ;

Challenge 2 : Systèmes de production coopératifs : développement d'algorithmes d'apprentissages, de consensus et de détection distribuée ;
Challenge 3 : Identification et prédiction des systèmes dynamiques ;
Challenge 4 : Ordonnancement robuste pour la gestion des perturbations qui peuvent se produire au cours de l'exécution de ce dernier.
Challenge 5 : Fusion entre les systèmes réels et virtuels ;
Challenge 6 : Interaction Homme-Machine.

L'ordonnancement des ateliers de production fait parti intégrante de ces challenges [Vaidya et al., 2018] en mettant l'accent sur la notion de perturbations induites par une plus grande variabilité de la demande, une plus grande flexibilité de l'outil de production et de manière plus générale par de plus grandes incertitudes sur l'environnement de production. Ainsi, le quatrième challenge introduit l'ordonnancement robuste comme une condition nécessaire au développement des ateliers 4.0. Les travaux présentés dans ce manuscrit s'inscrivent dans le cadre de ce challenge.

1.3.2 Contexte général de l'ordonnancement sous perturbations

L'implantation d'ateliers 4.0 ou de systèmes cyber-physiques de production repose notamment sur une brique technologique liée au traitement de données : le big data et la science des données. En effet, cette brique permet d'obtenir un flux important de données et donc d'informations permettant de construire des modèles de comportement de l'atelier de production. La place de la données dans la mise en place de l'atelier 4.0 est une opportunité non négligeable. Les informations collectées peuvent constituer un support important pour l'aide à la décision au sein de l'atelier 4.0. Dans [Harrison et al., 1996], la décision est définie comme étant un moyen d'évaluation des alternatives pour atteindre un objectif souhaité par le décideur.

Parmi les types de décisions présentes dans l'atelier de production, les décisions opérationnelles permettent de mettre en évidence les décisions liées à l'organisation de la production [Schmidt, 1992]. L'ordonnancement de l'atelier de production est une question qui revient souvent et ce depuis l'évolution de la complexité des ateliers de productions devenant de plus en plus automatisés et flexibles [Pinedo, 2012]. L'implantation d'un atelier 4.0 doit répondre à des contraintes fortes de flexibilité et d'agilité pour pouvoir répondre aux besoins de production qui passe d'une production de masse à une personnalisation de masse. Cette personnalisation de masse induit notamment de fortes incertitudes sur le volume et le mix de la demande. De plus elle raccourcit le cycle de vie des produits, source supplémentaire d'incertitudes.

La prise en compte de ces perturbations dans la problématique de l'ordonnancement est donc souhaitable afin de considérer l'environnement réel du système de production. La figure 1.2 propose un schéma fonctionnel pour l'ordonnancement en contexte perturbé vu comme un problème d'aide à la décision. À partir de l'implantation de l'atelier 4.0, les données de l'atelier sont récoltées puis traitées en utilisant les outils et techniques des "Data Science". Les données traitées permettent de construire des modèles de données qui peuvent être utilisés comme support pour l'aide à la décision. A l'aide d'un processus d'aide à la décision, ces données peuvent être exploitées pour traiter le problème de l'ordonnancement de la production en prenant en compte les perturbations. A l'issue de la résolution de ce problème, le décideur peut exécuter l'ordonnancement.

Dans le cadre de cette thèse, nous nous intéressons à la problématique d'ordonnancement sous perturbations et son rôle dans la garantie de performances pour le décideur.

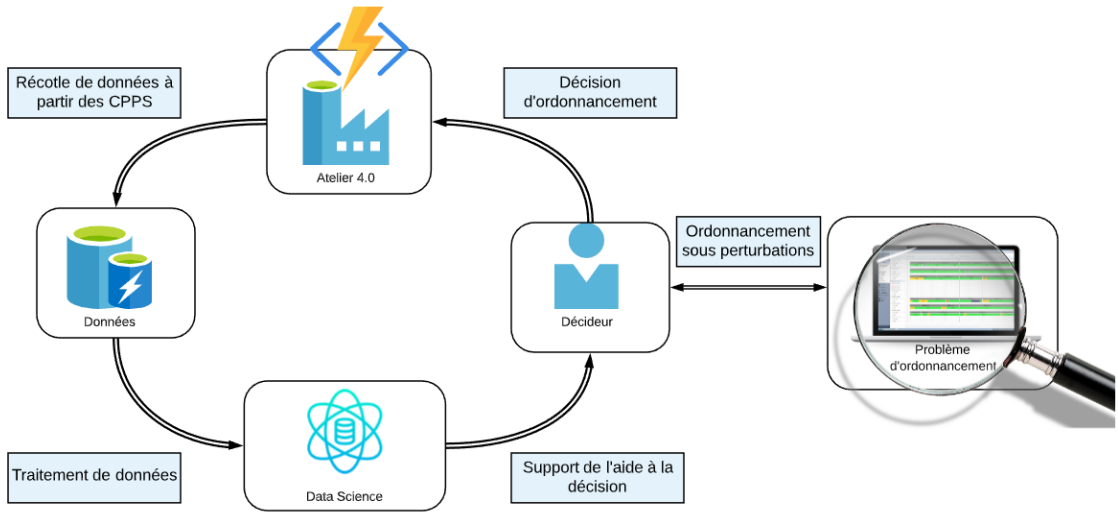


FIGURE 1.2 – Problématique d'ordonnancement dans le contexte de l'industrie 4.0

1.4 Ordonnancement sous perturbations

L'ordonnancement de l'atelier de production est une problématique souvent traitée dans la littérature. Cependant, encore beaucoup de travaux proposent des solutions d'ordonnancement qui considère que l'environnement de production est stable avec des données certaines. Pour s'approcher de plus en plus de la réalité de l'environnement de production réel, il est donc impératif de prendre en considération les perturbations en ordonnancement. Ces perturbations peuvent prendre différentes formes et provenir de différentes sources. Une perturbation est définie comme étant une déviation dans le comportement prédictif du système de production ou dans l'état de fonctionnement normal [Mezgebe, 2020].

Dans ce qui suit, nous présentons un état de l'art de la problématique de l'ordonnancement sous perturbations permettant de constater que cette dernière est une problématique commune à plusieurs communautés de recherche.

Dans un premier temps, nous proposons une classification des perturbations. Nous explorons ensuite les stratégies d'ordonnancement lorsqu'on veut prendre en compte des perturbations. Enfin, nous nous intéresserons aux perturbations à travers leur classification et leurs modèles permettant de les définir et les prendre en compte.

1.4.1 Classification des perturbations

Une perturbation peut se manifester sous la forme d'une incertitude ou d'un aléa. Dans [Mula et al., 2006], un état de l'art traite la question de l'incertitude dans le domaine de la production. Dans ce contexte, [Galbraith, 1973] a défini l'incertitude comme étant la différence entre la quantité d'informations requises pour exécuter une tâche et la quantité d'informations présentes. Pour déployer cette définition dans le contexte industriel, [Ho, 1989] a catégorisé les incertitudes selon leur source : incertitudes environnementales et incertitudes dues au système de production. L'incertitude environnementale étant externe au processus de production et l'incertitude du système étant interne au processus de production. La définition des aléas est reliée à l'occurrence d'événements incontrôlables ou imprévus dans l'atelier de production [Aubry et al., 2009], comme

Source \ Type	**Incertitude**	**Aléa**
Environnement	Demande client Durée de livraison approvisionnement ...	Rupture d'approvisionnement Arrivée de commande urgente ...
Système	Durée d'exécution d'opérations Durée de réparation ...	Panne machine Produit défectueux ...

TABLE 1.2 – Classification des perturbations

la panne ressource ou encore la rupture de stock. Nous pouvons catégoriser les aléas suivant leurs sources aussi et donc la catégorisation introduite par [Ho, 1989] peut être adaptée aussi pour les perturbations de type aléas. À partir de ces définitions, nous proposons de classifier les perturbations selon leur type et leur source (Table 1.2).

1.4.2 Stratégies d'ordonnancement sous perturbations

Le problème de la prise en compte des perturbations pour l'ordonnancement de production n'est pas nouveau dans la littérature. Pour prendre en compte les perturbations, trois approches principales sont définies dans la littérature [Billaut et al., 2010] l'approche proactive, l'approche réactive et l'approche proactive/réactive.

Approche proactive

L'approche proactive consiste généralement à trouver une solution optimale de l'ordonnancement en étant hors ligne et en prenant en compte les perturbations *a priori*. Les approches proactives peuvent avoir deux types d'objectifs. Le premier est d'évaluer la performance vis-à-vis des perturbations d'une solution d'ordonnancement prédictive existante ou de plusieurs solutions afin de garder la meilleure (aide à la décision). Le deuxième est d'intégrer les perturbations dans la définition du problème afin de trouver une solution apte à absorber les perturbations.

Approche réactive

L'approche réactive consiste à trouver une solution d'ordonnancement au fur et à mesure de la production (en ligne). Dans le cas où il est difficile de prédire le comportement des perturbations *a priori* de la production, cette approche permet de réagir en temps réel et de trouver une solution en temps réel.

Approche proactive-réactive

Cette approche regroupe les approches proactives et réactives. Il s'agit, dans un premier temps, de trouver la solution de façon proactive. Dès l'arrivée d'une perturbations, une action réactive est effectuée de sorte à ce qu'une partie de la solution soit ré-ordonnancée pour intégrer l'effet de la perturbation ou, si l'impact de la perturbation est trop important, de calculer une nouvelle solution en ligne. L'approche proactive-réactive permet de réagir aux perturbations qui n'ont pas été prises en compte de façon proactive mais aussi de prendre en compte les perturbations qui remettent en cause la solution d'ordonnancement. L'aspect réactif de l'approche permet de reconfigurer l'ordonnancement à partir du moment où la perturbation arrive dans l'atelier.

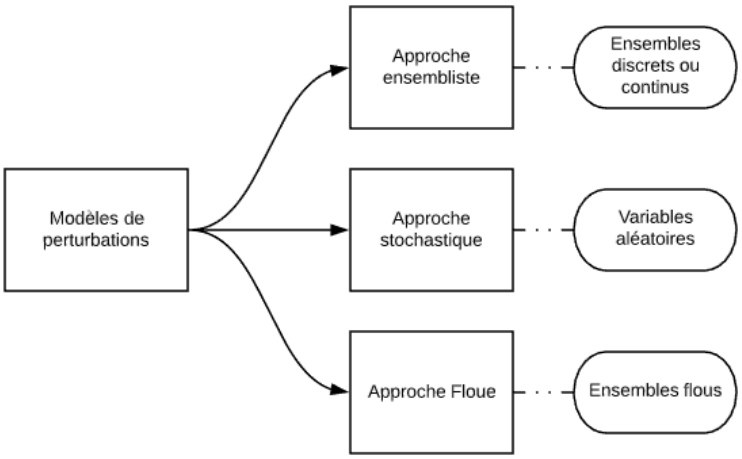


FIGURE 1.3 – Classification des modèles de perturbations

Dans le cadre de cette thèse nous nous intéressons à l'approche proactive afin de prendre en compte d'une manière préventive l'occurrence de perturbations. L'objectif est alors de prévoir la capacité d'un ordonnancement à faire face à des perturbations. L'approche proactive impliquant de prendre en compte les perturbations *a priori*, il faut être capable de prendre en compte ces perturbations puis de proposer un modèle de ces perturbations. L'état de l'art qui suit s'intéresse donc aux approches de modélisation des perturbations proposées dans la littérature.

1.4.3 Modélisation des perturbations

Afin de prendre en compte les perturbations en ordonnancement, il est impératif de pouvoir les représenter. Dans la littérature plusieurs approches ont été proposées. Nous recensons les trois approches les plus utilisées (Figure. 1.3) [Li and Ierapetritou, 2008] [Dias and Ierapetritou, 2016] :

— Approche ensembliste (par Intervalle ou par Scénario) : les perturbations sont représentées dans des ensembles finis. Chaque valeur de cet ensemble peut constituer un scénario de perturbation possible.
— Approche floue (Fuzzy) : les perturbations sont associées à des ensembles flous et donc sont liées à des fonctions d'appartenance.
— Approche stochastique (Probabiliste) : les perturbations sont des variables aléatoires suivant une distribution de probabilité définie.

1.4.3.1 Approche ensembliste

L'approche ensembliste permet de représenter les perturbations en se basant sur les données du décideur considérant l'indisponibilité de données historiques ou la faible fiabilité de ces dernières. Les perturbations sont le plus souvent représentées sous forme d'ensembles finis ou d'intervalles bornés contenant les valeurs possibles de la perturbation. Ces valeurs construisent ensuite l'ensemble de scénarios discrets présentant les situations possibles si la perturbation est un aléa ou les valeurs possibles si la perturbation est une incertitude.

Dans la littérature, plusieurs travaux représentent les perturbations en utilisant la modélisation ensembliste. Dans [Abdeddaïm et al., 2006], les valeurs des durées d'exécution incertaines sont comprise dans un intervalle borné $[l, u]$ de sorte que chaque valeur de cet intervalle présente une instance du problème qui peut être traitée avec une approche déterministe.

Dans [Aubry et al., 2009], les incertitudes sur la demande client sont présentées dans un ensemble fini P d'instances.

Dans [Briand et al., 2007], plusieurs incertitudes sur les durées sont prises en compte (durées d'exécutions des jobs, dates de début des jobs et dates dues des jobs). Les auteurs présentent ces perturbations sous forme de couple $< I, C >$ modélisant un ensemble d'intervalles I et un ensemble de contraintes permettant d'établir un lien entre les différentes valeurs des intervalles. Plus précisément, ils considèrent la combinaison des trois intervalles suivants : la date de début d'un job i est comprise dans l'intervalle $r_i \in [\underline{r_i}, \overline{r_i}]$, l'intervalle de la date due $d_i \in [\underline{d_i}, \overline{d_i}]$ et la durée d'exécution d'un job i tel que $p_i \in [\underline{p_i}, \overline{p_i}]$. Un scénario est construit à partir de trois valeurs appartenant aux intervalles décrits avant avec la contrainte suivante : "$r_i \leq p_i \leq d_i$". Pour illustrer l'approche ensembliste, l'exemple (3) est présenté dans ce qui suit.

Exemple 3. *Considérons un atelier à machine unique (Single machine workshop) ainsi que trois jobs j_1, j_2 et j_3 ayant des durées d'exécution incertaines. En se basant sur l'approche ensembliste, nous considérons que le décideur dispose d'un ensemble d'intervalles pour modéliser les durées incertaines (Table. 1.3). Ces intervalles permettent éventuellement au décideur d'extraire un ensemble de scénarios qu'il souhaite analyser (Table. 1.4). À partir des intervalles, on pourrait identifier une infinité de scénarios.*

Job	Durée	Intervalle
j_1	p_1	$[3, 5]$
j_2	p_2	$[2, 6]$
j_3	p_3	$[10, 13]$

TABLE 1.3 – Exemple de la modélisation par approche ensembliste

Scénario	p_1	p_2	p_3
s_1	3	2	10
s_2	4	4	11
s_3	5	6	13

TABLE 1.4 – Approche ensembliste : Construction de scénarios

La construction de scénarios ne peut relever que du décideur. En effet, les scénarios représentent souvent de la connaissance de ce dernier par rapport à son atelier de production et son environnement. Les scénarios peuvent représenter le pire cas (comme le scénario s_3 (Table. 1.4) ou le meilleur cas (le scénario s_1).

L'approche ensembliste est l'une des plus utilisées dans la littérature [Peng et al., 2016] [Feng et al., 2016]. Elle permet de refléter les contraintes économiques, technologiques et autres contraintes de l'environnement et ce du fait qu'elle repose sur les données et modèles du décideur. Le décideur est dans ce cas un élément actif du processus de l'ordonnancement. Dans ce cadre,

le décideur structure les perturbations quand aucune donnée n'est disponible. Les scénarios de perturbations important pour le décideur sont forcément traités. Le rôle du décideur, d'après [Kouvelis and Yu, 1997] [Daniels and Carrillo, 1997], consiste à :

— Identifier les éléments connus de l'environnement d'ordonnancement : les événements qui vont se produire et qui auront sûrement une influence sur l'ordonnancement ;
— Formaliser les liens et contraintes existant entre les différents éléments de l'environnement et spécialement ceux entraînant de l'incertitude ou une modification sur l'ordonnancement.

1.4.3.2 Approche floue

L'approche floue est basée sur la logique floue, les perturbations sont prises en compte dans des ensembles flous ayant des fonctions d'appartenance [1]. Dans la Figure. 1.4, un exemple de fonction d'appartenance liée à l'ensemble D est proposé.

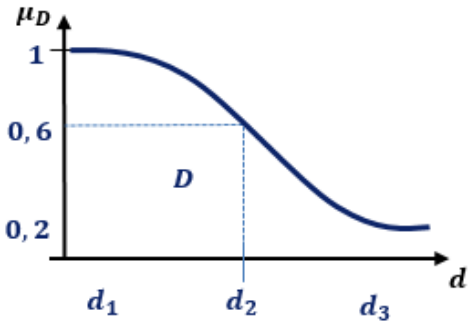


FIGURE 1.4 – Fonction d'appartenance de D

Cette approche est associée dans la littérature avec un type de problème précis le "Fuzzy Scheduling problem". Dans [Palacios et al., 2016], les auteurs présentent une approche floue pour modéliser les incertitudes sur les durées d'exécution. L'approche floue peut être une extension de l'approche par ensembliste. Lorsque certaines valeurs d'un intervalle peuvent être plus plausibles que d'autres, cet intervalle peut être étendu en ensemble flou.

Dans [Dubois et al., 2003], les auteurs considèrent différentes incertitudes du système de production notamment sur la date de début r_i, la date due d_i et la durée d'exécution p_i des jobs. Pour chaque paramètre incertain, un ensemble flou et une fonction d'appartenance sont définis. Ces fonctions sont construites en suivant les contraintes et les caractéristiques de l'atelier de production. La fonction d'appartenance de la date due est construite en fonction de la satisfaction

1. Soit D un ensemble flou ayant comme éléments d_1, d_2, d_3.
Alors : $D = \{\frac{1}{d_1}, \frac{0.6}{d_2}, \frac{0.2}{d_3}\}$ où $\{1, 0.6, 0.2\}$ représentent les degrés d'appartenance des éléments d_1, d_2 et d_3. Un degré d'appartenance μ appartient à l'intervalle $[0, 1]$.
La logique floue est basée sur la théorie des possibilités. Par exemple, pour l'ensemble D, l'élément d_1 appartient sûrement à l'ensemble ($\mu(d_1) = 1$) alors que d_2 appartient vraisemblablement à D ($\mu(d_2) = 0.6$). Tandis qu'il est peu possible que d_3 appartienne à D ($\mu(d_3) = 0.2$ s'approche de 0).
A partir de cet ensemble une fonction d'appartenance peut être construite, elle peut être ascendante ou descendante comme elle peut prendre la forme d'une distribution gaussienne ou autre. Ceci dépend du sens physique de D et de son interprétation.

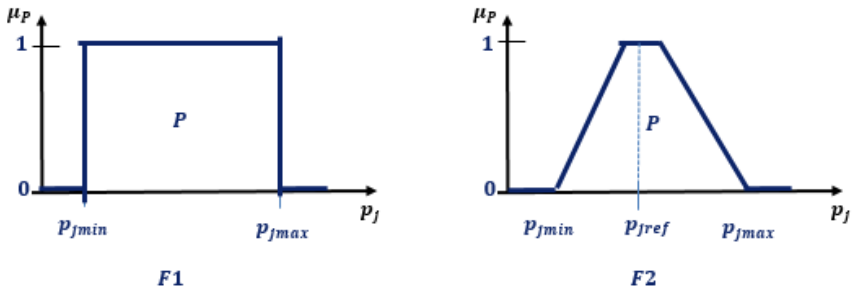


FIGURE 1.5 – Exemple de fonctions d'appartenance de P

client : "Plus la date due est dépassée plus la satisfaction client est basse ". Les autres fonctions sont construites de la même manière et l'interprétation de ces fonctions est liée à la notion de satisfaction (exemple : $sat(t_\omega) = \mu_D(t_\omega)$). Pour traiter l'ensemble des incertitudes, les auteurs utilisent les opérations sur les ensembles flous afin de combiner les fonctions d'appartenance et d'en déduire une fonction de satisfaction globale.

Pour illustrer la description floue des perturbation, l'exemple 4 est présenté.

Exemple 4. *Considérons le même exemple de machine unique que celui de la section précédente. Si la perturbation considérée est l'incertitude sur les durées d'exécution des jobs alors nous pouvons considérer qu'un minimum d'information est disponible. Pour modéliser cette perturbation avec l'approche floue, il suffit d'étendre l'intervalle $[p_{jmin}, p_{jmax}]$ à un intervalle flou ou les valeurs sont plus au moins possible. La fonction d'appartenance peut suivre les logiques suivantes.*

- *Si la valeur de la durée est entre p_{jmin} et p_{jmax} alors son degré d'appartenance à l'ensemble flou des durées d'exécution P est égale à 1 ($\mu_D(d) = 1$), sinon en dehors de ces valeurs le degré d'appartenance est égale à 0 (Figure 1.5 "F1 ").*
- *Si l'on considère que dans l'intervalle $[p_{jmin}, p_{jmax}]$ existe une valeur de référence p_{jref} alors le degré d'appartenance de $p_j = p_{jref}$ est égale à 1, si la valeur de p_j est $p_{jmin} \leq p_j \leq p_{jref}$ la fonction d'appartenance est croissante et si $p_{jref} \leq p_j \leq p_{jmax}$ alors la fonction est décroissante (Figure. 1.5 "F2 ").*

L'approche floue permet de modéliser les perturbations à partir de descriptions linguistiques et ainsi de construire de l'information autour de ces dernières. L'avantage de cette approche est la prise en compte du degré de satisfaction dans l'atelier et donc des exigences du client ou du décideur. Elle demande néanmoins un travail important de construction et d'interprétation des fonctions.

1.4.3.3 Approche stochastique

L'approche stochastique est souvent adoptée pour modéliser les perturbations quand l'information et les données historiques sont disponibles. Les perturbations sont considérées comme étant des variables aléatoires suivant des distributions de probabilité connues.

Dans [V.J.Leon et al., 1994, Beck and Wilson, 2004, Goren and Sabuncuoglu, 2008, Al-hinai and Elmekkawy, 2011, Davenport et al., 2014] les pannes ressource sont modélisées comme des variables aléatoires suivant des lois de distribution différentes (exponentielle, normale, uniforme, et gamma). Pour une loi uniforme, cela traduit le fait que toutes les ressources de l'atelier ont

la même probabilité de tomber en panne. Dans [Cai et al., 2009, Elyasi and Salmasi, 2012], les incertitudes sur les durées d'exécution sont modélisées comme des variables aléatoires suivant des lois de distribution différentes (exponentielle, normale et gamma).

Le choix de la distribution est souvent justifié par la quantité d'informations disponible dans l'atelier ainsi que l'interprétation de ces dernières dans le modèle et de son impact sur la résolution du problème. La distribution exponentielle est souvent utilisée car permettant de faciliter la résolution du problème.

L'approche stochastique est illustrée dans l'exemple (5).

Exemple 5. *Considérons l'exemple de la machine unique, les durées d'exécution des opérations o_{j1}, o_{j2} et o_{j3} suivent une distribution de probabilités dans un intervalle $[d_{jkr}^{min}, d_{jkr}^{max}]$. Cette distribution peut avoir différentes formes, les fonctions de répartition associées sont données dans les figures suivantes : distribution uniforme continue ou discrète (Figures 1.7 et 1.6) ou distribution exponentielle (Figure 1.8).*

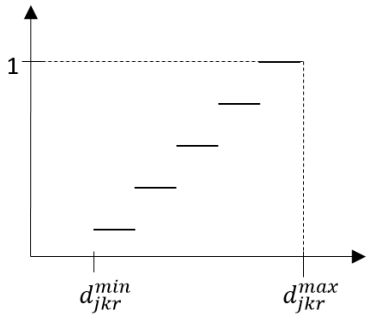


FIGURE 1.6 – Fonction de répartition uniforme discrète

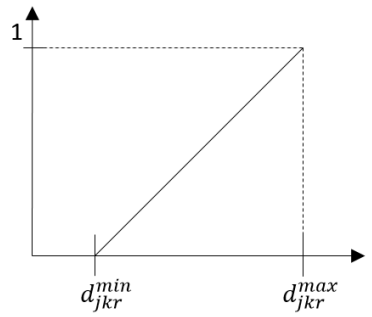


FIGURE 1.7 – Fonction de répartition uniforme continue

L'avantage majeur de cette approche est le fait qu'elle permet une exploitation des données existantes dans l'atelier et du coup profiter de des informations collectées concernant l'état des ressources ou encore l'historique de production.

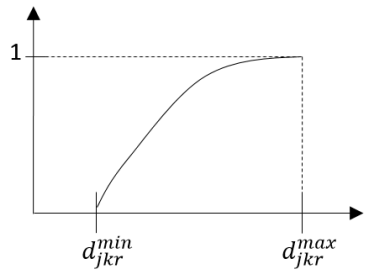


FIGURE 1.8 – Fonction de répartition exponentielle continue

1.4.3.4 Modélisation des perturbations : synthèse

Le choix du modèle des perturbations dépend de la quantité d'informations disponible et du type d'atelier de production, en effet dans le contexte de l'industrie 4.0, les ateliers ont une architecture basée sur les systèmes de production cyber-physiques et des composants intelligents ce qui implique une collecte de données plus précise et structurées. Dans ce cas, le modèle stochastique semble particulièrement pertinent pour prendre en compte les perturbations.

Dans ce qui suit, l'état de l'art se focalise sur les approches proactives pour traiter le problème d'ordonnancement en considérant que les perturbations sont décrites à l'aide de modèles stochastiques.

1.5 Ordonnancement proactif soumis à des perturbations stochastiques

La communauté qui s'intéresse le plus à l'ordonnancement est celle de la Recherche Opérationnelle (RO).

Une autre communauté, celle des Systèmes à Èvénements Discrets (SED) propose des approches pour traiter le problème d'ordonnancement sous perturbations.
Dans le domaine de la sûreté et de la sécurité, les SED sont généralement utilisés pour évaluer la fiabilité, la disponibilité, la maintenabilité. Ainsi, ils ont prouvé leur efficacité dans la modélisation des systèmes stochastiques et dynamiques. Les systèmes à événements discrets présentent ainsi des atouts considérables faisant d'eux de bons candidats pour résoudre le problème d'ordonnancement en général [Cassandras and Lafortune, 2009]. Plusieurs approches ont prouvé leur efficacité dans la génération d'ordonnancement en considérant un environnement de production certain [Panek et al., 2006, Subbiah and Engell, 2010, Marangé et al., 2011, Marangé et al., 2016]. La prise en compte des perturbations a été également traitée en utilisant les SED [Lefebvre and Mejia, 2018] [Cherif et al., 2019] [Abdeddaïm et al., 2006].

Pour prendre en compte les perturbations en ordonnancement, les approches proposées dans la littérature se basent sur deux étapes : la modélisation et la résolution (Figure 1.9). La modélisation permet de décrire le problème et les perturbations identifiées. En se basant sur les modèles, des techniques de résolution permettent de répondre à la problématique traitée. Dans ce qui suit, nous nous intéressons tout d'abord à la description des perturbations et aux différentes approches permettant de prendre en compte les perturbations en ordonnancement. Un état de l'art s'intéressent aux approches de modélisation et de résolution les plus communes dans la littérature en

commençant par les approches RO puis les approches SED. L'état de l'art présenté dans la suite de ce chapitre est construit en trois parties explorant chacune une question :

1. Quel est la performance recherchée dans l'ordonnancement sous perturbations ?
2. Comment modéliser le problème d'ordonnancement sous perturbations ?
3. Comment résoudre le problème d'ordonnancement sous perturbations ?

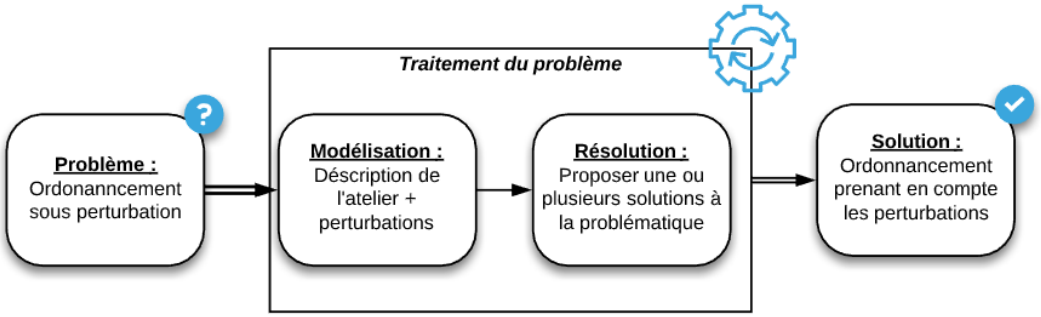


FIGURE 1.9 – Méthode de traitement de la problématique

1.5.1 Performances d'un ordonnancement sous perturbations

La problématique d'ordonnancement de la production est largement discutée dans la littérature. Les approches classiquement connues sont basées sur la recherche opérationnelle (RO), leur objectif principal est l'optimisation des performances de l'ordonnancement.

Avant de résoudre le problème de l'ordonnancement sous perturbations, l'objectif de cette résolution doit être identifié. En effet, pour traiter ce problème il est important de choisir le critère à évaluer ainsi que la métrique à mesurer (Figure 1.10).
Dans la littérature, nous avons identifié trois principales catégories de performances.

La première consiste à la mesure de l'espérance d'un critère. Parmi les critères choisis, le Makespan (C_{max}) est souvent utilisé dans le cadre de cette mesure. En effet, parmi les premiers travaux prenant en compte les perturbations comme variables aléatoires, [V.J.Leon et al., 1994] proposent une approche RO ayant pour objectif d'évaluer la robustesse de l'ordonnancement sous perturbations en se basant sur la variabilité entre l'espérance du Makespan perturbé et celle du Makespan attendu. Dans les travaux de [Abdeddaïm et al., 2006], les auteurs proposent une approche SED qui permet également de mesurer l'espérance de la durée totale de l'ordonnancement. Dans [Goren and Sabuncuoglu, 2008], les auteurs se basent sur les concepts de robustesse et de stabilité pour proposer également une mesure de l'espérance du Makespan. Les même travaux identifient la mesure de l'espérance à travers une fonction permettant de choisir entre différents critères notamment le retard total T_{max}. D'autres critères peuvent être considérés dans le calcul de l'espérance. Dans [Cai et al., 2009], la mesure proposée par les auteurs concerne l'espérance du temps d'occupation des ressources symbolisé par T_i. Cette mesure est proposée dans le cadre spécifique d'un ordonnancement soumis à des pannes ressources.

La deuxième mesure possible considère la probabilité d'atteindre une performance ou de satisfaire une contrainte. Dans [Beck and Wilson, 2004], les auteurs proposent une mesure probabiliste

permettant le mesurer si le Makespan de l'ordonnancement sous perturbation est supérieure à une durée définie. Dans ce cadre, les auteurs expriment également une dérivée de cette mesure permettant d'identifier la durée minimale possible pour l'ordonnancement sous perturbation. Dans [Dauzère-Pérès et al., 2010], les auteurs proposent également de mesurer la probabilité que le Makespan de l'ordonnancement sous perturbations soit inférieure ou égale à une deadline. Ils introduisent également la notion du niveau de service qui permet de comparer la mesure de probabilité à un seuil de service limité.

Traiter le problème de l'ordonnancement sous perturbations peut également dépendre de la mesure d'un critère avec un objectif d'optimisation bien défini. Dans [Elyasi and Salmasi, 2012], les auteurs proposent une approche d'optimisation permettant de prendre en considération les perturbations tout en minimisant le nombre de jobs en retard ($min(\sum U_j)$. Dans [Rahimi et al., 2018] l'objectif principal ces travaux est de trouver la solution d'ordonnancement ayant un Makespan optimale et ce malgré l'occurrence des perturbations ($min(C_{max})$).

Plusieurs autres mesures sont proposées dans la littérature. Dans le cas de pannes ressource, nous retrouvons également des mesures de maintenabilité et de fiabilité comme dans [Ballarini et al., 2011]. Les auteurs proposent de prendre en compte les pannes ressources ainsi que leur durée de maintenance stochastique avec l'objectif de garantir des performances de maintenabilité telles que le temps moyen de bon fonctionnement avant panne $MTTF$.

Dans [Davenport et al., 2014], les auteurs proposent de prendre en considération des perturbations de type pannes ressource avec un objectif de minimisation de la somme des jobs en retard. Dans ce cas, les auteurs proposent de comparer le retard total d'ordonnancement prédictif et celui d'ordonnancement simulé (notée $TARD(p, s)$). Cette mesure permet ainsi d'avoir une moyenne du retard simulé en considérant l'occurrence des pannes.

Dans [Ivanov et al., 2016], les auteurs proposent une approche basée sur l'atteignabilité pour évaluer la robustesse d'ordonnancements sous la perturbations de capacité des ressources dans un atelier Flow Shop Flexible. Les deux objectifs considérés sont le débit et le retard total. Pour prendre en compte les perturbations, une mesure de robustesse est définie se basant sur un index de risque que l'ordonnancement ne soit pas achevé en présence de ces dernières. La notion de mesure du risque revient également dans [Lefebvre and Mejia, 2018].

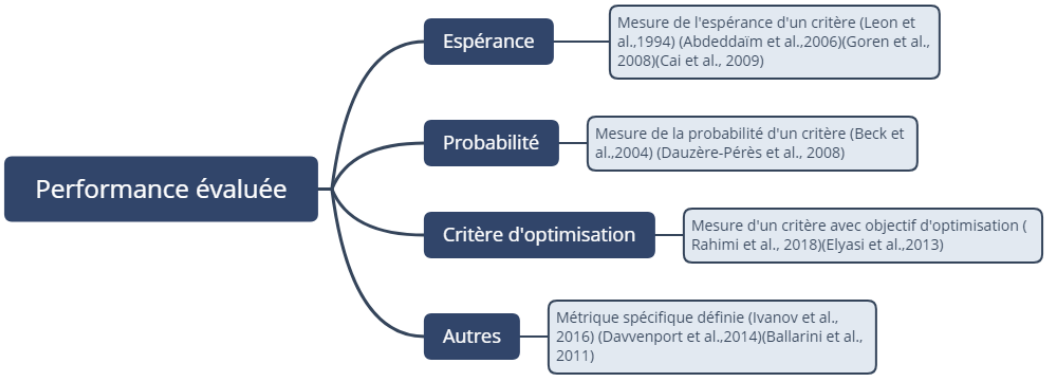


FIGURE 1.10 – Synthèse des performances de l'ordonnancement sous perturbations

1.5.2 Modèles pour le problème d'ordonnancement sous perturbations

Les approches de modélisation proposées pour les problèmes d'ordonnancement peuvent être classifiées en deux types de modèles : non stochastique et stochastique.

Les modèles non stochastiques sont ceux qui permettent de modéliser le problème sans avoir recours aux caractéristiques stochastiques du problème. Parmi ces modèles, nous retrouvons la programmation mathématique, la programmation floue et les SED déterministes. Les modèles stochastiques, permettent une modélisation probabiliste du problème comme la programmation stochastique, le chance constraint programming ou encore les SED stochastiques (Figure 1.11).

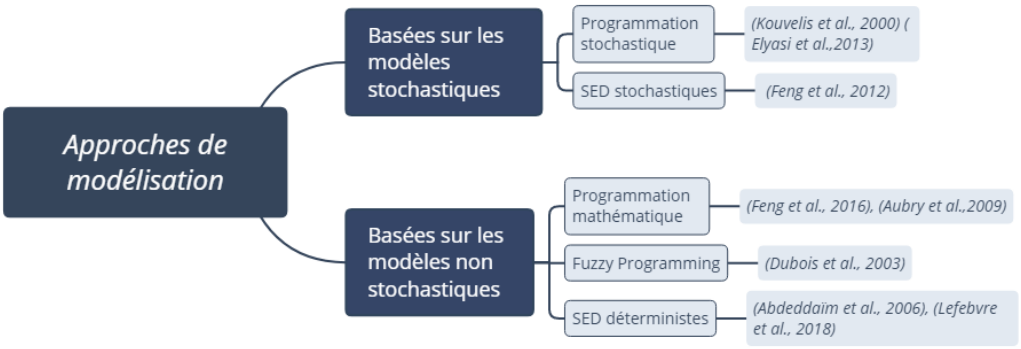


FIGURE 1.11 – Synthèse des approches de modélisation du problème

1.5.2.1 Approches non stochastiques

Les approches non stochastiques sont les approches qui permettent de modéliser le problème d'ordonnancement sous perturbations sans pour autant utiliser des modèles probabilistes. Les approches non stochastiques recensées jusqu'ici sont d'un point de vue recherche opérationnelle la programmation mathématique linéaire ou non linéaire (MILP, MINLP, etc.). Dans la logique floue, la programmation floue non probabiliste est définie (basée sur la théorie des possibilités).

Programmation mathématique

La programmation mathématique est une méthode basée sur les systèmes d'équations permettant de déterminer l'ordonnancement optimal. Une méthode qui en plus d'assurer la faisabilité de l'ordonnancement, permet une modélisation accessible surtout pour les systèmes de production définis dans la littérature. L'utilisation de la programmation mathématique permet de trouver la solution dans un temps CPU raisonnable pour des problèmes de taille petite à moyenne. L'histoire de la programmation mathématique a connu une tournure remarquable due à l'augmentation de la capacité des ordinateurs et leur capabilité à résoudre les systèmes d'équations.
Ceci a incité les ingénieurs à l'utiliser et la développer dans le cadre de la planification et l'organisation de la production. Parmi les programmes mathématiques les plus utilisé, nous retrouvons le MILP (Mixed-Integer Linear Programming).

Ce type de modélisation consiste tout d'abord à définir les caractéristiques et les contraintes du problème, ainsi en déduire les variables du problème (ressources, jobs, durée d'exécution, etc.) ainsi que les variables de décision (qui vont permettre la modélisation de l'objectif du

problèmes). L'élément important du modèle est la fonction objectif [V.J.Leon et al., 1994] [Goren and Sabuncuoglu, 2008].

La *programmation floue (Fuzzy programming)* figure parmi les approches de modélisation non stochastiques. Dans le cas de la programmation floue, les données d'entrées du problème sont considérées comme étant des nombres flous suivant une fonction d'appartenance et dépendant d'ensembles flous. Dans [Sahinidis, 2004], deux types de programmations floues sont définies comme étant les plus connus dans le domaine du flou : la programmation flexible (*flexible programming*) et la programmation basée sur les possibilités (*possibilistic programming*).

SED déterministes

Dans le domaine des systèmes à événements discrets (SED), plusieurs modèles ont fait leur preuve. [Panek et al., 2006,Marangé et al., 2011,Marangé et al., 2016] avaient déjà utilisé ce type de modèle pour l'ordonnancement déterministe.

Dans [Abdeddaïm et al., 2006], les auteurs proposent d'utiliser les automates temporisés tel que défini dans [Alur and Dill, 1994] pour modéliser le problème d'ordonnancement déterministe dans un premier temps puis ils prennent en compte les incertitudes sur les durées d'exécution modélisé par une approche par intervalle. Les auteurs justifient le choix de ce type de modélisation par la capacité de ce dernier à représenter d'une manière réaliste et fidèle le comportement dynamique des systèmes de production. Il permet donc de formuler de manière naturelle les propriétés des systèmes distribués contenant des éléments interactifs. Les automates temporisés sont également utilisés par [Rahimi et al., 2018] pour modéliser la problématique d'ordonnancement sous perturbations.

Un autre formalisme de représentation des SED, les réseaux de Petri (RdP), peut également être utilisé pour modéliser le problème d'ordonnancement dans un environnement incertain [Daoui and Lefebvre, 2017]. Leur variante temporisée (RdP temporisés) est utilisée par [Lefebvre and Mejia, 2018] et [Lefebvre and Daoui, 2020]. Dans [Lefebvre and Mejia, 2018], les auteurs proposent de modéliser le problème d'ordonnancement robuste sous perturbations en considérant que la perturbation à laquelle le système de production doit faire face est l'arrivée d'événements incontrôlables représentée dans le RdP temporisé par des transitions incontrôlables. Ce formalisme de modélisation permet une grande adaptabilité aux autres problèmes du même type. Dans [Lefebvre and Daoui, 2020], les auteurs proposent d'utiliser les RdP temporisés partiellement contrôlés (PC-TPN) pour modéliser le problème d'ordonnancement dans un contexte temporisé et incertain par l'occurrence d'événements incontrôlable qui risquent de mettre en cause la validité de la séquence d'ordonnancement.

1.5.2.2 Approches stochastiques

La deuxième catégorie d'approches regroupe les approches stochastiques. Ce type d'approches permet une prise en compte stochastique du problème d'ordonnancement dans des modèles intégrant l'aléatoire, du calcul de la probabilité. L'utilisation des modèles stochastiques est intéressante quand les perturbations sont considérées aussi par une modélisation stochastique. Nous relevons trois approches stochastiques à partir de la littérature : la programmation stochastique, le chance constraint programming (programmation basée sur des contraintes probabilistes) et les SED stochastiques.

Programmation stochastique

La *programmation stochastique* est une version probabiliste des programme linéaires déterministes tels que ceux définis dans la section précédente. L'origine de la programmation mathé-

matique remonte aux années 50. Les paramètres réels des systèmes étant dans la plupart du temps aléatoire, ce type de programmation permet d'intégrer cette nature de paramètres à des programmes mathématiques linéaires ou non linéaires modélisant des fonctions probabilistes liées à des distributions de probabilités. Plusieurs ouvrages permettent de se familiariser avec ce type de programmation et offrent une vision sur le type de problème que cette programmation permet de modéliser [Kall et al., 1994] [Birge and Louveaux, 1997].

Dans [Sahinidis, 2004], nous retrouvons une catégorisation des types de programmation stochastique existants dans la littérature. Dans la littérature, la programmation stochastique est parmi les approches les plus utilisées [Mula et al., 2006]. Dans [Kira et al., 1997], les auteurs proposent un modèle stochastique linéaire pour modéliser le problème de planification de la production d'un atelier multi-produits, multi-périodes avec de l'incertitude sur la demande client. Dans [Cai et al., 2009], les auteurs proposent également une programmation mathématique stochastique afin de prendre en considération les perturbations dans le problème d'ordonnancement.

SED Stochastiques

Les modèles *SED stochastiques* sont une autre alternative de modélisation permettant de prendre en compte le problème d'ordonnancement sous perturbations. Les modèles SED stochastique, modélisent non seulement l'aspect dynamique mais aussi les caractéristiques stochastiques du comportement du système de production. Plusieurs langages de modélisation permettent ceci (Réseaux d'automates stochastiques, Automates temporisés stochastiques, Réseaux de Pétri stochastiques, Chaînes de Markov, etc.). Les caractéristiques probabiliste de ces modèles prennent en compte les perturbations modélisées avec l'approche stochastique. Ces modèles sont souvent adoptés dans la littérature dans le domaine du contrôle et vérification des procédés de production et de leur fiabilité [Castañeda et al., 2011]. Dans [Feng et al., 2012], les auteurs proposent une modélisation stochastique basée sur les Chaînes de Markov Continues pour présenter un système dynamique de production.

Dans [Ballarini et al., 2011], les auteurs proposent d'utiliser les réseaux de Petri stochastiques pour modéliser le comportement d'un atelier de production flexible sujet aux perturbations. Les auteurs ont fait le choix de ce langage afin de permettre la modélisation de toutes les caractéristiques stochastiques de l'atelier de production et de prendre en compte son comportement.

1.5.3 Résolution du problème d'ordonnancement sous perturbations

Dans cette section les méthodes de résolution sont explorées. Elles peuvent être classifiées en deux catégories : exactes et approchées. Les approches exactes permettent une résolution analytique des modèles afin d'obtenir des résultats précis (sans marge d'erreur).Les méthodes approchées permettent une résolution des modèles basées sur l'approximation, en d'autres termes, la solution obtenue n'est pas optimale. La marge d'approximation diffère d'une méthode à une autre (Figure 1.12).

1.5.3.1 Méthodes exactes

Algorithmes exacts

Un algorithme de résolution exacte classiquement utilisé en RO est l'algorithme "*Branch and Bound*" [Lawler and Wood, 1966]. Le "Branching and bounding" ou "*Séparation et évaluation progressive*" est une méthode de résolution structurée intelligemment afin de parcourir tout l'espace des solutions admissibles mais de manière non nécessairement explicites afin d'identifier la

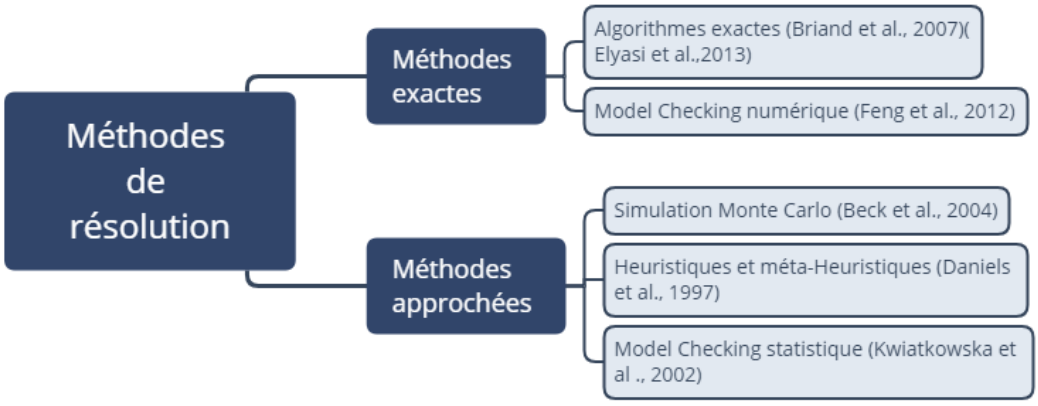


FIGURE 1.12 – Synthèse des approches de résolution des modèles

solution optimale. La qualité de ces algorithmes reposent sur leur capacité à pouvoir éliminer des solutions ayant des caractéristiques communes sans les évaluer nécessairement.

L'algorithme de type Brand and Bound est générique dans sa définition mais doit être adaptée si on souhaite l'appliquer à un problème. Une autre famille d'algorithme contient les algorithmes dédiés, dont la structure et les mécanismes sont liés à la nature du problème. [Elyasi and Salmasi, 2012] utilise l'algorithme de Moore-Hodgson afin de traiter le problème $1||\sum U_j$.

Model checking probabiliste numérique

Le Model checking numérique est une méthode de résolution utilisée dans le cas où le problème est modélisé par des langages des systèmes à évènements discrets. La méthode de résolution est celle basée sur la vérification formelle par model checking.

Le principe du model checking est de vérifier la satisfaction d'une propriété (contenant l'objectif de résolution) par le modèle. L'objectif est d'explorer l'espace d'états pour trouver un ou plusieurs chemins permettant de satisfaire l'objectif de résolution du problème. Pour le model checking probabiliste numérique, le principe consiste à calculer la probabilité d'occurrence de certains évènements durant l'exécution du système. La théorie du Model checking probabiliste est présentée dans [Kwiatkowska et al., 2007] comme étant basée sur la théorie et mesures de probabilités.

Le model checking probabiliste numérique permet de calculer la probabilité en explorant les différents chemins du modèle. Ceci permet d'avoir une valeur précise de cette dernière. Mais comme pour les autres approches exactes, le traitement des problèmes de grande taille reste un défi à cause de l'explosion combinatoire.

Dans la littérature, le model checking est utilisé dans [Ballarini et al., 2011] et [Rahimi et al., 2018].

1.5.3.2 Méthodes approchées

Les méthodes approchées permettent de résoudre le problème mais en obtenant une solution ou une valeur de solution approchée. Plusieurs méthodes existent dans la littérature, nous retenons les trois grandes catégories de méthodes (Figure. 1.12) : les méthodes mathématiques (Heuristiques et Méta-heuristiques), les approches par simulation simulation événementielle et Monte Carlo) et le model checking statistique.

Heuristiques et Méta-heuristiques

Une heuristique est une méthode basée sur des algorithmes approchés qui permettent de résoudre dans un temps polynomial un problème défini et formulé pour trouver, au moins, une solution réalisable mais pas forcément optimale. Une heuristique est une méthode dédiée au problème, c'est à dire, que l'algorithme est construit pour un problème donné. Ce type d'approche permet d'accélérer le processus de résolution mais sans offrir de garantie sur la qualité de la solution. Deux types de méthodes heuristiques sont connues en littérature :

— Méthodes constructives : partent d'une solution incomplète et y ajoutent des éléments afin d'obtenir à la fin une solution complète admissible.
— Méthodes basées sur les fouilles locales (recherche de solution) : les fouilles démarrent avec une solution existante puis essayent d'améliorer de manière itérative cette solution en explorant son voisinage.

Les heuristiques sont utilisées, dans la littérature, pour résoudre le problème d'ordonnancement proactif sous perturbations. Dans [Daniels and Carrillo, 1997], une heuristique nommée *β-heuristique* est conçue pour résoudre le problème d'ordonnancement d'un atelier à une machine avec des incertitudes sur la durée d'exécution des jobs. Cette heuristique s'inscrit dans une approche itérative permettant de trouver une solution proche de l'optimale. Dans cet article une comparaison entre l'algorithme Branch and Bound et l'heuristique est proposée concluant à l'efficacité des heuristiques pour traiter ce type de problèmes. Dans [Kouvelis and Yu, 1997], une heuristique est définie pour résoudre le problème d'un atelier flow shop à deux machines avec des incertitudes sur les durées d'exécution. Dans ce cas aussi, l'heuristique est basée sur une méthode itérative permettant de tendre vers une solution proche de l'optimale.

Les heuristiques dépendent souvent des problèmes à résoudre et sont souvent basées sur la recherche dans l'espace de solutions. Pour palier aux faiblesses des heuristiques, des algorithmes plus poussés ont vu le jour : les méta-heuristiques. L'objectif des méta-heuristiques est de trouver l'optimal global sans arriver à une explosion de l'espace de solution et donc rester bloqué par optimum local. Pour ce faire, les méta-heuristiques sont des algorithmes souvent inspirés de processus naturels.

En effet, les algorithmes méta-heuristiques permettent de trouver la solution d'un problème en se basant sur une méthode approchée. Les problèmes nécessitant un grand temps de calcul ou une grande capacité de stockage sont souvent résolus utilisant les méta-heuristiques. Ces méthodes peuvent être adaptées à différents problèmes permettant de construire des algorithmes de résolution plus complets et complexes que les heuristiques. De nombreuses méta-heuristiques sont définies dans la littérature. Un grand nombre d'entre elles est fortement inspiré de systèmes naturels tels que les systèmes biologiques ou naturels (algorithme évolutionnaire, algorithme génétique, algorithme neuronal, colonie de fourmis).

Pour traiter le problème d'ordonnancement robuste, nous retrouvons dans la littérature, plusieurs articles utilisant les algorithmes génétiques [Jensen, 2001] [V.J.Leon et al., 1994] [Sevaux and Sörensen, 2004]. Le principe de ces algorithmes est fortement inspiré de la sélection naturelle et la génétique. Le point de départ d'un algorithme génétique est une population initiale de solutions (des individus), choisies aléatoirement. Une évaluation de la performance de ces solutions est effectuée. L'étape clé de cet algorithme consiste à exécuter des opérateurs génétiques : la sélection, le croisement et la mutation sur les différentes solutions afin de permettre la reproduction des individus (sachant que seuls les meilleurs individus survivent à la reproduction) permettant de construire de nouvelles solutions. Ce cycle est répété jusqu'à trouver une solution satisfaisante, i.e. qui répond aux critères d'optimisation du problème.

Dans [Xiong et al., 2013], un autre type d'algorithme, nommé algorithme évolutionnaire est proposé pour résoudre le problème d'ordonnancement robuste pour un atelier job shop flexible soumis à des pannes machines.

Simulation à événements discrets

Dans le cas de la simulation à événements discrets, la résolution de ce type de problématique est basée sur la modélisation numérique du processus et la solution de la problématique est trouvé à l'aide de la simulation de l'occurrence des événements dans un cadre discret.

En effet, comme son nom l'indique, l'approche par simulation utilise des méthodes de traitement par simulation ce qui permet de diminuer le temps de traitement des modèles.
Les approches par simulations se sont imposées dans le domaine manufacturier pour définir le contrôle des systèmes de production ou la visualisation des flux [Simon et al., 2018]. Plusieurs travaux permettent de prendre en compte les perturbations dans le contexte de l'ordonnancement. Dans [Kádár et al., 2004], les auteurs proposent de prendre en considération les incertitudes durant l'exécution d'ordonnancement prédictif dans un environnement dynamique. La simulation est utilisée, dans ce cas, pour évaluer la performance de l'ordonnancement face aux incertitudes dues au temps de réparation, à la durée d'exécution ou encore à des problèmes de qualité. Plusieurs travaux proposent des approches par simulation pour traiter les perturbations dans l'ordonnancement [Allen et al., 2014] [Zandieh and Motallebi, 2018] [Vieira and Frazzon, 2020] [Gyulai et al., 2017].

Simulation de Monte Carlo

La simulation de Monte Carlo est une méthode de résolution basée sur un algorithme itératif visant à évaluer statistiquement des solutions en utilisant une procédure indépendante et aléatoire pour générer les expériences. Pour chaque simulation un calcul probabiliste est effectué afin de déterminer si cette simulation répond au critère de vérification ou non en fonction de l'intervalle de confiance fixé. Les résultats peuvent être traités par des méthodes statistiques usuelles.

[Liu and Sahinidis, 1996,Beck and Wilson, 2004] utilisent la simulation de Monte Carlo pour résoudre un problème d'optimisation d'un ordonnancement sous incertitudes.

Model checking statistique

Le model checking statistique est une alternative au model checking numérique qui rencontre des difficultés dans la résolution de problèmes de grande taille. L'explosion combinatoire est, en effet, un limite considérable pour les méthodes numériques de Model checking et ce dû aux calculs onéreux faits pour trouver la valeur exacte de la probabilité d'un chemin. Afin de palier à ces faiblesses, la méthode du Model Checking statistique à été introduite. L'objectif étant de se baser sur la simulation de Monte Carlo pour effectuer la vérification des propriétés probabilistes. Les propriétés peuvent être qualitatives ou quantitatives. Le principe du model checking probabiliste statistique change en fonction du type de la propriétés à vérifier.

Les propriétés quantitatives ont pour objectif d'estimer la probabilité que le système satisfasse une propriété propriété. le résultat de vérification de ce type de propriété est donc une valeur de probabilité estimée. Ceci est traité en utilisant la méthode de Monte Carlo présentée dans le paragraphe précédent. La précision et le taux d'erreur des résultats dépend tout d'abord du nombre de simulations effectuées et ensuite des paramètres de l'utilisateur. L'objectif des propriétés quantitatives est différent, elles permettent de comparer la probabilité d'une propriété avec une valeur donnée. La propriété est présentée sous forme d'inégalité (supérieure ou inférieure) et la réponse attendue dans ce cas est booléenne (Vrai ou Faux).

1.5.4 Synthèse de l'état de l'art

Dans l'état de l'art présenté dans ce chapitre, nous nous sommes intéressés à la question de l'ordonnancement soumis à des perturbations stohastiques dans deux communautés : la Recherche Opérationnelle et les Systèmes à Évènements Discrets. La RO est l'approche classique pour résoudre ce type de problématique. L'intérêt porté aux approches SED est motivé par leur capacité avérée à modéliser et résoudre des problématiques dans des contextes dynamiques. Elles permettent ainsi de s'approcher de le réalité des systèmes étudiés. Cette différence entre les deux types d'approches en termes de modélisation a d'ailleurs fait l'objet d'une comparaison dans un de nos articles [Himmiche et al., 2017].
L'état de l'art présenté dans ce chapitre est certes non-exhaustif, il permet cependant de répertorier les différents familles d'approches existante permettant de traiter le problème d'ordonnancement sous perturbations.

Les performances identifiées à travers cet état de l'art diffèrent mais sont toutes reliées à la performance de l'ordonnancement dans un contexte perturbé. Les performances étudiées fréquemment dans la littérature peuvent être reliées à une mesure de l'espérance, à une probabilité ou à un autre critère. La performance peut être également liée à l'optimisation d'un critère. L'espérance comme mesure de performance est certes très utilisée mais elle garantit peu de choses sur la valeur du critère. Il a en effet forcément de la dispersion autour de cette espérance, due notamment aux perturbations (ainsi il faudrait aussi connaître l'écart-type). L'optimisation d'un critère permet, quant à elle, de générer un ordonnancement intégrant les perturbations mais ne présente pas d'évaluation à proprement dit de l'impact des perturbations sur l'ordonnancement. Pour obtenir cette mesure de performance, les approches mesurant la probabilité d'un critère semble les plus adaptée. Cette constatation est d'autant plus forte dans le cas de la description stochastique des perturbations.

Pour modéliser le problème d'ordonnancement sous perturbations, les approches stochastiques sont plus à même de prendre en compte les caractéristiques stochastiques des perturbations. Pour résoudre le modèle du problème, les méthodes approchées sont souvent privilégiées. Bien que les résultats obtenus avec ce type de méthode soient approximatifs, leur capacité à traiter des problèmes complexes et de grande taille en fait de bons candidats.

En conclusion de cet état de l'art, les approches proposées dans la littérature sont souvent dédiées à un type de problème donné. En effet, chaque article de l'état de l'art considère un type d'atelier précis (Job Shop, Flow shop, etc.) en considérant des perturbations bien définies (panne machine, incertitudes sur les durées d'exécution). Dans le contexte de l'industrie 4.0, la problématique de l'ordonnancement est soumise à plus de contraintes de flexibilité. De ce fait, la configuration des ateliers de production est soumise à des changements récurrents. Pour proposer une approche d'aide à la décision dans ce contexte, il est important d'adopter une approche générique vis-à-vis de l'atelier de production et des perturbations.

Pour répondre à ce verrou scientifique, en termes de modélisation les systèmes à événements discrets stochastiques permettent une modélisation dynamique du problème prenant ainsi en compte le comportement réel de l'atelier de production et d'introduire la description des perturbations. Pour résoudre ce type de modèle, la méthode de résolution approchée basée sur le Model-Checking statistique est particulièrement adaptée pour prendre en compte des problèmes de taille importante.

1.6 Hypothèses et questions de recherche

L'objectif de cette thèse est de pallier au problème de généricité identifié à partir de l'état de l'art.

- **QR :1** : Comment développer des modèles adaptables et agiles pour considérer le problème d'ordonnancement soumis à des perturbations ?
- **QR :2** : Comment les modèles obtenus peuvent-ils être utilisés pour évaluer la performance d'ordonnancement face aux perturbations identifiées ?

La première question permet de poser la problématique de modélisation générique de l'ordonnancement sous perturbations alors que la deuxième question s'intéresse à l'évaluation de l'ordonnancement et au choix de résolution adopté dans ce cas. Nous rappelons que les caractéristiques suivantes sont considérées pour la suite des travaux de cette thèse.

1. Les perturbations considérés sont décrites à l'aide de modèles stochastiques.
2. Les perturbations sont prises en compte de manière proactive.

Dans le contexte de l'industrie 4.0, nous considérons que les informations sur les perturbations sont obtenues depuis le traitement des données de l'atelier. En effet, ces informations permettent de construire des modèles stochastiques des perturbations de l'atelier. Les modèles de perturbations sont obtenues à l'aide du traitement effectué au préalable par les spécialistes de l'analyse des données.

Dans le contexte de l'aide à la décision, l'ordonnancement choisi pour être exécuté dans l'atelier de production est identifié d'une manière proactive permettant de prendre en compte les perturbations à priori dans l'atelier de production.

Pour pouvoir répondre aux verrous identifiés, l'hypothèse de recherche identifiée dans le cadre de la thèse est la suivante :

HR : *Les modèles et outils des Systèmes à Événements Discrets permettent de modéliser et de résoudre le problème d'ordonnancement sous perturbations.*

Les modèles et outils SED sont une alternative intéressante aux modèles classiques de RO de par leur capacité intrinsèque à modéliser la dynamique d'un système. De plus les applications nombreuses en sûreté de fonctionnement montrent leur capacité à vérifier et évaluer des propriétés face à des perturbations.

1.7 Conclusion

Ce chapitre présente le contexte et l'état de l'art liés aux travaux présentés dans ce mémoire. Il permet dans un premier temps d'établir les définitions et notions nécessaires pour traiter la problématique de l'ordonnancement de la production. Obtenir une solution d'ordonnancement déterministe n'est plus un objectif suffisant dans l'atelier de production. L'optimisation des performances de l'ordonnancement en considérant que les données sont certaines et que l'environnement de production est stable est une hypothèse forte par rapport à la réalité. Cette hypothèse est d'autant plus contraignante dans le contexte actuel de l'Industrie 4.0. Pour lever cette hypothèse, il est donc important de prendre en compte les perturbations dans la problématique d'ordonnancement pour éviter que leur présence ne détériore la performance de ce dernier.

Le positionnement de nos travaux dans le contexte de l'industrie 4.0 permet de considérer que les données de l'atelier sont disponibles en continu et que le traitement de ces données permet de décrire le comportement des perturbations. Une description stochastique est alors plus proche du comportement réel des perturbations.

L'état de l'art présenté dans ce chapitre permet d'émettre le constat que les approches existantes sont dédiés à des ateliers, des perturbations et des performances spécifiques. Dans le contexte de l'industrie 4.0, où les ateliers de production sont flexibles et agiles, il est donc important d'avoir une approche générique et adaptable à tout type d'atelier. À la fin de ce chapitre, nous considérons que les approches SED stochastiques représente un potentiel de modélisation et de résolution qui correspond à cet objectif.

Chapitre 2

Cadre d'étude pour l'ordonnancement robuste

Sommaire

2.1 Introduction 29
2.2 Performance de robustesse : spécification 30
2.2.1 À propos de robustesse 30
2.3 Formalisation des problèmes de robustesse 33
2.3.1 Problèmes d'évaluation 33
2.3.2 Problèmes d'optimisation 39
2.4 Conclusion 40

2.1 Introduction

Nous l'avons vu dans le chapitre 1, l'ordonnancement sous perturbations peut reposer sur deux stratégies complémentaires. La première consiste à mettre en œuvre une approche d'ordonnancement réactif en ligne qui utilise la connaissance progressive des événements en ligne (y compris l'occurrence de perturbations) pour, pendant l'exécution de la production, modifier une solution prédictive existante, ou construire une nouvelle solution. La deuxième stratégie consiste à mettre en œuvre une approche d'ordonnancement proactif qui utilise la connaissance a priori des perturbations (soit liée à un historique, soit liée à des spécifications) pour proposer, avant exécution, des solutions performantes malgré les perturbations prises en compte. C'est cette deuxième stratégie qui nous intéresse. Encore faut-il définir ce que veut dire être performant malgré les perturbations. Dans [Billaut et al., 2010], deux types de performance sont considérés : la flexibilité et la robustesse.

— **Flexibilité** :
La performance de flexibilité est définie comme étant le degré de liberté dont dispose l'ordonnancement durant son exécution pour répondre aux conditions réelles. Quatre types de flexibilité sont distingués par [Billaut et al., 2010] : la flexibilité temporelle, la flexibilité séquentielle, la flexibilité sur les affectations, et la flexibilité sur les modes d'exécution. La flexibilité temporelle consiste à autoriser certaines opérations à dériver dans le temps, si les conditions l'imposent. La flexibilité séquentielle consiste à autoriser la modification de l'ordre dans lequel les opérations doivent s'exécuter. La flexibilité sur les affectations consiste à autoriser la réalisation d'une

tâche par une ressource autre que celle initialement affectée à l'exécution de ladite tâche. Enfin, la flexibilité sur les modes d'exécution consiste à autoriser le changement du mode d'exécution d'une tâche. Le mode d'exécution comprend entre autre l'autorisation ou non de la préemption, du chevauchement, le changement de gamme, etc.
La performance de flexibilité permet donc d'offrir *a priori* des degrés de liberté à la solution d'ordonnancement afin de réagir plus facilement *a posteriori* à une perturbation. Ainsi cette performance ne vaut que si l'ordonnancement est couplé avec du réactif.

— **Robustesse** :

La performance de robustesse est intégrée dans la gestion de production depuis le siècle dernier. Sa définition n'a cessé d'évoluer depuis. En effet, dans [V.J.Leon et al., 1994], un ordonnancement robuste est défini comme étant insensible aux perturbations imprévisibles de l'atelier tout en considérant la politique de contrôle déployée. Dans [Sevaux and Sörensen, 2004], les auteurs affirment qu'une façon de prendre en compte des données stochastiques d'un problème d'ordonnancement est de trouver les solutions qui sont robustes. Dans [Billaut et al., 2010], la robustesse est présentée comme un qualificatif qui réfère généralement à la capacité de tolérance aux approximations sur les hypothèses, modèles ou données. [Davenport et al., 2014] définissent un ordonnancement robuste comme étant capable d'absorber un certain niveau d'événements imprévus (aléas) sans avoir recourt au ré-ordonnancement. Plus récemment, une autre définition est proposée par [Ivanov et al., 2016] : "Un ordonnancement qui est capable d'atteindre la performance planifiée en dépit des perturbations est dit robuste."
Ces différentes formulations convergent vers une même idée de la robustesse : la performance, la qualité ou la capabilité qu'un ordonnancement doit maintenir ou garantir malgré les différentes perturbations et variations émanant du système de production ou de son environnement.

La performance de robustesse vise donc à un principe de réalité dans les ateliers de production : la nécessaire prise en compte des perturbations a priori dès lors qu'on s'intéresse au pilotage de la production. Ainsi, l'évaluation de la performance de robustesse doit permettre au final de générer un ordonnancement résistant et de prédire le comportement de l'ordonnancement face à des perturbations attendues. Dans la suite de nos travaux, nous nous focaliserons sur ce concept de robustesse comme performance d'évaluation pour identifier un corpus de problèmes de robustesse pertinents et y apporter des éléments de réponse afin de contribuer au challenge de robustesse identifié notamment par [Monostori et al., 2016] pour la mise en œuvre de l'industrie 4.0. Ce chapitre vise donc à répondre aux questions suivantes :

— Comment spécifier la performance de robustesse ?
— Comment structurer les problèmes de robustesse à partir de cette spécification ?

2.2 Performance de robustesse : spécification

2.2.1 À propos de robustesse

La robustesse, est une notion de qualité qui permet de caractériser et qualifier une solution. Dans [Billaut et al., 2010], une solution est qualifiée de robuste si elle est *apte à résister à des à peu près ou à des zones d'ignorance.* La robustesse dans le contexte de l'ordonnancement doit répondre à une préoccupation du décideur de préserver la performance de l'ordonnancement malgré des perturbations . Ainsi cette performance de robustesse doit être parfaitement définie dès la formulation même du problème d'ordonnancement.

La performance de robustesse est liée à une solution d'ordonnancement précise. Toujours dans [Billaut et al., 2010] une solution d'ordonnancement est qualifiée de robuste si *sa performance est peu sensible aux incertitudes et aux aléas.* Cette définition si elle peut faire consensus impose tout de même de définir ce que veut dire "*être peu sensible*". Autrement dit, elle impose de pouvoir quantifier la robustesse.

2.2.1.1 Robustesse et Niveau de service

Pour intégrer la notion de robustesse dans le problème d'ordonnancement sous perturbations, il est important de spécifier la mesure de cette performance. Dans la littérature, cette mesure dépend du contexte du problème et de l'objectif de performances à atteindre. À titre d'exemples, dans [Goren and Sabuncuoglu, 2008], l'espérance est utilisée pour la mesure de la robustesse. Dans [Ghezail et al., 2010], c'est le regret maximum qui est adopté comme mesure. Dans [Beck and Wilson, 2004], la robustesse est quantifiée par la probabilité qu'une performance de l'ordonnancement atteigne un niveau donné. Dès lors que les perturbations sont modélisées comme des variables aléatoires, il semble assez naturel d'associer la robustesse à une mesure de probabilité permettant d'évaluer la chance que la performance de l'ordonnancement ne soit pas trop dégradée. C'est au décideur de définir la performance de l'ordonnancement par exemple, le makespan, un coût ou toute autre critère classique en ordonnancement (retard maximum, en-cours, ...). Dans le contexte de l'aide à la décision et de la sûreté de fonctionnement, la mesure de probabilité permet de quantifier l'état du système et de voir son évolution dans le temps. Pour l'ordonnancement de la production, la mesure de la probabilité permet de quantifier la capacité de l'ordonnancement à maintenir des performances dans un contexte perturbé.

Dans ce cadre, [Dauzère-Pérès et al., 2010] introduit la notion de niveau de service pour la mesure de la robustesse de l'ordonnancement. Ce niveau de service mesure la probabilité que la performance d'un ordonnancement soit inférieure (ou supérieure) à une valeur donnée.

Dans le cadre de cette thèse, nous reprenons à notre compte la notion du niveau de service introduite par [Dauzère-Pérès et al., 2010] : *"Le niveau de service dans le cadre de l'ordonnancement correspond à la probabilité qu'un critère soit inférieur à (ou supérieur) ou égal à une valeur donnée. Si l'on considère le makepan, c'est la probabilité que le makepan soit inférieur ou égal à un temps maximum donné pour exécuter toutes les opérations. Il est donc possible de prendre en compte le niveau de service de n'importe quel critère dans l'ordonnancement."* Cette définition permet donc de définir une spécification de la robustesse en choisissant le makespan comme critère de performance pour un ordonnancement.

Dans ce cadre, nous considérons les éléments suivants :
— Un ordonnancement à évaluer s
— Le critère de performance pour l'ordonnancement, soit sa durée totale notée C_{max}
— Un ensemble de perturbations $Pert$,
— Une deadline à satisfaire $\tilde{d}$ par s

Définition 2. *Le niveau de service est défini dans le cadre de cette thèse par :* $RL\left(s, Pert, \tilde{d}\right)$.
Le niveau de service d'un ordonnancement s est mesuré par la probabilité Pr que la durée C_{max} de s soit inférieure ou égale à une deadline $\tilde{d}$, malgré un ensemble de perturbations $Pert$.
Ce qui se traduit formellement par (2.1).

$$RL\left(s, Pert, \tilde{d}\right) = Pr\left(C_{max}(s, Pert) \leq \tilde{d}\right) \tag{2.1}$$

La définition 2.1 du niveau de service nous permet de donner une définition à la performance de la robustesse d'un ordonnancement.

Définition 3. *Dans le cadre de cette thèse, un ordonnancement s soumis à des perturbations $Pert$ et satisfaisant une deadline $\tilde{d}$ malgré ces perturbations est qualifié de robuste si son niveau de service $RL\left(s, Pert, \tilde{d}\right)$ est supérieur ou égal à un seuil de niveau de service RL_{lim}.*
Ce qui se traduit formellement par (2.2) *:*

$$RL\left(s, Pert, \tilde{d}\right) \geq RL_{lim} \tag{2.2}$$

RL_{lim} permet à un décideur de définir le risque accepté de ne pas satisfaire la deadline. $RL_{lim} = 95\%$ signifierait que le décideur accepterait que la deadline ne soit pas satisfaite dans 5% des cas uniquement. Cela permet notamment d'impliquer le décideur et d'imposer la performance de robustesse comme un indicateur d'aide à la décision.

Cette spécification de la robustesse inclut pleinement les paramètres entrant en jeu dans tout problème d'ordonnancement sous perturbations (un risque à couvrir $Pert$, une performance malgré les perturbations $\tilde{d}$, un niveau de performance à atteindre RL_{lim}).

Cette définition permet de mettre en évidence deux facettes de la robustesse qui sont complémentaires mais concurrentes : stabilité et sensibilité [Aubry, 2018]. On peut les voir comme deux facettes d'une même pièce (la robustesse).
Dans notre définition, la stabilité fait plutôt référence à l'ensemble $Pert$ qui permet de "mesurer" le risque que peut couvrir un ordonnancement donné sans trop remettre en cause sa performance (la dégradation de performance étant mesurée par $\tilde{d}$). Autrement dit, dans notre définition de la robustesse, $Pert$ permet de "quantifier" un ensemble de perturbations sur lequel l'ordonnancement est stable. Cette notion est proche de la notion de stabilité en automatique : si les entrées sont bornées par $Pert$, alors je peux garantir que l'ordonnancement ne diverge pas (ne va pas plus loin que $\tilde{d}$).
Et à l'opposé, la sensibilité fait plutôt référence à la deadline $\tilde{d}$, qui permet de mesurer à quel point mon ordonnancement est sensible aux perturbations en entrée définies par $Pert$.

Nous utilisons ainsi les définitions proposées dans [Aubry, 2018] :
- La stabilité mesure les variations autorisées en entrées du problème (mesurées par $Pert$) sans qu'elles n'engendrent de variations trop importantes sur la sortie du problèmes (contraintes par $\tilde{d}$).
- la sensibilité mesure les variations engendrées sur la sortie du problème (mesurées par $\tilde{d}$ lorsque les entrées du problème varient (contraintes par $Pert$).

Intuitivement, nous comprenons que ces deux notions sont concurrentes. Si je cherche à augmenter la stabilité ($Pert$ contient plus de perturbations) alors je risque d'augmenter également la sensibilité ($\tilde{d}$ croît également). Si je cherche à diminuer la sensibilité (diminuer $\tilde{d}$) alors je risque de dégrader la stabilité ($Pert$ devra contenir moins de perturbations).

Ainsi cette définition est fondamentale car elle permet de mettre en évidence plusieurs problèmes de robustesse suivant que le décideur va plutôt s'intéresser au niveau de service, à la

stabilité ou à la sensibilité, où alors chercher un compromis entre ces différentes performances. L'objectif de la prochaine section est de mettre en évidence et formaliser cette multitude de problèmes en définissant un cadre d'étude pour l'ordonnancement robuste. À notre connaissance, peu de travaux s'intéressent à l'étude théorique des problèmes de robustesse en proposant un cadre formalisant ceux-ci. En revanche, la quantification de la robustesse proposée dans cette thèse et la formalisation proposée ci-dessous peut être positionnée par rapport aux approches considérant les perturbations dans une approche par scénario. De ce fait, un parallèle entre la notion de robustesse détaillée dans la section suivante et la L_λ-robustesse présentée dans [Aubry, 2018] est proposée dans l'annexe C.

2.3 Formalisation des problèmes de robustesse

En considérant l'expression mathématique de la robustesse (2.2), il est possible de mettre en évidence plusieurs problèmes de robustesse liés notamment à la stabilité et/ou à la sensibilité. Ces problèmes apparaissent en fonction de la connaissance des différents paramètres d'entrée du problème : s, $Pert$, $\tilde{d}$ et RL_{lim}. Ceux-ci peuvent être connus ou non par le décideur.

Une des premières contributions de cette thèse consiste donc en l'identification de différents problèmes de robustesse, leur catégorisation et leur formalisation, offrant ainsi un cadre d'étude pour les problèmes d'ordonnancement sous perturbations.

Les problèmes de robustesse identifiés peuvent être classés en deux catégories
- Les problèmes d'évaluation, qui évaluent la robustesse d'un ordonnancement connu s en fonction des différents paramètres du problème.
- Les problèmes d'optimisation, qui ont pour objectif de générer l'ordonnancement s qui minimise ou maximise un ou plusieurs paramètre(s) du problème défini.

Les problèmes d'évaluation sont basés sur l'analyse des différentes performances, niveau de service, sensibilité et stabilité en considérant les définitions 2.2 et 2.1 pour un ordonnancement déterministe s donné. Les problèmes d'optimisation sont également définis en se basant sur ces trois performances mais avec l'objectif de trouver une solution optimale $s*$ répondant à ces trois performances.

2.3.1 Problèmes d'évaluation

Les problèmes d'évaluation permettent l'expression des différentes caractéristiques qui peuvent être satisfaites par l'ordonnancement. À partir des différents paramètres, la spécification de la robustesse (2.2) est adaptée pour exprimer les différents problèmes (Table 2.1).

En faisant varier tous les paramètres, huit problèmes de robustesse de type évaluation peuvent être identifiés puis formalisés. Pour chaque problème, les paramètres connus (en entrée) et les paramètres recherchés (en sortie) permettent de définir le type de problème et d'adapter ainsi sa spécification. Dans la table 2.1, les problèmes sont classés selon le nombre de paramètres inconnus en sortie du problème et donc potentiellement, selon leur difficulté.

2.3.1.1 EV_2 : Évaluation du niveau de service

Nous commençons volontairement par détailler le problème EV_2 car nous verrons dans la suite que les autres problèmes peuvent être traités à partir de l'évaluation du niveau de service.

L'évaluation du niveau de service (EV_2) consiste à évaluer le niveau de service $RL\left(s, Pert, \tilde{d}\right)$ d'un ordonnancement s connu soumis à des perturbations $Pert$ et devant satisfaire une deadline $\tilde{d}$.

Pour résoudre ce problème, nous proposons dans l'algorithme (1) une réponse à la question suivante *Quelle est la valeur du niveau de service RL pour un ordonnancement s et un ensemble de perturbations Pert ?*. Cet algorithme retourne $Pr\left(C_{max}\left(s, Pert\right) \leq \tilde{d}\right)$. Nous faisons ici l'hypothèse que cette valeur peut être calculée et une méthode pour obtenir cette valeur sera présentée dans le chapitre 3. Dans la suite de ce chapitre, nous partirons du principe que la fonction $RL(s, Pert, \tilde{d})$ existe et permet donc d'évaluer le niveau de service d'un ordonnancement s soumis aux perturbations $Pert$ pour une deadline définie $\tilde{d}$.

Algorithm 1 Analyse du niveau de service : $RL(s, Pert, \tilde{d})$

Require: $s, Pert, \tilde{d}$
  $RL := Pr\left(C_{max}\left(s, Pert\right) \leq \tilde{d}\right)$;
  return (RL) ;

2.3.1.2 EV_1 : Qualification de la robustesse

Le problème (EV_1) permet d'obtenir une information sur la performance de l'ordonnancement en fonction d'un seuil de robustesse souhaité par le décideur. Tous les paramètres du problème sont connus s, $Pert$, $\tilde{d}$ et RL_{lim}. Il s'agit ici de comparer le niveau de service de l'ordonnancement à un seuil défini RL_{lim}. Ce problème permet de répondre à la question suivante : *L'ordonnancement s permet-il de satisfaire le seuil de robustesse RL_{lim} défini par le décideur ?*

Étant donnée la définition donnée dans (2.2), ce problème revient exactement à poser la question de savoir si l'ordonnancement s peut être qualifié de robuste ou non.

L'algorithme de résolution (2), permet de répondre à cette question en faisant appel à l'algorithme (1) par le biais de la fonction $RL\left(s, Pert, \tilde{d}\right)$. Cet algorithme renvoie un résultat booléen en fonction du résultat.

Algorithm 2 Qualification de la robustesse

Require: $s, Pert, , RL_{lim}$
  $RL := RL(s, Pert, \tilde{d})$;
  if $(RL \geq RL_{lim})$ **then**
    return $(True)$;
  else
    return $(False)$;
  end if

2.3.1.3 EV_3 : Analyse de sensibilité

Le troisième problème identifié est l'analyse de la sensibilité. Pour ce problème, les paramètres connus du problème sont l'ordonnancement s, l'ensemble de perturbations $Pert$ et le seuil de robustesse RL_{lim}. L'élément recherché dans ce cas est la plus petite valeur de la deadline $\tilde{d}$ (notée $\underline{\tilde{d}}$) que peut respecter l'ordonnancement s par rapport au seuil de robustesse défini RL_{lim}. Nous

parlons ici d'analyse de sensibilité car il s'agit bien d'analyser la sensibilité de l'ordonnancement quant à ces performances temporelles malgré les perturbations définies par $Pert$. La question à laquelle répond ce problème est *Étant donné un ordonnancement s soumis aux perturbations Pert, quelle est la deadline minimale* $\underline{\tilde{d}}$ *qui permet de satisfaire le seuil de robustesse défini* RL_{lim} *?*

Ce problème peut être résolu par un algorithme de recherche par dichotomie (Algorithm 3) faisant lui aussi appel à l'évaluation de $RL(s, Pert, \tilde{d})$ (Algorithm 1). La précision de la recherche est spécifiée par le paramètre ξ représentant l'écart entre $\tilde{d}^+$ et $\tilde{d}^-$.

Algorithm 3 Analyse de sensibilité : $\tilde{d}\,(s, Pert, RL_{lim})$

Require: $s, Pert, RL_{lim}$, $\tilde{d}^+$, $\tilde{d}^-$, ξ ;
  while $\tilde{d}^+ - \tilde{d}^- > \xi$ **do**
    $\underline{\tilde{d}} := \frac{\tilde{d}^+ + \tilde{d}^-}{2}$;
    $RL :=$;
    if $(RL(s, Pert, \underline{\tilde{d}}) \geq RL_{lim})$ **then**
      $\tilde{d}^+ := \underline{\tilde{d}}$;
    else
      $\tilde{d}^- := \underline{\tilde{d}}$;
    end if
  end while
  return $(\underline{\tilde{d}})$;

L'algorithme de recherche par dichotomie présente le grand avantage d'être robuste puisqu'il permet de s'approcher aussi près que l'on veut de la valeur de $\underline{\tilde{d}}$ en fixant la valeur de ξ en conséquence. La contrepartie réside dans le fait que plus ξ sera petit et plus l'algorithme prendra de temps à renvoyer la valeur de $\underline{\tilde{d}}$. De plus, cet algorithme faisant appel à l'évaluation de $RL(s, Pert, \tilde{d})$, son efficience est intimement liée à l'efficience de l'algorithme permettant d'évaluer $RL(s, Pert, \tilde{d})$ (problème EV_1).

De plus, une question essentielle afin de s'assurer de l'efficacité de cet algorithme repose sur la détermination des valeurs en entrée pour $\tilde{d}^+$ et $\tilde{d}^-$ qui doivent être respectivement une borne supérieure et inférieure de $\underline{\tilde{d}}$. Des valeurs naïves consistent à fixer $\tilde{d}^+ = C_{max}(s, I_{max})$ et $\tilde{d}^- = 0$. Avec I_{max} qui correspond au scénario représentant le pire cas pour les perturbations $Pert$, en supposant que celui-ci est identifiable. Si ce n'est pas le cas, fixer une très grande valeur à $\tilde{d}^+$ est souvent efficace (par exemple un très grand multiple de C^{ref}_{max}).

2.3.1.4 EV_4 : Analyse de stabilité

L'analyse de stabilité permet d'évaluer le plus grand ensemble de perturbations $Pert$ sur lequel l'ordonnancement garde des performances acceptables. Dans le cas de l'analyse stabilité, les éléments connus sont l'ordonnancement s, la deadline $\tilde{d}$, et un niveau de service à atteindre RL_{lim}. La question que l'analyse de stabilité traite est : *Étant donnés un ordonnancement s et une deadline* $\tilde{d}$*, quel est le plus grand ensemble de perturbations Pert permettant de satisfaire le seuil de robustesse* RL_{lim} *?* En pratique, résoudre ce problème nécessite de définir une métrique afin de pouvoir associer une grandeur à l'ensemble $Pert$ qui permet de le quantifier. Ainsi l'analyse de la stabilité dépend de la mesure associée aux perturbations considérées. Dans le cadre de cette

thèse, les types de perturbations définis sont les incertitudes et les aléas qui ne se mesurent pas de la même manière.

Un aléas étant défini comme l'occurrence d'un événement non prévu, il semblerait que sa probabilité d'occurrence puisse être considérée comme une métrique. Dans ce cas-là, l'analyse de stabilité consisterait à chercher quelle est la probabilité d'occurrence maximum que peut supporter l'ordonnancement s sans dépasser la deadline $\tilde{d}$.

Les incertitudes sont définies comme étant la différence entre les informations prévues (ou attendues) et celles disponibles dans l'atelier (ou réelles). Imaginons que les informations prévues peuvent être modélisées comme un vecteur de référence que l'on peut noter I^{ref} fixant des valeurs de référence aux données d'entrée du problème (les durées opératoires d_{jKr} par exemple). Dans ce cas, une incertitude peut être modélisée comme un vecteur de variation que l'on peut noter ΔI. À titre d'exemple, pour mieux comprendre, si on s'intéresse à des incertitudes sur la durée d'exécution des opérations, cela revient à dire que l'incertitude est vue comme une variation positive ou négative autour d'une durée de référence. Pour chaque opération o_{jk}, on aurait la durée réelle d_{jkr} qui vérifierait : $d_{jkr} = d^{ref}_{jkr} + \delta d_{jkr}$. Dans ce cas-là, $\Delta I = [d_{jkr}]_{jkr}$ représente l'ensemble $Pert$. À partir de là, il faut définir une métrique $|\ |$ pour ΔI. Cela pourrait par exemple être les normes bien connues en mathématique : $||\ ||_1$, $||\ ||_2$ et $||\ ||_\infty$.

Afin de définir un algorithme permettant de résoudre ce problème, nous faisons l'hypothèse que nous disposons d'une métrique permettant de mesurer $Pert$. Nous la noterons $|Pert|$.

L'algorithme (Algorithm 4) permet de résoudre le problème. Il fonctionne lui aussi sur une recherche par dichotomie avec un précision fixée par le paramètre ξ et utilise également l'analyse du niveau de service $RL(s, Pert, \tilde{d})$. Et son efficacité dépend également de la capacité de l'utilisateur à fixer des bornes $|Pert|^-$ et $|Pert|^+$ pertinentes pour le problème. $|Pert|^- = 0$ est une borne triviale. C'est plus compliqué pour $|Pert|^+$.

Algorithm 4 Analyse de stabilité : $Pert\left(s, \tilde{d}, RL_{lim}\right)$

Require: s, $\tilde{d}$, RL_{lim}, $|Pert|^-$, $|Pert|^+$, ξ
  while $|Pert|^+ - |Pert|^- > \xi$ **do**
    $\overline{Pert} = \frac{|Pert|^+ + |Pert|^-}{2}$;
    $RL = RL(s, Pert, \tilde{d})$;
    if $(RL \geq RL_{lim})$ **then**
      $|Pert|^- = \overline{Pert}$;
    else
      $|Pert|^+ = \overline{Pert}$;
    end if
  end while
  return $(\overline{Pert})$;

2.3.1.5 EV_5 : Analyse du compromis (sensibilité/niveau de service)

Dans le cas de l'analyse du compromis entre la sensibilité et le niveau de service, deux objectifs sont à considérer. En effet, ce problème cherche à trouver la deadline minimale permettant de

satisfaire un niveau de service maximal. Les paramètres recherchés dans ce cas, sont donc la deadline $\underline{\tilde{d}}$ et le niveau de service $\overline{RL_{lim}}$. Le compromis entre la sensibilité et le niveau de service permet de répondre à la question suivante : *Étant donné un ordonnancement s soumis à l'ensemble de perturbations $Pert$, quelle est la deadline minimale $\underline{\tilde{d}}$ permettant de satisfaire un niveau de service maximal $\overline{RL_{lim}}$?*

L'analyse du compromis entre la sensibilité et le niveau de service revient finalement à trouver un front de Pareto entre la deadline $\tilde{d}$ et le niveau de service limite RL_{lim}. L'algorithme (Algorithm 5) permet de faire une approximation ce front de Pareto en utilisant des pas de variation rl et d. Cet algorithme fait appel aux algorithmes (1 et 3). Le front de Pareto est construit en deux étapes. Dans la première étape, l'algorithme parcourt les valeurs de RL_{lim} (en partant de 100% puis en retranchant à chaque itération la valeur rl) et en déduit les deadlines associées $\underline{\tilde{d}}$ à l'aide de l'algorithme 3 puis exécute l'algorithme 1 afin de trouver la valeur de $\overline{RL_{lim}}$ correspondant à cette deadline. Cela permet de construire à chaque itération un point du front de Pareto. Dans la deuxième étape l'algorithme parcourt le front de Pareto dans l'autre sens. Il parcourt les valeurs de $\tilde{d}$ (en partant de 0 et en ajoutant à chaque itération la valeur d) et en déduit les niveaux de services associés $\overline{RL_{lim}}$ à l'aide de l'algorithme 1 puis exécute l'algorithme 3 afin de trouver la valeur de $\underline{\tilde{d}}$ correspondant à ce niveau de service. Cela permet de construire à chaque itération un point du front de Pareto. Cet algorithme ne garantit pas que tous les points du front soient trouvés.

Algorithm 5 Analyse du compromis : Pareto $\tilde{d}/RL$

Require: s, $Pert$, $\tilde{d}_{max}$, rl, d
  $ParetoFront = \emptyset$
  $RL = 100\%$;
  while $(RL > 0)$ **do**
    Execute Algorithm 3 : $\underline{\tilde{d}} = \tilde{d}\,(s, Pert, RL)$;
    Execute Algorithm 1 : $\overline{RL_{lim}} = RL\left(s, Pert, \underline{\tilde{d}}\right)$;
    $ParetoFront = ParetoFront \bigcup (\underline{\tilde{d}}, \overline{RL_{lim}})$
    $RL = RL - rl$;
  end while
  $\tilde{d} = 0$;
  while $(\tilde{d} < \tilde{d}_{max})$ **do**
    Execute Algorithm 1 $\overline{RL_{lim}} = RL\left(s, Pert, \tilde{d}\right)$;
    Execute Algorithm 3 : $\underline{\tilde{d}} = \tilde{d}\left(s, Pert, \overline{RL_{lim}}\right)$;
    $ParetoFront = ParetoFront \bigcup (\underline{\tilde{d}}, \overline{RL_{lim}})$
    $\tilde{d} = \tilde{d} + d$;
  end while
  return $ParetoFront$

2.3.1.6 EV_6 : Analyse du compromis (stabilité/niveau de service)

Le problème d'analyse du compromis entre stabilité et le niveau de service cherche à trouver l'ensemble maximal de perturbations $\overline{Pert}$ qu'un ordonnancement s peut supporter tout en maximisant son niveau de service RL_{lim}. Ce problème permet de répondre à la question suivante : *Étant donnés un ordonnancement s et une deadline $\tilde{d}$ connus, quel est le plus grand ensemble*

de perturbations Pert que l'ordonnancement peut supporter tout en maximisant son niveau de service RL_{lim} ?

En pratique, l'analyse du compromis entre la stabilité et le niveau de service revient finalement à trouver un front de Pareto entre l'ensemble de perturbations *Pert* et le niveau de service limite RL_{lim}. L'algorithme (6) permet de faire une approximation de ce front de Pareto en utilisant des pas de variation *rl* et *pe*. Cet algorithme fait appel aux algorithmes (1 et 4). Le principe de l'algorithme est identique que pour le problème précédent (EV_5). En effet, dans ce cas l'algorithme de résolution cherche à construire un front de Pareto avec les valeurs de la métrique associée à *Pert* (comme pour le problème d'analyse de stabilité) et le niveau de service.

Algorithm 6 Analyse du compromis : Pareto $Pert/RL$

Require: $s, \tilde{d}, rl, pe$
  $ParetoFront = \emptyset$
  $RL = 100\%$;
  while $(RL > 0)$ **do**
    Execute Algorithm 4 : $\overline{Pert} = pert\left(s, \tilde{d}, RL\right)$;
    Execute Algorithm 1 : $\overline{RL_{lim}} = RL\left(s, \overline{Pert}, \tilde{d}\right)$;
    $ParetoFront = ParetoFront \bigcup (|\overline{Pert}|, \overline{RL_{lim}})$
    $RL = RL - rl$;
  end while
  $|Pert| = |\overline{Pert}|$;
  while $(|Pert| > 0)$ **do**
    Execute Algorithm 1 $\overline{RL_{lim}} = RL\left(s, Pert, \tilde{d}\right)$;
    Execute Algorithm 4 : $\overline{Pert} = Pert\left(s, \tilde{d}, \overline{RL_{lim}}\right)$;
    $ParetoFront = ParetoFront \bigcup (\overline{Pert}, \overline{RL_{lim}})$
    $|Pert| = |Pert| - pe$;
  end while
  return $ParetoFront$

2.3.1.7 EV_7 : Analyse du compromis (sensibilité /stabilité)

Dans le cas du compromis entre la sensibilité et la stabilité (EV_7), on cherche à trouver la plus petite deadline $\tilde{d}$ qui permet de satisfaire le plus grand ensemble de perturbations $\overline{Pert}$ tout en atteignant un niveau de service limite RL_{lim}. Ce problème permet de répondre à la question suivante : *Étant donné un ordonnancement s et un seuil de robustesse RL_{lim} connus, quel est le plus grand ensemble de perturbations $\overline{Pert}$ que l'ordonnancement s peut supporter avec une deadline minimale $\underline{\tilde{d}}$ tout en satisfaisant RL_{lim} ?*

Comme pour les deux problèmes précédents, l'objectif est finalement de construire le front de Pareto entre $|\overline{Pert}|$ et la deadline $\tilde{d}$. Pour ce faire, nous appliquons dans l'algorithme (7) le même principe que l'algorithme 6 en faisons appel cette fois aux deux algorithme permettant d'analyser la sensibilité (Algorithme 3) et la stabilité (Algorithme 4).

Algorithm 7 Analyse du compromis : Pareto $Pert/\tilde{d}$

Require: s, $\tilde{d}_{max}$, d, pe
  $ParetoFront = \emptyset$
  $\tilde{d} = 0$;
  while $(\tilde{d} < \tilde{d}_{max})$ **do**
    Execute Algorithm 4 : $\overline{Pert} = pert\left(s, \tilde{d}, RL_{lim}\right)$;
    Execute Algorithm 3 : $\underline{\tilde{d}} = \tilde{d} = \left(s, \overline{Pert}, RL_{lim}\right)$;
    $ParetoFront = ParetoFront \bigcup (|\overline{Pert}|, \underline{\tilde{d}})$
    $\tilde{d} = \tilde{d} + d$;
  end while
  $|Pert| = |\overline{Pert}|$;
  while $(|Pert| > 0)$ **do**
    Execute Algorithm 3 $\underline{\tilde{d}} = \tilde{d}\,(s, Pert, RL_{lim})$;
    Execute Algorithm 4 : $\overline{Pert} = Pert\left(s, \underline{\tilde{d}}, RL_{lim}\right)$;
    $ParetoFront = ParetoFront \bigcup (\overline{Pert}, \underline{\tilde{d}})$
    $|Pert| = |Pert| - pe$;
  end while
  return $ParetoFront$

2.3.1.8 EV_8 : Analyse du compromis sensibilité/stabilité/niveau de service

Dans le dernier problème d'évaluation identifié, le nombre de paramètres à trouver augmente pour analyser le compromis global entre les différents paramètres (EV_8). En effet, pour ce problème tous les éléments sont recherchés, la deadline $\tilde{d}$, l'ensemble de perturbations $Pert$ et le niveau de service RL_{lim}. L'objectif de ce problème est de trouver un compromis entre tous les paramètres du problème. La question formulée à partir de ce problème est : *Étant donné un ordonnancement s, quel est le niveau de service maximal pour une deadline minimale et un ensemble maximal de perturbations ?*

Comme pour les problèmes précédents, on recherche ici une surface de Pareto mais cette fois-ci en trois dimensions liées aux trois paramètres recherchés.

Les algorithmes proposés précédemment pour analyser le compromis entre deux paramètres sont des algorithmes efficaces au sens qu'ils permettent de trouver le front de Pareto avec un niveau de précision voulu même s'ils ne sont pas forcément efficients. La recherche du front de Pareto est souvent utilisée pour résoudre des problème multi-objectifs. Plus d'informations sur la construction d'un front de Pareto sont fournies dans l'annexe D.

2.3.2 Problèmes d'optimisation

Les problèmes d'optimisation permettent de traiter un autre aspect de la robustesse. Pour les problèmes d'évaluation il s'agissait d'analyser les performances de robustesse d'un ordonnancement donné s selon différent point de vue (sensibilité, stabilité, niveau de service).

Pour les problèmes d'optimisation, il s'agit de calculer une solution d'ordonnancement s^* qui optimise une ou des performances de robustesse.

En faisant varier les paramètres connus et inconnus, il est possible comme pour les problèmes

d'évaluation, de définir huit problèmes d'optimisation. Ils sont en fait le pendant des problèmes d'évaluation : tout paramètre pouvant être évalué pour une solution s connue peut faire l'objet d'une optimisation dès que la solution n'est pas connue et doit être calculée. Ces problèmes sont identifiés et détaillés dans la table 2.2.

Un algorithme naïf (Algorithm 8) permettant de résoudre n'importe quel problème d'optimisation consiste en deux phases répétées itérativement jusqu'à atteindre une solution optimale. La première phase consiste à générer une solution s et la deuxième phase consiste à évaluer cette solution selon la performance de robustesse sélectionnée (en appliquant l'algorithme correspondant au problème d'évaluation). Une nouvelle itération est lancée si la solution n'est pas optimale (selon un critère permettant d'arbitrer sur l'optimalité de la solution ou non).

Dans cet algorithme, "Input Parameters" sont les paramètres d'entrées du problème (identifiés grâce à la deuxième colonne du tableau 2.2) et "Output Parameters" correspondent aux paramètres de sortie (identifiés grâce à la troisième colonne du tableau 2.2).

Algorithm 8 Algorithme général de résolution des problèmes d'optimisation

Require: Input Parameters $(i_1, \ldots, i_n)$, Output Parameters z ;
  while Optimality=False **do**
    Generate a new solution s ;
    Evaluate $z\,(s, i_1, \ldots, i_n)$;
  end while
  return s

La performance de cet algorithme dépend fortement : (i) de sa capacité à générer des solutions admissibles, (ii) de la capacité à vérifier le caractère optimal d'une solution (c'est parfois loin d'être un problème facile que de trouver un critère permettant d'arbitrer sur l'optimalité d'une solution ou non).

De manière basique, cet algorithme pourrait se contenter d'énumérer toutes les solutions et de ne garder à la fin que la meilleure. Le critère d'optimalité serait ici : "toutes les solutions ont été parcourues". En pratique, cette tactique est très mauvaise puisque souvent le nombre de solutions qui existent est exponentiel. Tout l'enjeu est donc de trouver un moyen de parcourir toutes les solutions de manière implicite. Cependant, ceci est en dehors des objectifs de cette thèse.

2.4 Conclusion

Ce chapitre propose une formalisation des problèmes d'ordonnancement robuste se basant sur le niveau de service comme performance de robustesse. Là où on aurait pu se contenter de chercher à évaluer puis éventuellement maximiser cette seule performance, nous avons montré dans ce chapitre que le problème d'ordonnancement robuste est beaucoup plus riche. En effet, l'ordonnancement robuste est un problème multi-facettes qui peut être abordé selon différents points de vue (stabilité, sensibilité, niveau de service). Il nous semble important, dès lors que l'objectif final est d'aider le décideur, de pouvoir préserver ces différents points de vue dans un cadre d'étude pour l'ordonnancement robuste. Cela a permis de mettre en évidence huit problèmes d'évaluation de la robustesse et leur problème d'optimisation associé. Nous illustrerons dans le chapitre 5 comment ces problèmes répondent effectivement aux préoccupations du décideur dans un contexte réel.

Concernant les problèmes d'évaluation, nous avons proposé des algorithmes naïfs (donc certainement peu efficients) qui font tous appel au problème d'analyse du niveau de service (problème EV_2 – Algorithm 1). Nous avons fait l'hypothèse dans ce chapitre qu'il existait une approche afin d'évaluer $RL\left(s, Pert, \tilde{d}\right)$. L'objectif du prochain chapitre est de proposer une telle approche.

ID	Paramètres d'entrée	Paramètres de sortie	Type de problème	Spécification
EV_1	$s, Pert,$ $\tilde{d}$, RL_{lim}	Booléen : vrai ou faux	Qualification de la robustesse	Est-ce que s est robuste ? $RL\left(s, Pert, \tilde{d}\right) \geq RL_{lim}$?
EV_2	s, $Pert$, $\tilde{d}$	$RL\left(s, Pert, \tilde{d}\right)$	Évaluation du niveau de service	Quelle est la valeur de $RL\left(s, Pert, \tilde{d}\right)$?
EV_3	s, $Pert$, RL_{lim}	$\underline{\tilde{d}}$	Analyse de sensibilité	Quelle est la valeur minimum de $\tilde{d}$ notée $\underline{\tilde{d}}$ telle que : $RL\left(s, Pert, \underline{\tilde{d}}\right) \geq RL_{lim}$?
EV_4	s, $\tilde{d}$, RL_{lim}	$\overline{Pert}$	Analyse de stabilité	Quelle est le plus grand ensemble $Pert$ noté $\overline{Pert}$ tel que : $RL\left(s, \overline{Pert}, \tilde{d}\right) \geq RL_{lim}$?
EV_5	s, $Pert$	$\overline{RL_{lim}}$, $\underline{\tilde{d}}$	Analyse du compromis sensibilité/niveau de service	Quel est le couple $\left(\overline{RL_{lim}}, \underline{\tilde{d}}\right)$, tel que : $RL\left(s, Pert, \underline{\tilde{d}}\right) \geq \overline{RL_{lim}}$?
EV_6	s, $\tilde{d}$	$\overline{RL_{lim}}$, $\overline{Pert}$	Analyse du compromis stabilité/niveau de service	Quel est le couple $\left(\overline{RL_{lim}}, \overline{Pert}\right)$, tel que : $RL\left(s, \overline{Pert}, \tilde{d}\right) \geq \overline{RL_{lim}}$?
EV_7	s, RL_{lim}	$\underline{\tilde{d}}$, $\overline{Pert}$	Analyse du compromis sensibilité/stabilité	Quel est le couple $\left(\overline{Pert}, \underline{\tilde{d}}\right)$, tel que : $RL\left(s, \overline{Pert}, \underline{\tilde{d}}\right) \geq RL_{lim}$?
EV_8	s	$\underline{\tilde{d}}$, $\overline{Pert}$, $\overline{RL_{lim}}$	Analyse du compromis sensibilité/stabilité/niveau de service	Quels sont $\left(\underline{\tilde{d}}, \overline{Pert}, \overline{RL_{lim}}\right)$, telles que : $RL\left(s, \overline{Pert}, \underline{\tilde{d}}\right) \geq \overline{RL_{lim}}$?

TABLE 2.1 – Problèmes d'évaluation de la robustesse

ID	Paramètres d'entrée	Paramètres de sortie	Type de problème	Spécification
OP_1	$Pert$, $\tilde{d}$, RL_{lim}	s	Problème de faisabilité	Quel est s, tel que : $RL\left(s, Pert, \tilde{d}\right) \geq RL_{lim}$
OP_2	$Pert, \tilde{d}$	s^*, $\max\{RL_{lim}\}$	Maximisation du niveau de service	Quel est s^* qui maximise RL_{lim} tel que : $RL\left(s^*, Pert, \tilde{d}\right) \geq \max\{RL_{lim}\}$
OP_3	$Pert, RL_{lim}$	s^*, $\min\left\{\tilde{d}\right\}$	Minimisation de la sensibilité	Quel est s^* qui minimise $\tilde{d}$, tel que $RL\left(s^*, Pert, \min\left\{\tilde{d}\right\}\right) \geq RL_{lim}$
OP_4	$\tilde{d}$, RL_{lim}	s^*, $\max\{Pert\}$	Maximisation de la stabilité	Quel est s^* qui maximise $Pert$, tel que : $RL\left(s^*, \max\{Pert\}, \tilde{d}\right) \geq RL_{lim}$
OP_5	$Pert$	s^*, $\max\{RL_{lim}\}$, $\min\left\{\tilde{d}\right\}$	Maximisation du niveau de service et minimisation de la sensibilité	Quel est s^* qui maximise RL_{lim} et minimise $\tilde{d}$ tel que : $RL\left(s, Pert, \min\left\{\tilde{d}\right\}\right) \geq RL_{lim}$
OP_6	$\tilde{d}$	s^*, $\max\{RL_{lim}\}$, $\max\{Pert\}$	Maximisation du niveau de robustesse et maximisation de la stabilité	Quel est s^* qui maximise RL_{lim} et $Pert$ tel que : $RL\left(s^*, \max\{Pert\}, \tilde{d}\right) \geq \max\{RL_{lim}\}$
OP_7	RL_{lim}	s^*, $\min\left\{\tilde{d}\right\}$, $\max\{Pert\}$	Minimisation de la sensibilité et maximisation de la stabilité	Quel est s^* qui minimise $\tilde{d}$ et maximise $Pert$ tel que : $RL\left(s^*, \max\{Pert\}, \min\left\{\tilde{d}\right\}\right) \geq RL_{lim}$
OP_8	$\emptyset$	s^*, $\max\{RL_{lim}\}$, $\min\left\{\tilde{d}\right\}$ $\max\{Pert\}$,	Maximisation du niveau de service et minimisation de la sensibilité et maximisation de la stabilité	Quel est s^* qui maximise RL_{lim} et minimise $\tilde{d}$ et maximise $Pert$ tel que : $RL\left(s^*, \max\{Pert\}, \min\left\{\tilde{d}\right\}\right) \geq \max\{RL_{lim}\}$

TABLE 2.2 – Problèmes d'optimisation de la robustesse

Chapitre 3

Évaluation du niveau de service par automates stochastiques

Sommaire

3.1 Introduction 46
3.2 Présentation du processus d'évaluation 46
3.2.1 Approche proposée 46
3.2.2 Présentation de l'étude de cas 47
3.3 Modélisation du problème : ordonnancement et perturbations . . . 48
3.3.1 Modèles d'automates temporisés stochastiques 48
3.3.2 Structure de modélisation 53
3.3.3 Modélisation de l'ordonnancement déterministe 54
3.3.4 Modélisation des perturbations 57
3.3.5 Simulation de l'évolution des modèles 67
3.4 Évaluation du niveau de service 69
3.4.1 Modélisation de la propriété d'analyse du niveau de service 69
3.4.2 Model-checking probabiliste 73
3.5 Implémentation du processus d'évaluation 76
3.5.1 Implémentation du processus d'évaluation dans UppAal SMC 76
3.6 Conclusion 80

3.1 Introduction

Comme nous avons pu le voir dans l'état de l'art (cf. Chapitre 1), les approches d'évaluation de la robustesse traitent généralement le problème de manière dédiée et dépendante du type de paramètres d'entrée. Dans le contexte de l'aide à la décision et dans le cadre de l'industrie 4.0, l'applicabilité de ces approches peut donc être remise en question. En effet, l'évolution permanente des ateliers génère des contraintes d'agilité et de flexibilité aux changements de configuration, de produits et des stratégies de production. Ces contraintes nécessitent donc que le processus d'aide à la décision soit générique afin de répondre rapidement et facilement à de nouvelles situations de l'atelier ou à de nouveaux problèmes. Le verrou scientifique identifié à partir de cette exigence est donc : **Comment traiter le problème de la robustesse de l'ordonnancement soumis aux perturbations tout en étant générique face aux paramètres d'entrée ?**

Pour traiter ce verrou, nous avons identifié et classifié un ensemble de problèmes de robustesse dans le chapitre 2. Tous les problèmes de robustesse définis se basent sur l'analyse du niveau de service $RL(s, Pert, \tilde{d})$ (problème $EV2$). Dans le présent chapitre, nous proposons un processus d'évaluation permettant l'analyse du niveau de service pour un ordonnancement donné s.

Le processus d'évaluation proposé dans ce chapitre doit être générique et agile pour prendre en considération :

— Différents types d'ateliers de production ;

— Différentes perturbations.

Dans ce qui suit, une vue d'ensemble du processus d'évaluation est présentée puis la mise en œuvre de ce dernier est détaillé. Tout au long du chapitre, un cas d'étude est traité afin d'illustrer chaque étape du processus.

3.2 Présentation du processus d'évaluation

3.2.1 Approche proposée

L'objectif du processus est d'obtenir des informations sur la robustesse d'un ordonnancement donné (Figure 3.1), à destination d'un décideur. En effet, dans ce contexte, le décideur doit choisir l'ordonnancement qui répond, non seulement aux contraintes de l'atelier de production, mais également qui tient compte des perturbations qui peuvent se produire pendant l'exécution de celui-ci. Pour calculer le niveau de service, le décideur dispose des éléments d'entrée suivants :

— L'ordonnancement déterministe s : le séquencement des opérations sur les ressources et le Makespan déterministe (i.e. la durée totale de l'ordonnancement déterministe) $C^{ref}_{max}(s)$.

— L'ensemble de perturbations $Pert$ à prendre en compte.

— La deadline $\tilde{d}$ que doit satisfaire l'ordonnancement perturbé.

En sortie du processus d'évaluation le décideur obtient le niveau de service RL, qui est la performance de la robustesse de l'ordonnancement d'entrée sous perturbations pour $\tilde{d}$.

Le processus d'évaluation est constitué de deux étapes en phase d'exploitation (Figure 3.1) : instanciation et évaluation du problème de robustesse. La conception du processus d'évaluation passe donc par une première étape de modélisation du comportement de l'ordonnancement et celui des perturbations. Cette étape permet de modéliser les caractéristiques de l'ordonnancement d'entrée s et ceux de l'ensemble de perturbations $Pert$ à considérer. La modélisation du problème

d'évaluation se veut générique face aux différents types d'ordonnancement et des différentes perturbations.

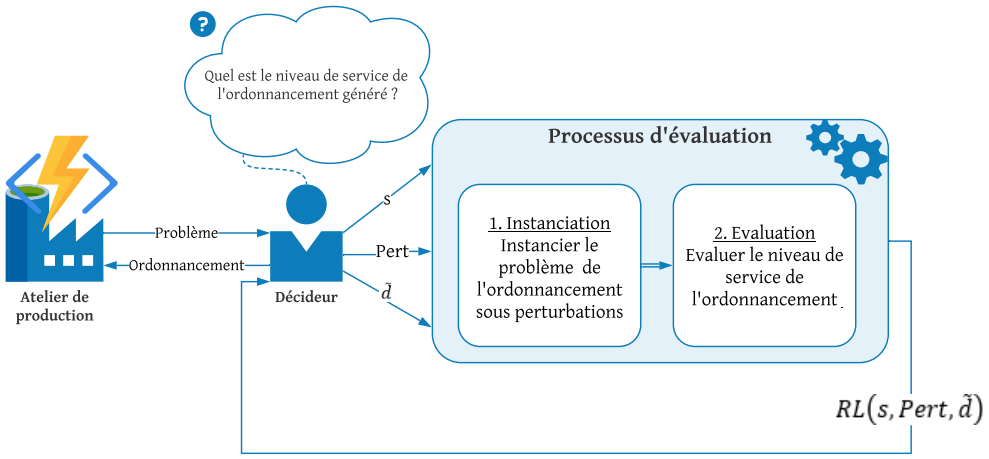


FIGURE 3.1 – Processus d'évaluation de robustesse

La deuxième étape, pour permettre l'évaluation de la robustesse, est la modélisation de la spécification du niveau de service définie dans le chapitre précédent. L'objectif de cette étape est d'implémenter l'indicateur de robustesse afin de pouvoir vérifier ce dernier sur les modèles générés lors de l'étape précédente.

La sortie du processus d'évaluation est le niveau de service $RL(s, Pert, \tilde{d})$ qui est associé à l'ordonnancement et aux perturbations considérées.

3.2.2 Présentation de l'étude de cas

Afin d'illustrer le processus d'évaluation, nous utilisons, tout au long de ce chapitre l'étude de cas suivante adaptée de [Giard, 2003].

Le système de production étudié dans ce cas, est un "Flexible Job Shop" avec 8 jobs (35 opérations) et 7 ressources. L'atelier comprend un tour CNC (r_1), une rectifieuse (r_2), trois fraiseuses (r_3 à r_5) et deux fours (r_6 et r_7) (Figure 3.2). Nous considérons que le décideur a besoin d'évaluer un ordonnancement déterministe s obtenu en considérant que les données d'entrées sont certaines et stables. Les informations déterministes sont extraites du diagramme de Gantt établi (Figure 3.3). En effet, les données récupérées sont le Makespan déterministe ($C^{ref}_{max}(s)$, dans ce cas $38UT$), les durées de référence de l'atelier de production ainsi que les contraintes de précédences dans la gamme de job et sur les ressources (Table 3.1).
Les perturbations prises en compte dans le cadre de l'exécution du processus d'évaluation sont également définies par le décideur.

Les perturbations généralement traitées dans la littérature, sont les incertitudes sur les durées (en particulier sur les durées d'exécution des opérations) et les pannes ressources. Pour illustrer notre approche, nous considérons les trois perturbations suivantes.

— Incertitudes sur les durées d'exécution,
— Panne ressource,
— Incertitude sur les durées de réparation.

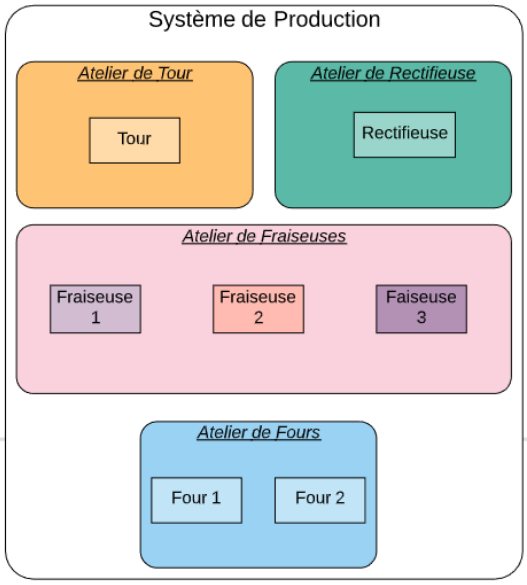


FIGURE 3.2 – Atelier de production (Étude de cas de [Giard, 2003])

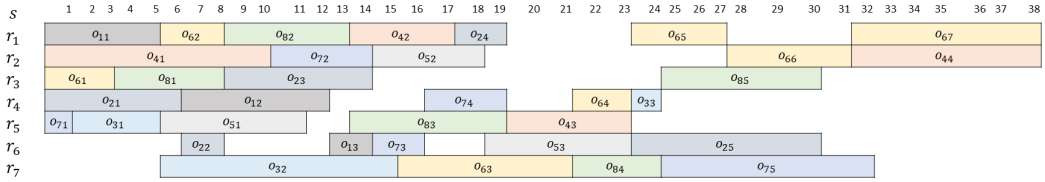


FIGURE 3.3 – Diagramme de Gantt de s pour l'étude de cas de [Giard, 2003]

3.3 Modélisation du problème : ordonnancement et perturbations

Cette partie présente l'approche de modélisation choisi pour représenter le comportement de l'ordonnancement et des perturbations.

3.3.1 Modèles d'automates temporisés stochastiques

Pour modéliser le comportement des ordonnancements et des perturbations, les modèles choisis doivent tenir compte (i) des caractéristiques dynamiques de l'atelier (ii) le temps d'exécution pour permettre le traitement du C_{max} et (iii) le comportement probabiliste des perturbations. Plusieurs modèles basés sur les SED stochastiques permettent la modélisation de ces caractéristiques. À titre d'exemple, les réseaux de Petri stochastiques [Chiola et al., 1993], les automates temporisés stochastiques [Kwiatkowska et al., 2006], les réseaux d'automates stochastiques [Plateau and Atif, 1991], les chaînes de Markov [Kemeny and Snell, 1976].

Dans le contexte de l'ordonnancement, plusieurs travaux ont choisi les SED pour traiter le problème d'ordonnancement déterministe. Dans ce contexte, [Panek et al., 2006] et [Abdeddaïm and Maler, 2001] proposent de modéliser un ordonnancement déterministe répondant aux contraintes de l'atelier. Dans [Ballarini et al., 2011], les réseaux de Petri sont choisis pour traiter

Job	Opération	R	Précédence gamme	Précédence séquence	d_{jkr}^{ref}
1	o_{11}	r_1	$--$	$--$	5
1	o_{12}	r_4	o_{11}	o_{12}	6
1	o_{13}	r_6	o_{12}	o_{22}	2
2	o_{21}	r_4	$--$	$--$	6
2	o_{22}	r_6	o_{21}	$--$	2
2	o_{23}	r_3	o_{22}	o_{81}	6
2	o_{24}	r_1	o_{23}	o_{42}	2
2	o_{25}	r_6	o_{24}	o_{53}	7
3	o_{31}	r_5	$--$	o_{71}	4
3	o_{32}	r_7	o_{31}	$--$	10
3	o_{33}	r_3	o_{32}	o_{64}	1
4	o_{41}	r_2	$--$	$--$	10
4	o_{42}	r_1	o_{41}	o_{82}	4
4	o_{43}	r_5	o_{42}	o_{83}	4
4	o_{44}	r_2	o_{43}	o_{66}	7
5	o_{51}	r_5	$--$	o_{31}	6
5	o_{52}	r_2	o_{51}	o_{72}	4
5	o_{53}	r_6	o_{52}	o_{73}	5
6	o_{61}	r_3	$--$	$--$	3
6	o_{62}	r_1	o_{61}	o_{11}	3
6	o_{63}	r_7	o_{62}	o_{63}	6
6	o_{64}	r_4	o_{63}	o_{74}	2
6	o_{65}	r_1	o_{64}	o_{24}	4
6	o_{66}	r_2	o_{65}	o_{52}	4
6	o_{67}	r_1	o_{66}	o_{65}	7
7	o_{71}	r_5	$--$	$--$	1
7	o_{72}	r_2	o_{71}	o_{42}	4
7	o_{73}	r_6	o_{72}	o_{13}	2
7	o_{74}	r_4	o_{73}	o_{12}	3
7	o_{75}	r_7	o_{74}	o_{84}	7
8	o_{81}	r_3	$--$	o_{61}	5
8	o_{82}	r_1	o_{81}	o_{62}	5
8	o_{83}	r_5	o_{82}	o_{51}	6
8	o_{84}	r_7	o_{83}	o_{63}	3
8	o_{85}	r_3	o_{84}	o_{23}	6

TABLE 3.1 – Données de l'atelier de production pour s

les systèmes flexibles de production. Pour prendre en compte les perturbations, les automates temporisés sont utilisé pour modéliser des incertitude dans [Abdeddaïm et al., 2006]. Pour traiter une perturbation de type aléa [Lefebvre and Mejia, 2018] ont choisi les réseaux de Petri. Dans [Yao and Cassandras, 2012], les auteurs utilisent les files d'attente pour traiter ce problème alors que [Tan, 2002] se base sur les processus Markoviens pour prendre en compte les perturbations dans le problème d'ordonnancement.

Parmi les modèles SED stochastiques, les Réseaux de Petri stochastiques (RPS) permettant la modélisation de la propriété temporelle du système de production et ces caractéristiques stochastiques [Bause and Kritzinger, 2002]. Pour pouvoir utiliser les RPS dans la modélisation des systèmes de production [Ballarini et al., 2011] propose de se baser sur la sémantique des processus stochastiques à événements discrets ($DESP$). La structure des $DESP$, permet non seulement de modéliser la propriété stochastique du système, mais également de prendre en compte son évolution temporelle.

Les réseaux d'automates stochastiques (RAS) sont utilisés pour modéliser des systèmes complexes [Stewart et al., 1995]. En effet, les RAS permettent de modéliser des systèmes parallèles tels que les processus de communication et le partage de mémoire [Plateau and Atif, 1991]. Les

RAS sont constitués de plusieurs automates stochastiques qui fonctionnent indépendamment les uns des autres. Ce langage permet donc une modélisation modulaire via plusieurs automates communicants à l'aide d'événements synchronisant. Les caractéristiques stochastiques du système sont également prises en compte dans la sémantique de modélisation.

L'utilisation des automates temporisés a permis de générer des ordonnancements dans [Marangé et al., 2016] . L'extension stochastique : les automates temporisés stochastiques ATS permettent d'associer le principe de modélisation des RAS avec les caractéristiques temporelles et stochastiques. En effet, les ATS sont une extension des automates temporisés discrets définis par [Alur and Dill, 1994].

Dans le cadre de cette thèse, nous avons choisi de réaliser la modélisation en automates temporisés stochastiques (ATS).

3.3.1.1 Automates temporisés stochastiques : définition

La définition formelle des automates temporisés stochastiques ATS est basée sur la définition des automates temporisés introduit par [Alur and Dill, 1994] Les ATS sont donc des automates temporisés enrichis par le partage de variables, les événements synchronisant et les caractéristiques probabilistes [Larsen et al., 1997] (Définition 4) .

Définition 4. *Formellement, un automate stochastique temporisé est représenté comme le n-uplet* $A = (L, V, E, C, Inv, P, T, L_m, l_0, v_0)$ *avec :*

- L *est un ensemble fini de localités.*
- V *est un ensemble fini de variables.*
- E *est un ensemble fini d'événements synchronisant avec* $E = E_u \cup E_{\bar{u}}$.
  - E_u *est un ensemble fini d'événements urgents. Les événements urgents occurrent sans consommer de temps dès que l'événement peut se produire.*
  - $E_{\bar{u}}$ *est un ensemble fini d'événements non urgents.*
- C *est un ensemble fini d'horloges.*
- Inv *est un ensemble fini d'invariants.*
- P *est un ensemble de probabilités : (i) discrète sur l'ensemble des transitions, à partir d'une localité de départ* l_j*, une transition peut conduire dans différentes localités d'arrivées* l_i *avec des probabilités* p_{ji} *où* $\sum_i p_{ji} = 1$ *. (ii) continue sur des variables d'automates, la condition de franchissement d'une transition est définie de façon aléatoire par une distribution de probabilités.*
- T *est un ensemble fini de fonctions de transition* $(l, e, g, m, l') \in L \times E \times G \times M \times L$ *où* l *et* l' *sont respectivement les localités de début et d'arrivée. Une transition est définie par trois éléments : (i) Une garde (condition sur les variables)* $g \in G$*, (ii) Une mise à jour (sur les variables ou les horloges),*$m \in M$*, (iii) L'événement de synchronisation* $e \in E$.
- $L_m \subseteq L$ *est l'ensemble des localités marquées.*
- $l_0 \in L$ *est la localité initiale de l'automate.*
- v_0 *est le vecteur d'initialisation des variables.*

Graphiquement, un automate temporisé stochastique est représenté par l'ensemble des éléments suivants (un exemple graphique est donné dans la figure 6) :

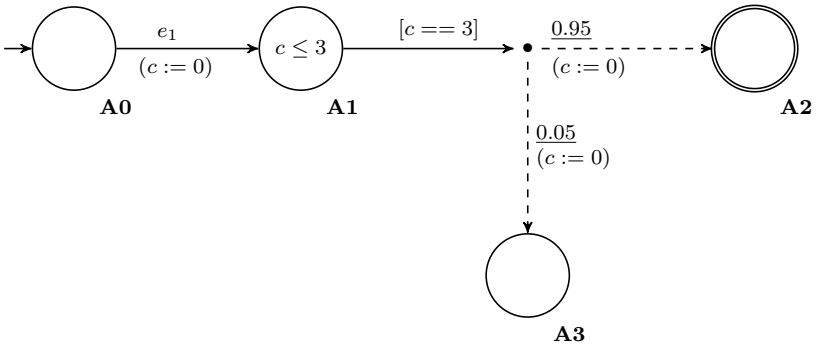
- Un ensemble de sommets représentant les localités avec un label pour chaque localité ;
- Un ensemble d'arcs présentant les transitions ;
- Un arc vide au début de l'automate pour modéliser la localité initiale ;

— Un double sommet pour modéliser les localités initialement marquées ;
— L'invariant est représenté à l'intérieur du sommet ;
— Sur les arcs, les éléments d'une transition t sont modélisé :
  — Les gardes sont modélisées entre crochets [ ] ;
  — Les évènements synchronisants sont notés en italique au dessus des arcs ;
  — Les mises à jour sont notées entre parenthèses ().
— Les probabilités discrètes sont modélisées par les arcs en pointillés. Le poids de probabilité discrète est souligné.
— Les probabilités continues sont directement liées à la déclaration de la variable probabiliste représentée en écriture gras.

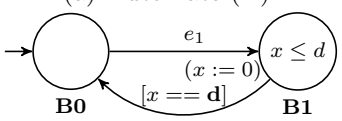
Un état d'un ATS définit la situation globale où l'ATS se trouve dans une localité l_i avec un vecteur v_i des valeurs des variables déclarées et c le vecteur des valeurs des horloges. À partir de l'exemple (Figure 3.4), la sémantique de modélisation des ATS est expliquée. Plus précisément, la modélisation du temps, la synchronisation entre automates et la modélisation stochastique sont présentées.

Exemple 6. *Considérons deux automates ATS (A) et (B) (Figure 3.4) comme exemple représentatif de la construction de deux automates synchronisés à l'aide d'un événement synchronisant. Trois variables sont définies : deux horloges c pour (A) et x pour (B) et un événement synchronisant* e_1.

Le comportement des deux automates est contraint par la synchronisation de l'événement e_1 *et le comportement probabiliste est modélisé par deux arcs probabilistes ayant chacun un poids de probabilité différent .*



(a) Automate (A)



(b) Automate (B)

FIGURE 3.4 – Exemple d'un automate temporisé stochastique

L'évolution des ATS peut être du à l'écoulement du temps, à l'occurrence d'un événement de synchronisation et au franchissement de transitions probabilisées. Les trois parties suivantes permettent de détailler ces trois aspects.

Évolution par écoulement du temps

Un automate temporisé comporte un ensemble fini de localités L et de transitions T. Les transitions permettent le passage d'une localité à une autre si l'ensemble des contraintes de la transition est satisfait. En plus de prendre en compte les variables du système, l'automate temporisé stochastique permet également de représenter les horloges qui permettent une modélisation discrète du temps.

En d'autres termes, dans chaque localité, le temps est susceptible de s'écouler et de faire évoluer le modèle. La valeur d'une horloge c est donc mise à jour à chaque moment écoulé, et ce, à partir de sa dernière mise à zéro. Pour limiter le temps s'écoulant dans une localité, la définition d'un invariant sur c est possible. L'invariant exprime une limite que c peut atteindre en étant dans la localité. Si cette limite est atteinte, l'automate évolue obligatoirement vers la localité suivante. Lors de l'utilisation de ce modèle pour faire de la vérification par exemple, chaque évolution possible à chaque pas de temps devra être analysée.

Sur la transition, une garde peut être définie sur c. Une garde est une condition de franchissement de la transition. Si celle-ci n'est pas satisfaite, la transition ne peut être franchie. Lorsque plusieurs horloges sont utilisées, elles évoluent de manière synchronisées.

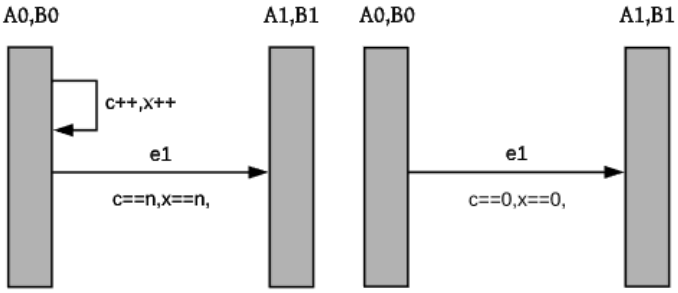
Dans l'exemple 6, l'occurrence de l'événement e_1, le franchissement de ces transitions met à 0 l'horloge c dans l'automate (A) et l'horloge x dans l'automate (B) ($c := 0$ et $x := 0$). À partir de ces localités, les automates évoluent en fonction de leur horloge. Dans l'automate (A), l'invariant de la localité **A1** permet de contraindre l'horloge c à une limite supérieure 3. Cet invariant implique que l'horloge ne doit pas dépasser la valeur de 3 dans la localité **A1**. Pour s'assurer que l'horloge va atteindre cette limite, une contrainte est ajoutée sur les transitions sortante de **A1**, cette contrainte est sous forme de garde ($[c == 3]$). La garde permet d'exprimer le fait que l'automate ne peut évoluer que quand la valeur de l'horloge c est égale à 3.

Évolution due à un événement de synchronisation

Les ATS permettent la modélisation modulaire du comportement global d'un système par le biais de sa décomposition en sous-systèmes (i.e. plusieurs automates). Ils se basent sur un réseaux d'automates communiquants dont les évolutions peuvent être conditionnées par des événements de synchronisation : deux transitions appartenant à deux automates distincts sont franchies simultanément si leurs gardes sont validées et si elles sont labelisées par même événement de synchronisation.

Lors de la simulation du comportement des automates, toutes les situations doivent être simulées : (i) rester dans la localité ou (ii) franchir la transition. Cette possibilité peut conduire des automates dans une situation de blocage deadlock (i.e. blocage dans l'évolution des automates). Pour éviter cette situation, il y a deux possibilités soit ajouter un invariant dans les localités soit définir l'événement comme urgent $E_u \in E$.

Dans l'exemple, l'événement e_1 permet la synchronisation entre les deux automates A et B. Si l'événement n'est pas urgent $e_1 \in E_{\bar{u}}$ (Figure 3.5a), alors les automates A et B peuvent rester dans leurs localités initiales et se synchroniser à un instant donné n. Dans le cas où ce comportement n'est pas souhaité et que l'évolution des automates doit être à vitesse maximale,



(a) Événement e_1 non urgent (b) Événement e_1 urgent

FIGURE 3.5 – Évolution de l'événement e_1 (Exemple 6)

deux actions sont possibles :

— Ajouter un invariant dans les deux localités **A0,B0**. En effet, pour ne pas consommer de temps dans ces localités, nous pouvons considérer une horloge globale clk qui va être contrainte par $clk \leq 0$. Cet invariant commun aux deux automates va permettre de sortir au plus tôt des localités initiales et ainsi se synchroniser à l'aide de l'événement e_1.

— Déclarer un l'événement e_1 comme urgent $e_1 \in E_u$ (Figure 3.5b). Dans ce cas, la synchronisation est effectuée dès que l'événement se produit dans les deux automates permettant ainsi une évolution instantanée vers les prochaines localités.

Évolution par franchissement d'une transition probabiliste

Les automates temporisés stochastiques permettent de considérer la modélisation des caractéristiques probabilistes des systèmes. Ce langage de modélisation permet d'une part de modéliser les choix probabilistes discrets, en se basant sur les processus Markoviens de décision et d'autre part de modéliser une probabilité continue souvent limité à la distribution exponentielle.

A partir d'une localité l, plusieurs transitions probabilistes P peuvent être modélisées. Lorsque l'automate franchit une transition probabiliste discrète, la localité atteinte dépend du poids de probabilité p qui lui est associé. La somme des poids de probabilité des transition liées à la localité l doit impérativement être égale à 1.

Dans l'exemple (6), les deux types de probabilités sont exprimées. En effet, dans l'automate (A), deux transitions probabilistes sont modélisées. À partir de la localité **A1**, avec un poids de 0.95 l'automate évolue vers la localité **A2** alors qu'avec un poids 0.05 il évolue vers la localité **A3**.
Pour les probabilités continues, l'automate (B) l'exprime à l'aide de la variable d. En effet, la valeur de cette variable dépend de sa distribution de probabilité.

3.3.2 Structure de modélisation

Afin d'avoir une modélisation adaptable aux caractéristiques d'un atelier, et celles des perturbation, une structure de modélisation est proposée dans ce qui suit (Figure 3.6). Pour modéliser à la fois l'ordonnancement et les perturbations, nous proposons une approche modulaire basée sur le concept de *plug and play* [Himmiche et al., 2019]. En effet, notre approche est basée sur

plusieurs patrons qui permettent de représenter le comportement et la dynamique des éléments de l'ordonnancement et des éventuelles perturbations qui se produiront durant l'exécution de l'ordonnancement. Ce principe de modélisation permet ainsi de contribuer la généricité des modèles.

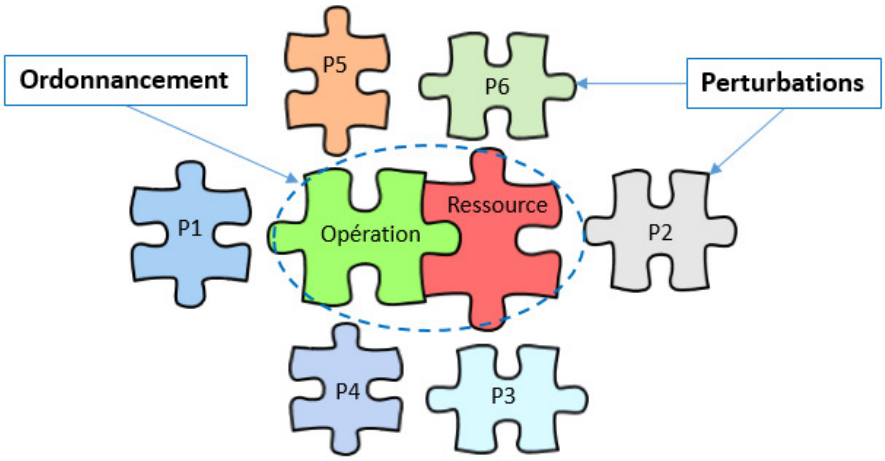


FIGURE 3.6 – Structure de modélisation

Les deux composants essentiels d'un ordonnancement sont les opérations et les ressources. D'une part, le premier patron représente le comportement de l'opération à exécuter alors que le deuxième modélise le comportement de la ressource qui exécute l'opération. D'autre part, les perturbations sont représentées par deux patrons distincts qui modélisent soit le comportement d'une incertitude soit celui d'un aléa.

La modélisation par patron permet de décrire les comportements des différents éléments une seule fois mais de les dupliquer autant de fois que d'éléments existants dans le problème traité. Par exemple, un patron d'opération va décrire le comportement d'une opération générique o_{jk} et son instanciation par $j = 1$ et $i = 1$ permettra de modéliser l'opération o_{11}.
La communication entre les différents patrons est basée sur les événements synchronisants et le partage de variables. Les modèles de perturbations évoluent tout d'abord et déterminent l'impact des perturbations sur les durées d'exécution des opérations pour permettre la prise en compte de la fluctuation du temps liée à celles-ci. Cette initialisation des perturbations effectuées, les patrons d'opérations peuvent ensuite évoluer en fonction des contraintes (gamme, séquence, disponibilité) satisfaites. Pour les opérations pouvant être réalisées, les patrons d'opération seront alors synchronisé avec ceux des ressources associées. À la fin de l'exécution, la ressource et l'opération seront synchronisées par l'événement de fin d'exécution (Figure 3.7).

Par la suite, nous présentons ces patrons en commençant par ceux de l'ordonnancement puis les patrons de perturbations. À la fin de cette section, l'interaction entre les perturbations et l'ordonnancement est présentée.

3.3.3 Modélisation de l'ordonnancement déterministe

Selon la définition proposée précédemment (Définition 4), une solution d'ordonnancement est :

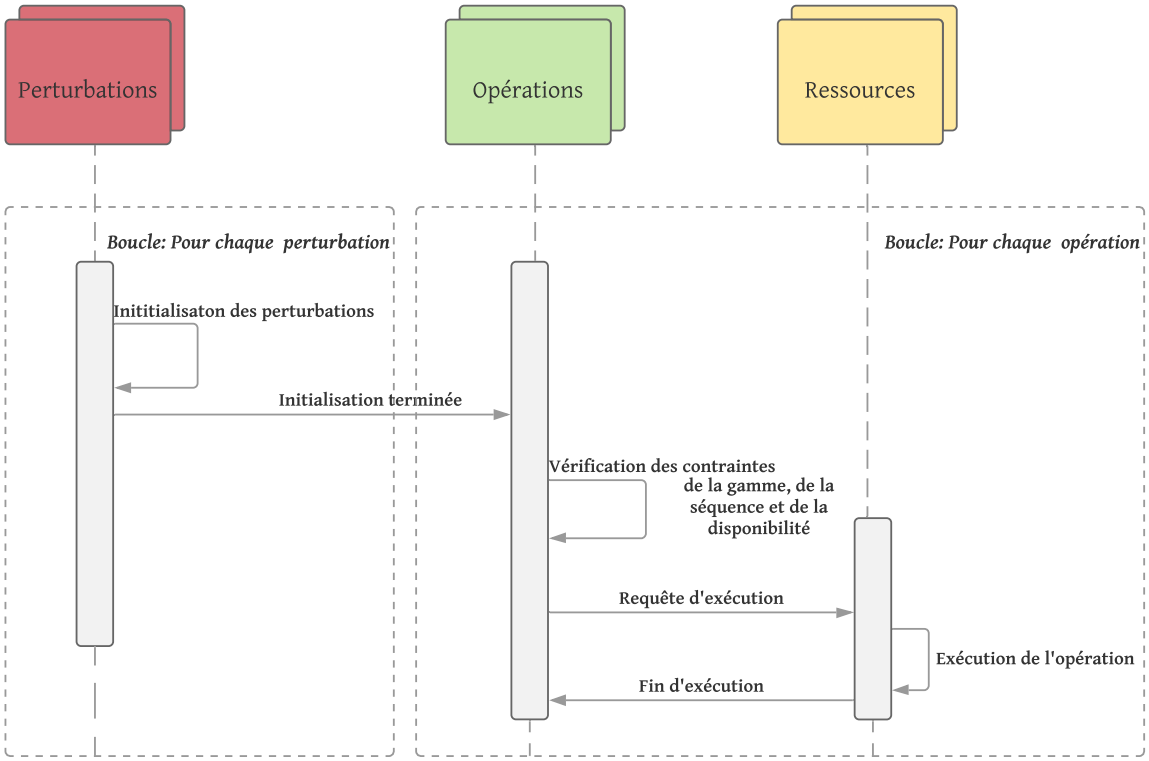


FIGURE 3.7 – Évolution des modèles proposés

i) une affectation de ressources aux opérations,
ii) un séquencement des opérations sur les ressources affectées.

Une solution d'ordonnancement permet de définir les contraintes de précédence entre les opérations sur chaque ressource (précédence liée à la séquence de la ressource) et vérifier les contraintes de précédence dues à la gamme du job (selon le type d'atelier de production). Pour modéliser le comportement de l'ordonnancement déterministe, nous proposons l'instanciation des deux patrons (opération + ressource). Les variables nécessaires (utilisées dans la modélisation) et leurs notations sont définies dans la Table 3.2.

3.3.3.1 Patron d'opération

Le patron d'opération (Figure 3.8a) représente le comportement d'une opération o_{jk}. Á partir de la localité **Idle**, les conditions d'exécution de l'opération o_{jk} sont vérifiées à l'aide de la garde contenant trois contraintes : (i) la première liée à la gamme du job j vérifiant que l'opération précédente à o_{jk} dans la gamme du job j a fini d'être exécutée ($Route(o_{jk}) == 1$) ; (ii) la deuxième liée à l'ordre d'exécution sur la ressource vérifiant que l'opération précédente à o_{jk} sur la ressource r a fini d'être exécutée ($Sequence(o_{jk}) == 1$) ; (iii) la dernière liée à la disponibilité du job j permettant de s'assurer qu'aucune autre opération $o_{jk'}$ du même job j n'est en cours d'exécution ($Avail(j) == 1$). Cette dernière contrainte permet notamment de s'assurer qu'il n'y a pas de chevauchement entre deux ou plusieurs opérations du même job.

Lorsque cette garde est satisfaite, l'opération o_{jk} est synchronisée avec la ressource qui lui est

	Notation	Nature	Signification
Globale	*clk*	horloge	Horloge globale permettant le calcul de C_{max}
	operation	entier	variable partagée représentant l'opération courante
	$Req(r)$	événement urgent	Évènement de la requête d'exécution r
	$Comp(o_{jk})$	évènement non-urgent	Évènement de fin d'exécution de l'opération o_{jk}
Opération	NbJ	entier	Nombre de Jobs
	$NbOp$	entier	Nombre total d'opérations dans l'atelier de production
	o_{jk}	entier	Identifiant de l'opération
	d_{jkr}	entier	Durée d'exécution de o_{jk} sur r
	r	entier	La ressource qui exécute l'opération o_{jk}
	$OpComp$	booléen	État d'exécution de l'opération
	$Avail(j)$	booléen	État de disponibilité du job j
	$Route(o_{jk})$	entier	Précédence de l'opération dans la gamme du job
	$Sequence(o_{jk})$	entier	Précédence de l'opération sur la séquence de la ressource
Ressource	NbR	entier	Nombre de ressources
	r	entier	Identifiant de la ressource
	o_{jkr}	entier	L'opération courante sur la ressource r
	x_r	horloge	Horloge locale permettant le calcul de durée d'exécution de l'opération d_{jk}

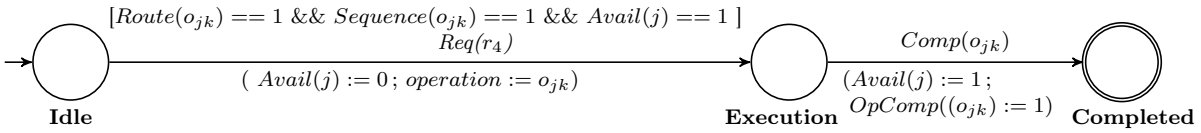
TABLE 3.2 – Données des patrons d'ordonnancement

allouée r en utilisant l'événement urgent (Req(r)). La disponibilité du job est alors mise à jour ($Avail(j) := 0$) et l'identifiant de l'opération en cours d'exécution est alors sauvegardé dans la variable *operation* (*operation* := o_{jk}). Dans la localité **Execution**, l'opération o_{jk} est en attente de la fin de l'exécution. L'événement synchronisant (Comp(o_{jk})) permet alors à l'opération de franchir la transition et d'aller vers la localité marquée **Completed** en mettant à jour l'état d'exécution de l'opération (*OpComp :=1*) et la disponibilité du job ($Avail(j) := 1$).

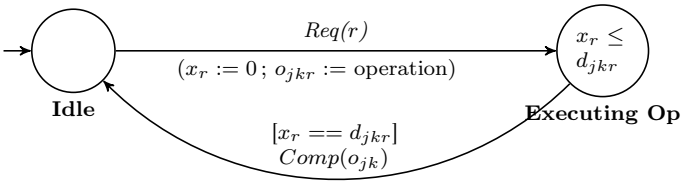
Ce patron partage les informations en utilisant les événements de synchronisation (*Req, Comp*) et la variable (*operation*). L'événement *Req(r)* est déclaré comme étant urgent pour permettre l'exécution au plus tôt de l'opération. En effet les deux patrons de l'opération et de la ressource peuvent consommer du temps dans leurs localités initiales. Pour éviter ce comportement, l'événement de requête *Req(r)* est déclaré comme urgent permettant la synchronisation des deux patrons dès que la garde du patron d'opération est satisfaite . Dans le cas de l'événement $Comp(oj_k)$, les contraintes générées durant la synchronisation entre les deux patrons permettent de ne pas consommer de temps.

3.3.3.2 Patron de ressource

Le patron de ressource (Figure 3.8b) modélise le comportement d'une ressource r. Dans la localité initiale **Idle**, la ressource r est en attente d'une demande d'exécution d'une opération. Quand l'événement urgent (*Req(r)*) est émis, la synchronisation avec le patron d'opération est faite. En franchissant la transition depuis cette localité, l'horloge locale est initialisée à zéro ($x_r := 0$) et l'identité de l'opération est obtenue par la variable partagée *operation* permettant ainsi d'identifier l'opération o_{jk} à exécuter. À partir de la localité **Executing Op**, l'exécution de l'opération est lancée. L'invariant ($x_r \leq d_{jkr}$), défini dans cette localité, garantit que l'exécution de l'opération ne dépasse pas la durée d_{jkr}. Lorsque la valeur de la durée est atteinte, la garde [$x_r == d_{jkr}$] est alors satisfaite permettant la synchronisation avec le patron d'opération. En



(a) Patron d'opération



(b) Patron de ressource

FIGURE 3.8 – Patrons de l'ordonnancement

franchissant la transition, la synchronisation est effectuée à l'aide de l'événement *Comp(o$_{jkr}$)*, permettant ainsi de modéliser la fin de l'exécution de l'opération. À la fin de cette exécution, la ressource retourne à la localité initiale **Idle**.

Dans cette section, nous avons décrit les patrons nécessaires pour modéliser l'exécution déterministe de l'ordonnancement. Dans le cas de la modélisation de l'ordonnancement déterministe, toutes les données et éléments sont considérés comme connus et certains. La durée totale de l'ordonnancement est obtenu par l'utilisation de l'horloge global *clk*. L'instanciation des modèles permet de prendre en considération différentes caractéristiques de l'atelier sans modifier les éléments des modèles. Avant d'introduire les patrons de perturbations modélisant l'impact de ces dernières sur l'ordonnancement, un exemple d'instanciation est donné.

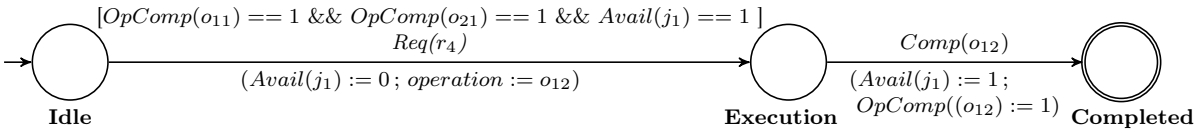
3.3.3.3 Exemple d'illustration des patrons d'ordonnancement

Pour illustrer le principe de modélisation de l'ordonnancement, nous nous basons sur l'ordonnancement généré à partir de l'atelier de l'étude de cas de [Giard, 2003] (Figure 3.3). Les patrons présentés sont instanciés pour l'exécution de l'opération o_{12} sur la ressource r_4. L'opération précédente sur la ressource est o_{21} ce qui implique que la contrainte $Sequence(o_{jk}) == 1$ devient $OpComp(o_{21}) == 1$. Alors que la précédence dans la gamme du job est o_{11} qui est exécutée sur la ressource r_1, la contrainte de précédence $Route(o_{jk}) == 1$ est instanciée à $OpComp(o_{11}) == 1$. La durée d'exécution de o_{12} est $d^{ref}_{12r_4} = 6UT$ sans considérer les perturbations.

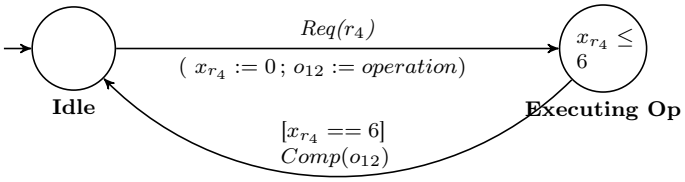
Le patron de l'opération et de la ressource sont instanciés (Figure 3.9) avec les informations disponibles dans l'ordonnancement et dans l'atelier de production. Cette instanciation permet d'obtenir un modèle pour l'opération o_{12} et le modèle de la ressource r_4.

3.3.4 Modélisation des perturbations

En prenant en compte les perturbations, les données de l'atelier ne sont plus considérées comme certaines. En effet, la modélisation des perturbations permet d'introduire les différentes informations incertaines dans l'ordonnancement déterministe. Deux types de perturbations sont modélisées, l'incertitude et l'aléa. L'approche de modélisation proposée permet de définir deux patrons de modélisation distincts, car ils n'ont pas le même comportement. Un aléa caractérise



(a) Modèle de l'opération (o_{12})



(b) Modèle de la ressource r_4

FIGURE 3.9 – Exemple d'instanciation des patrons d'ordonnancement

l'occurrence d'un événement imprévu dans l'atelier de production, alors que l'incertitude est due à un manque d'information dans l'atelier de production impliquant des variations des paramètres de production. Les deux questions sur la modélisation des perturbations sont exprimées comme suit.

— Comment modéliser les deux comportements distincts des perturbations et les intégrer dans l'ordonnancement déterministe ?
— Comment modéliser la représentation stochastique des perturbations ?

Dans cette section, nous proposons de répondre à ces deux questions avec une prise en compte des perturbations comme étant des éléments impactant les durées d'exécution des opérations et donc la durée totale de l'ordonnancement (C_{max}).

3.3.4.1 Prise en compte de l'impact des perturbations

Pour prendre en compte les différentes perturbations, deux hypothèses sont faites :

— les perturbations ont un impact sur la durée d'exécution des opérations, c'est à dire qu'elles génèrent une variation sur la durée d'exécution des opérations et donc sur la durée totale de l'ordonnancement déterministe (C^{ref}_{max}).
— les perturbations sont indépendantes les unes des autres, c'est-à-dire que la prise en compte et l'évolution d'une perturbation n'influence pas l'apparition ou l'évolution d'une autre perturbation.

Pour exprimer ces hypothèses, nous considérons qu'un ensemble de perturbations $Pert$ doit être défini. La durée de l'ordonnancement C^{ref}_{max} peut être impacté soit par des incertitudes u. Une incertitude peut impacter directement la durée d'exécution u^{ex} d'une opération, ou la durée d'occurrence d'aléas u^h. Pour considérer cela, deux sous-ensembles sont définis : U pour les incertitudes et H pour les aléas.
Chaque durée d'exécution d_{jkr} de l'opération o_{jk} est exprimée par la formule (3.1) à partir de la durée d'exécution de référence d^{ref}_{jkr}, obtenue à partir de données déterministes, puis, par une

agrégation des fluctuations liées aux perturbations considérées.

$$d_{jkr} = d^{ref}_{jkr} + \sum_{u^{ex}=1}^{NbU^{ex}} \delta d^{u^{ex}}_{jkr} + \sum_{h=1}^{NbH} H^h_{jk} \times \left(d^{ref,h}_{jkr} + \sum_{u^h=1}^{NbU^h} \delta d^{u^h}_{jkr} \right) \tag{3.1}$$

Pour considérer les aléas, nous faisons l'hypothèse que chaque opération peut être touchée par un ou plusieurs aléas (défaillance des ressources, rupture d'approvisionnement, etc.). Chaque aléa $h \in H$ est lié à :

— une probabilité d'occurrence $p\,(h, o_{jk})$ qui représente la probabilité que cet aléa h ait un impact sur l'opération o_{jk}.

— une variable booléenne H^h_{jk} qui indique si l'opération o_{jk} est impactée par l'aléa h.

— une durée $d^{ref,h}_{jkr}$ qui représente le décalage de référence généré par l'aléa.

Pour considérer le premier type d'incertitudes (u^{ex}), chaque incertitude u^{ex} va générer une fluctuation $\delta d^{u^{ex}}_{jkr}$ (qui peut être positive ou négative) sur la durée d'exécution d_{jkr} ($\sum_{u^{ex}=1}^{NbU^{ex}} \delta d^{u^{ex}}_{jkr}$).

La durée de référence générée par l'occurrence de l'aléa h peut également être affectée par une ou plusieurs incertitudes u^h. Cette incertitude génère une fluctuation ($\delta d^{u^h}_{jkr}$) sur la durée considérée.

Toutes les fluctuations exprimées par une variable δd dans l'équation (3.1) sont des variables aléatoires qui suivent des distributions de probabilités connues supposées être limitées à des intervalles comme $[\delta d^-, \delta d^+]$.

Les paramètres nécessaires pour l'approche de modélisation sont détaillés dans (Table 3.3)

	Notation	Nature	Signification
Patron d'incertitude	δd^-	entier	Variation minimum de la fluctuation
	δd^+	entier	Variation maximum de la fluctuation
	δd	entier	Paramètre de fluctuation
	t	entier	Pas de discrétisation
	l	entier	Paramètres d'itération
	$p(l)$	entier	Poids de probabilité liée à la fluctuation
Patron de l'aléa	H^h_{jk}	booléen	Occurrence de l'aléa
	$p(h, o_{jk})$	entier	Probabilité de l'occurrence de l'aléa
Variables globales	NbU^{ex}	entier	Nombre d'incertitudes sur la durée d'exécution
	NbH	entier	Nombre d'aléas
	NbU^h	entier	Nombre d'incertitudes associée à l'aléa h
	d^{ref}_{jkr}	entier	durée d'exécution de référence de l'opération o_{jk}
	$d^{ref,h}_{jkr}$	entier	Durée de références liée à l'occurrence de l'aléa.
	c	entier	Compteur de perturbation

TABLE 3.3 – Données de perturbations

3.3.4.2 Illustration de l'impact des perturbation

Dans le cas d'illustration du chapitre [Giard, 2003], le décideur veut traiter l'incertitude sur les durées d'exécution u^{ex}_1, la panne ressource h_1 ainsi que l'incertitude sur les durées de réparation $u^{h_1}_1$. En prenant en considération ces trois perturbations, nous pouvons illustrer, non seulement, les deux types de perturbations (Incertitude et Aléa) mais également l'incertitude liée à l'occurrence d'un aléa (Figure 3.10). Les caractéristiques des perturbations sont les suivantes.

— Les seuils de variation des durées d'exécution est $\delta d^-_{jkr} = -25\% d^{ref}_{jkr}$ et $\delta d^+_{jkr} = +25\% d^{ref}_{jkr}$,

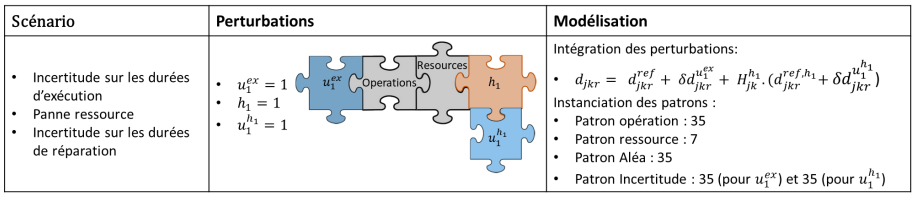


FIGURE 3.10 – Scénario de perturbation de l'étude de cas

— Les seuils de variation des durées de réparation est $\delta d^{-}_{jkr,h_1} = -25\% d^{ref}_{jkr,h_1}$ et $\delta d^{+}_{jkr,h_1} = +25\% d^{ref}_{jkr,h_1}$,
— La durée de réparation est la même pour toutes les ressources $d^{ref,h_1}_{jkr} = 7$,
— Les incertitudes sur la durée d'exécution et de réparation suivent une distribution de probabilité exponentielle,
— La probabilité d'occurrence de la panne ressource pendant l'horizon d'ordonnancement est $p(h_1, o_{jk}) = 3\%$. Cette probabilité indique que pendant cet horizon au moins une ressource va tomber en panne.

À partir de ces informations, l'équation de la durée d'exécution de l'opération o_{12} impactée par les perturbations est donnée par l'équation (3.2).

$$d_{jkr} = d^{ref}_{jkr} + \delta d^{u_1^{ex}}_{jkr} + H^{h_1}_{jk} \times (d^{ref,h_1}_{jkr} + \delta d^{u^{h_1}}_{jkr}) \tag{3.2}$$

Sur la base de la description des perturbations proposée dans l'équation (3.1), nous présentons dans les sous-sections suivantes une approche de modélisation pour considérer ces perturbations. Cette approche est basée sur deux patrons. Le premier modèle permet de représenter le comportement d'un aléa alors que le deuxième modèle permet de représenter le comportement d'une incertitude. Chaque instanciation de ces deux patrons (donc chaque perturbation) et leur évolution sera exécutée indépendamment des autres. Cela est possible car nous avons fait l'hypothèse d'indépendance entre les perturbations. Ces deux patrons sont le résultat d'un travail de généralisation de modèles dédiés proposés dans des travaux initiaux de la thèse et détaillés dans [Himmiche et al., 2018b] (aléa lié à une panne machine) et [Himmiche et al., 2018a] (incertitude sur la durée d'exécution).

Enfin, l'initialisation des perturbations et la procédure pour les intégrer à l'ordonnancement déterministe sont détaillées ensuite.

3.3.4.3 Modélisation de perturbation de type aléa

Pour représenter le comportement de l'aléa, un patron d'automate temporisé stochastique est représentée dans (Figure 3.11). Chaque opération o_{jk} peut être affectée par un aléa $h \in H$ avec une probabilité connue $p(h, o_{jk})$. Ce comportement est modélisée à l'aide de la caractéristique stochastique des ATS, permettant de représenter les choix probabiliste à l'aide d'arcs de probabilités discrètes.

En effet, à partir de la localité **Idle**, deux transitions probabilistes sont possibles. Dans le premier cas, avec la probabilité $p(h, o_{jk})$, le patron évolue vers la localité **Impacted**. En fran-

chissant cette transition probabiliste, la variable de l'aléa H^h_{jk} est mise à jour à 1. Dans ce cas, l'opération o_{jk} est impactée par l'occurrence de l'aléa h. Dans le cas où la transition avec la probabilité $1 - p(h, o_{jk})$ est franchie, le modèle évolue vers la localité **NotImpacted** avec mise à jour de la variable H^h_{jk} à 0 tel que l'aléa h n'affecte pas l'opération o_{jk}. En déclenchant l'une des deux transitions mentionnées, le compteur c est mis à jour ($c := c + 1$) afin de prendre en compte l'exécution du patron de l'aléa.

Dans la localité initiale **Idle**, un invariant permet d'exécuter le patron de l'aléa au plut tôt et sans consommation de temps ($clk \leq 0$). Pour respecter l'approche de modélisation définie, les perturbations sont exécutées en premier, cette exécution ne doit pas consommer de temps pour ne pas impacter la durée totale de l'ordonnancement C_{max}. L'invariant exprime que le patron doit sortir de la localité quand l'horloge globale clk est égale à 0. À partir de ce patron, l'équation (3.1) peut être mise à jour pour l'opération o_{jk}, avec la mise à jour de la variable H^h_{jk}.

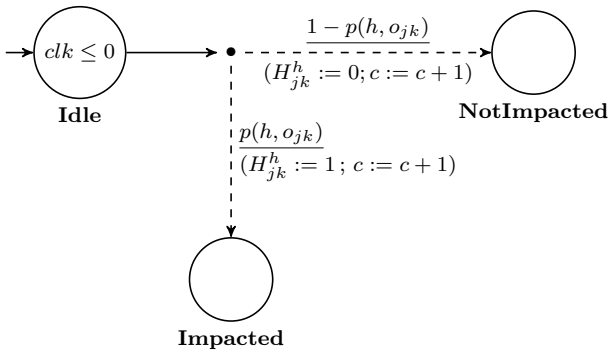


FIGURE 3.11 – Patron de l'aléa

3.3.4.3.1 Exemple d'instanciation du patron de l'aléa

Considérons une perturbation de type aléa h_1 impactant l'opération o_{12} de l'ordonnancement déterministe (Figure 3.3). Cet aléa représente une panne ressource pouvant altérer l'exécution de l'opération. La probabilité d'occurrence de cet aléa est de $p(h_1, o_{12}) = 3\%$. Pour modéliser ces informations, le patron de l'aléa est instancié pour l'opération o_{12} (Figure 3.12).

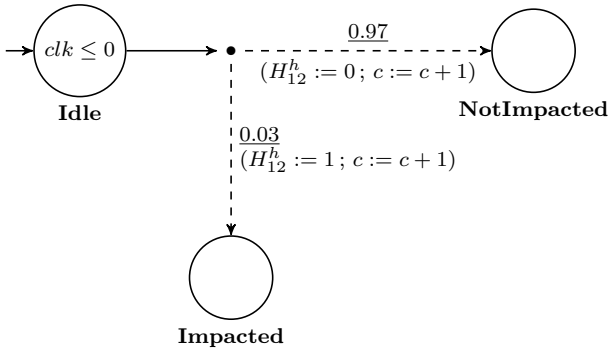


FIGURE 3.12 – Instanciation du patron de l'aléa pour l'opération o_{12}

L'exécution du modèle de l'aléa de l'opération o_{12} (Figure 3.12) permet de déterminer si l'opération est impactée ou non par la panne ressource.

3.3.4.4 Modélisation de perturbation de type incertitude

Les caractéristiques recueillies auprès du décideur permettent de définir les incertitudes comme des variables aléatoires. En effet, chaque paramètre de fluctuation δd est lié à une variable aléatoire continue. Cette variable suit une distribution de probabilité continue supposée connue. Pour modéliser cette information, une discrétisation de la distribution est effectuée de telle sorte que le patron d'incertitude puisse être mis en œuvre sur tout logiciel de vérification formelle d'ATS sans être restreint par la capacité du logiciel à intégrer la loi de probabilité ou non. Cette discrétisation permet ainsi d'être le plus général possible vis-à-vis de la distribution de probabilité d'entrée sans être dépendant du logiciel d'implantation.

Le langage de modélisation choisi (ATS) permet la modélisation des probabilités discrètes prenant ainsi en compte l'approche de modélisation des incertitudes (Figure 3.13). Le patron proposé modélise le comportement des incertitudes que celles-ci soient liées à la durée d'exécution de l'opération u^{ex} ou associée à un aléa u^h. En appliquant les notations données dans l'expression (3.1), nous désignons par δd la fluctuation liée à une incertitude. Le pas de discrétisation de la distribution de probabilité est t tandis que l représente un compteur d'itérations. Le principe de ce patron est d'augmenter, de manière itérative, la valeur de δd dans l'intervalle $[\delta d^-, \delta d^+]$ en considérant la nature de la distribution de probabilité suivie par δd.

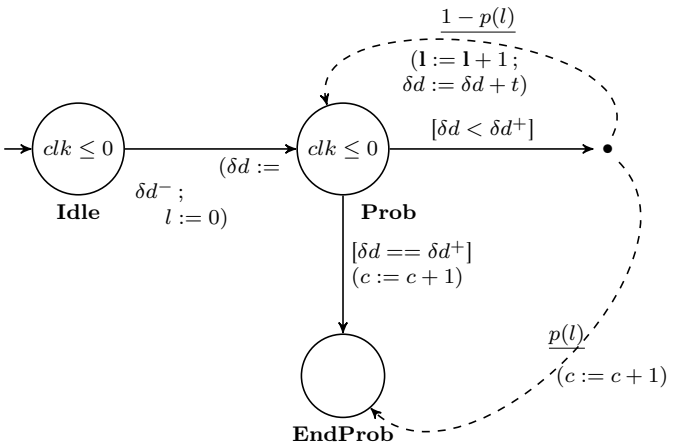


FIGURE 3.13 – Patron d'incertitude

À partir de la localité initiale du patron (Figure 3.13), δd est initialisé à la limite minimale δd^- et le compteur d'itération est initialisé à zéro ($l := 0$). En atteignant la localité **Prob**, deux transitions sont franchissables en fonction de la satisfaction de leurs gardes. La première garde modélise le cas où δd se trouve sous une limite maximale de fluctuation définie δd^+ ($[\delta d < \delta d^+]$). Lorsque cette garde est vraie, δd est mise à jour par $\delta d = \delta d + t$. Cette mise à jour permet d'incrémenter la valeur de δd d'un pas de discrétisation t. Quand cette garde est satisfaite le patron évolue vers deux transitions probabilistes. Avec le poids de probabilité $p(l)$, la localité de

fin d'exécution de l'incertitude **EndProb** est atteinte. La variable δd maintient donc la valeur mise à jour durant l'itération précédente. Sinon, avec la probabilité $1 - p(l)$, l'automate revient à la localité **Prob** en mettant à jour le paramètre d'itération l à $l + 1$. À partir de là, un autre pas de discrétisation est exécuté de telle sorte que la valeur de δd est mise à jour à $\delta d + t$. Si la limite maximale de δd est atteinte ($[\delta d == \delta d^+]$), l'automate évolue directement vers la localité **EndProb** et la valeur de δd est δd^+. Les transitions menant à la localité **EndProb** permettent la mise à jour du compteur de perturbation $c := c+1$ afin de considérer l'exécution de l'incertitude. Les invariants exprimés dans les localités **Idle** et **Prob** permettent l'exécution au plus tôt du patron d'incertitude ($clk \leq 0$). En effet, comme pour le patron de l'aléa, les perturbations de type incertitude doivent être exécutées sans consommation du temps.

Pour considérer les caractéristiques de la distribution de probabilité de δd, le calcul de la valeur de $p(l)$ est nécessaire. En effet, le poids de probabilité $p(l)$ est obtenu à partir de la discrétisation de la distribution de probabilité. Pour ne pas dépendre d'un seul type de distribution, nous introduisons dans ce qui suit, une approche de calcul de $p(l)$.

3.3.4.4.1 Description stochastique des incertitudes

À partir de (Figure 3.13), la probabilité que $\delta_d = \delta d^- + lt$ est la probabilité de boucler dans le modèle $l - 1$ fois et de sortir de la boucle à l'itération l. La probabilité $P_d(\delta d = \delta d^- + lt)$ peut donc être exprimée par l'expression (3.3).

$$P_d(\delta d = \delta d^- + lt) = p(l).\prod_{i=1}^{l-1}(1 - p(i)) \tag{3.3}$$

La variable $p(l)$ est un paramètre probabiliste qui peut être calculé à partir de la distribution suivie par δd.

Pour une variable aléatoire discrète X_d provenant de la discrétisation de X, nous désignons $P_d(X_d = x)$ la probabilité discrète liée à la distribution de probabilité initiale continue. Le pas de discrétisation est désigné par t. Cette probabilité est obtenue par la discrétisation suivante (3.4).

$$P_d(X_d = x) \equiv P(x \leq X < x + t) \tag{3.4}$$

L'approximation de cette discrétisation est considérée connue et acceptable dans le cadre de cette thèse. La variable aléatoire X est liée à une fonction de répartition. La fonction de répartition est mathématiquement définie par : $F_X(x) = P(X \leq x)$.
Dans notre cas, nous considérons qu'une perturbation suit une distribution tronquée dans un intervalle $[x^-, x^+]$. Compte tenu du pas de discrétisation, nous devons considérer que X se trouve dans $[x^-, x^+ + t[$ de sorte que X_d peut prendre toutes les valeurs dans $[x^-, x^+]$. La fonction de répartition d'une distribution tronquée est définie par (3.5).

$$F_T(x) = P(X \leq x | X \in [x^-, x^+ + t]) = \frac{F_X(x) - F_X(x^-)}{F_X(x^+ + t) - F_X(x^-)} \tag{3.5}$$

L'expression finale de la probabilité de $X_d = x$ est alors modifiée pour prendre en considération la caractéristique tronquée de la distribution comme dans (3.6).

$$P_d(X_d = x) \equiv P(x \leq X < x + t) = \frac{F_X(x + t) - F_X(x)}{F_X(x^+ + t) - F_X(x^-)} \tag{3.6}$$

Proposition 1. *Étant données une variable aléatoire δd et sa fonction de répartition connue $F_X(\delta d)$, les valeurs discrètes de $p(l)$ associées à la fluctuation δd (Figure 3.13) sont calculées selon le systèmes d'équations (3.7).*

$$\begin{cases} p(0) = \frac{F_X(\delta d^- + t) - F_X(\delta d^-)}{F_X(\delta d^+ + t) - F_X(\delta d^-)} \\ p(l) = \frac{F_X(\delta d^- + (l+1)t) - F_X(\delta d^- + lt)}{F_X(\delta d^- + lt) - F_X(\delta d^- + (l-1)t)} \times \frac{p(l-1)}{1-p(l-1)} \qquad for\ l \geq 1 \end{cases} \tag{3.7}$$

Démonstration 1. *Les valeurs de $p(l)$ indiquées dans la proposition 1 peuvent être démontrées comme suit :*

1. *Démontrons d'abord la valeur de $p(0)$. $p(0)$ est aussi la probabilité que $\delta d = \delta d^-$: $P_d(\delta d = \delta d^-)$. En utilisant l'équation (3.6), cela conduit exactement à $p(0) = \frac{F_X(\delta d^- + t) - F_X(\delta d^-)}{F_X(\delta d^+ + t) - F_X(\delta d^-)}$.*
2. *La deuxième partie de (3.7) peut être démontrée comme suit. En utilisant l'équation (3.3), pour l nous avons :*

$$\begin{gathered} p(l).\prod_{i=1}^{l-1}(1-p(i)) = P_d(\delta d = \delta d^- + lt) \\ \Rightarrow p(l).(1-p(l-1)).\prod_{i=1}^{l-2}(1-p(i)) = P_d(\delta d = \delta d^- + lt) \\ \Rightarrow p(l).\tfrac{(1-p(l-1))}{p(l-1)}.p(l-1)\prod_{i=1}^{l-2}(1-p(i)) = P_d(\delta d = \delta d^- + lt) \\ \Rightarrow p(l).\tfrac{(1-p(l-1))}{p(l-1)}.P_d(\delta d = \delta d^- + (l-1)t) = P_d(\delta d = \delta d^- + lt) \\ \Rightarrow p(l) = \tfrac{P_d(\delta d = \delta d^- + lt)}{P_d(\delta d = \delta d^- + (l-1)t)} \cdot \tfrac{p(l-1)}{(1-p(l-1))} \end{gathered}$$

En remplaçant par les valeurs de P_d données par l'équation (3.6) dans les derniers résultats d'égalité, nous retrouvons la proposition donnée dans (3.7).

3.3.4.4.2 Exemple d'instanciation du patron d'incertitude

L'opération o_{12} est impactée par l'incertitude sur la durée d'exécution u_1^{ex}. L'incertitude est représentée par la fluctuation δd. En effet, la durée d'exécution incertaine est exprimée par $d_{12r_4} = d_{12r_4}^{ref} + \delta d_{12r_4}^{u_1^{ex}}$. La valeur de $\delta d_{12r_4}^{u_1^{ex}}$ sera déterminée par le patron d'incertitude instancié par les paramètres suivants. La durée de référence d'exécution de l'opération est $d_{12r_4} = 6$, le pourcentage de fluctuation est $+/-25\%$, l'intervalle entier de $\delta d_{12r_4}^{u_1^{ex}}$ est donc $[-2, +2]$. L'instanciation du patron d'incertitude (Figure 3.14), va permettre de parcourir l'intervalle de fluctuation jusqu'à trouver la valeur de $\delta d_{12r_4}^{u_1^{ex}}$ à ajouter à la durée d'exécution. Pour ce faire, la valeur de $p(l)$ doit être calculée à chaque nouveau pas d'itération suivant la distribution de probabilités considérée. Dans le cas de cette illustration, la distribution considérée est l'exponentielle. Avec un pas de discrétisation $t = 1$, les valeurs de $p(l)$ sont calculées en appliquant la propriété (1) (Table 3.4).

l	p	δd
0	0.271	-2
1	0.315	-1
2	0.39	0
3	0.541	1
4	1	2

TABLE 3.4 – Illustration du calcul de $p(l)$

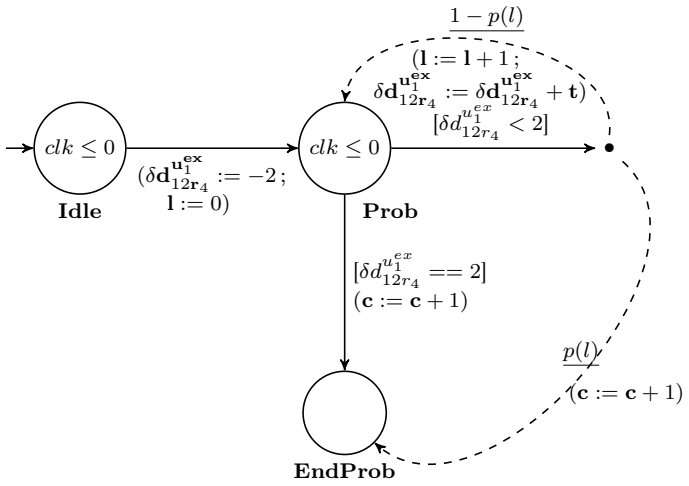


FIGURE 3.14 – Instanciation du patron d'incertitude u_1^{ex} pour o_{12}

3.3.4.5 Intégration et instanciation des perturbations à l'ordonnancement

Après l'introduction des différents patrons modélisant les éléments du problème d'évaluation à savoir l'ordonnancement déterministe et les perturbations, l'interaction entre ces éléments est proposée. Les différents patrons définis permettent de modéliser d'un côté l'ordonnancement déterministe (patrons d'opération (Figure 3.8a) et de ressource (Figure 3.8b)) et de l'autre côté les perturbations (patrons d'aléa (Figure 3.11) et d'incertitude (Figure 3.13)). La structure de modélisation définie permet d'établir l'ordre d'exécution des différents patrons (Figure 3.7). Les patrons de perturbations sont exécutés en premier, afin de connaître l'impact de celles-ci sur la durée d'exécution des opérations. Les patrons d'ordonnancement sont exécutées dans un second temps et la durée d'exécution de l'opération est obtenue à partir de l'équation (3.1). Cette équation va permettre d'établir l'interaction entre l'exécution des perturbations et le patron de l'opération.

L'initialisation des patrons de perturbations permet de récupérer plusieurs informations nécessaires à leur intégration dans les modèles de l'ordonnancement à savoir : (i) la valeur du compteur c qui permet de connaître le nombre de perturbations exécutées, (ii) les différentes fluctuations générées par les perturbations permettant de mettre à jour les durées d'exécution des opérations suivant la formule (3.1). L'interaction entre les deux parties du problème d'évaluation est effectuée par l'ajout d'une transition au patron d'opération (Figure 3.15). La localité initiale ne correspond plus à la localité **Idle** mais à la localité **Waiting**. À partir de cette nouvelle localité, une transition permet de vérifier l'initialisation des perturbations par la garde $[c == NbM]$ afin de s'assurer que le nombre de perturbations initialement défini par NbM a été exécutée. Quand cette garde est satisfaite, la durée d'exécution de l'opération o_{jk} est mise à jour en prenant en considérant l'expression (3.1). Quand cette transition est franchie, le patron d'opération continue son évolution comme expliquée dans (Figure 3.8a).

À partir des données fournies par le décideur au processus d'évaluation (Figure 3.1), le nombre et le type de perturbations sont déduits et les modèles peuvent être instanciés en commençant

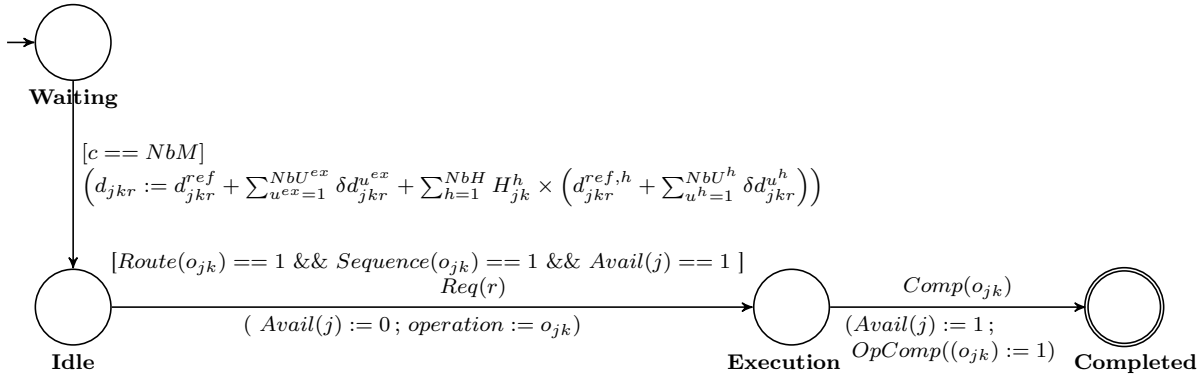


FIGURE 3.15 – Intégration des perturbations dans le patron d'opération

par la définitions des perturbations qui doivent être prises en compte $Pert$ ($Pert = \{U, H\}$). À partir de cet ensemble, les différents types de perturbations sont déduites ainsi que leur nombre pour chaque type de perturbation : NbU^{ex}, NbH et $NbU^H = \sum_{h \in H} NbU^h$.

Avec ces information, les modèles de perturbations sont instanciés. Dans le cas des incertitudes, le paramètre δd et ses bornes δd^- et δd^+ sont instanciés selon le type de durée impactée par l'incertitude (durée d'exécution, durée de réparation, etc.). Les probabilités $p(l)$ du modèle sont également évaluées en fonction des données probabilistes fournies par le décideur et respectant la proposition 1. Dans le cas des aléas, la probabilité $p(h, o_{jk})$ est également évaluée en fonction des informations fournies par le décideur.

Enfin, les modèles d'opération et de ressource et sont instanciés en fonction de les données de l'atelier (nombre de jobs, d'opérations et de ressources). Ensuite, le nombre de modèles de perturbations (Figure. 3.15) est calculé.

Le nombre de patrons à instancier (NbM) dépend du nombre d'opérations dans l'atelier $NbOp$ ainsi que du nombre de chaque type de perturbation. Ce nombre peut être déduit de (3.1) et est exprimé par (3.8) .

$$NbM = NbOp \times (NbU^{ex} + NbH + NbU^H) \tag{3.8}$$

3.3.4.6 Illustration de l'intégration des perturbations

Pour illustrer l'intégration des perturbations à l'ordonnancement déterministe, l'opération o_{12} instanciée dans les sections précédentes est considérée. Si le scénario de perturbations est celui décrit dans (Figure 3.10). Les incertitudes sur la durée d'exécution u_1^{ex} et sur les durées de réparation $u_1^{h_1}$ sont instanciées à l'aide du patron d'incertitude (Figure 3.14) et la panne ressource h_1 est instanciée à l'aide du patron de l'aléa (Figure 3.12). La première étape est l'expression de la durée d'exécution de o_{12} en fonction de ces perturbations (3.9) qui est obtenu à partir de l'équation 3.2. La deuxième étape est l'intégration de cette équation dans le modèle d'opération (Figure 3.9a). Pour l'exemple de l'opération o_{12} alors le nombre de modèle à exécuter est $NbM = 3$.

$$d_{12r_4} = d^{ref}_{12r_4} + \delta d^{u_1^{ex}}_{12r_4} + H^{h_1}_{12} \times (d^{ref,h_1}_{12r_4} + \delta d^{u_1^{h_1}}_{12r_4}) \tag{3.9}$$

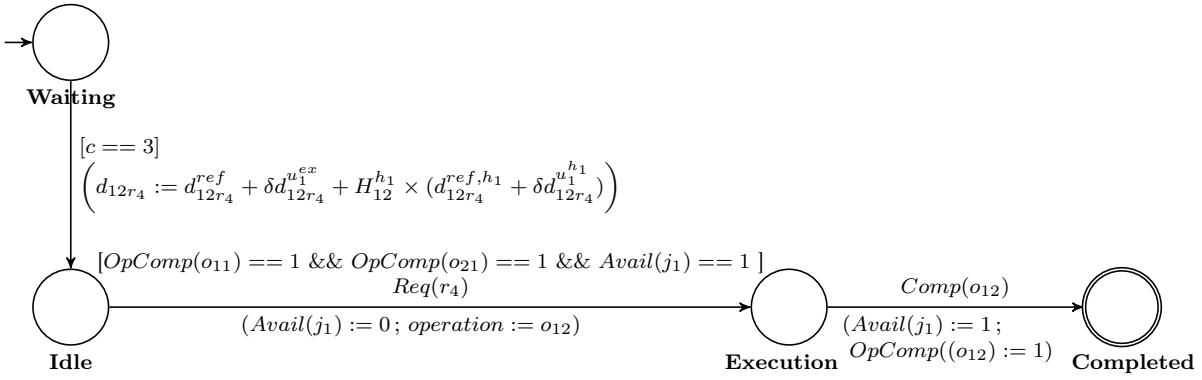


FIGURE 3.16 – Prise en compte des perturbations pour o_{12}

Le patron d'opération intégrant les perturbations est donc instancié en ajoutant la transition avec l'expression d'impact (Figure 3.16). La garde est également instancié en fonction du nombre de modèles de perturbations à exécuter.

3.3.5 Simulation de l'évolution des modèles

La structure de modélisation proposée pour modéliser la problématique d'ordonnancement sous perturbations nous a permis de proposer des patrons de modélisation représentant le comportement de chacun des éléments du problème. Pour assurer la modularité des patrons, nous prenons en compte l'impact des perturbations dans la durée d'exécution des opérations (équation 3.1). Les modèles de perturbations sont les premiers à évoluer durant la simulation de la structure. De plus la définition de ces modèles permet qu'il n'y a pas de consommation de temps pendant leur évolution (l'horloge globale restant à zéro). À la fin de leur évolution, le nombre de perturbations considérées c ainsi que leurs impacts sur la durée d'exécution de chaque opération via les variables associées sont définis. En effet, la valeur de H^h_{jk} est mise à jour dans le patron d'aléa (Figure 3.11) et les valeur de $\delta d^{u^{ex}}_{jkr}$ et $\delta d^{u^h}_{jkr}$ sont mises à jour à la fin de l'évolution des incertitudes (u^{ex} ou u^h) (Figure 3.13).

Quand les modèles de perturbations évoluent, les modèles d'ordonnancement (opérations et ressources) attendent dans leurs localités initiales. Quand le nombre de perturbations considérées est atteint ($c = NbM$) alors le modèle d'opération évolue à son tour après avoir mis à jour la durée d'exécution de l'opération. Ensuite la séquence de l'ordonnancement étant prédéfini par l'ordonnancement d'entrée s, la ressource qui doit exécuter l'opération est prédéfinie. Cette ressource exécute l'opération en faisant évoluer l'horloge de la ressource jusqu'à atteindre d_{jkr}. Quand la valeur de d_{jkr} est atteinte le modèle de ressource r est synchronisé avec l'opération par le biais de l'événement $Comp(o_{jk})$ qui permet de mettre à jour l'état de l'exécution de l'opération et de libérer la ressource qui retourne à sa localité initiale (Figure 3.17).

3.3.5.1 Graphe d'états de la structure de modélisation

Pour visualiser l'évolution synchronisée des patrons de perturbations et d'ordonnancement, nous avons construit les graphes d'états qui permettent de voir les évolutions possibles de la structure quand le patron de l'aléa est pris en compte (Figure 3.18) et les évolutions possibles

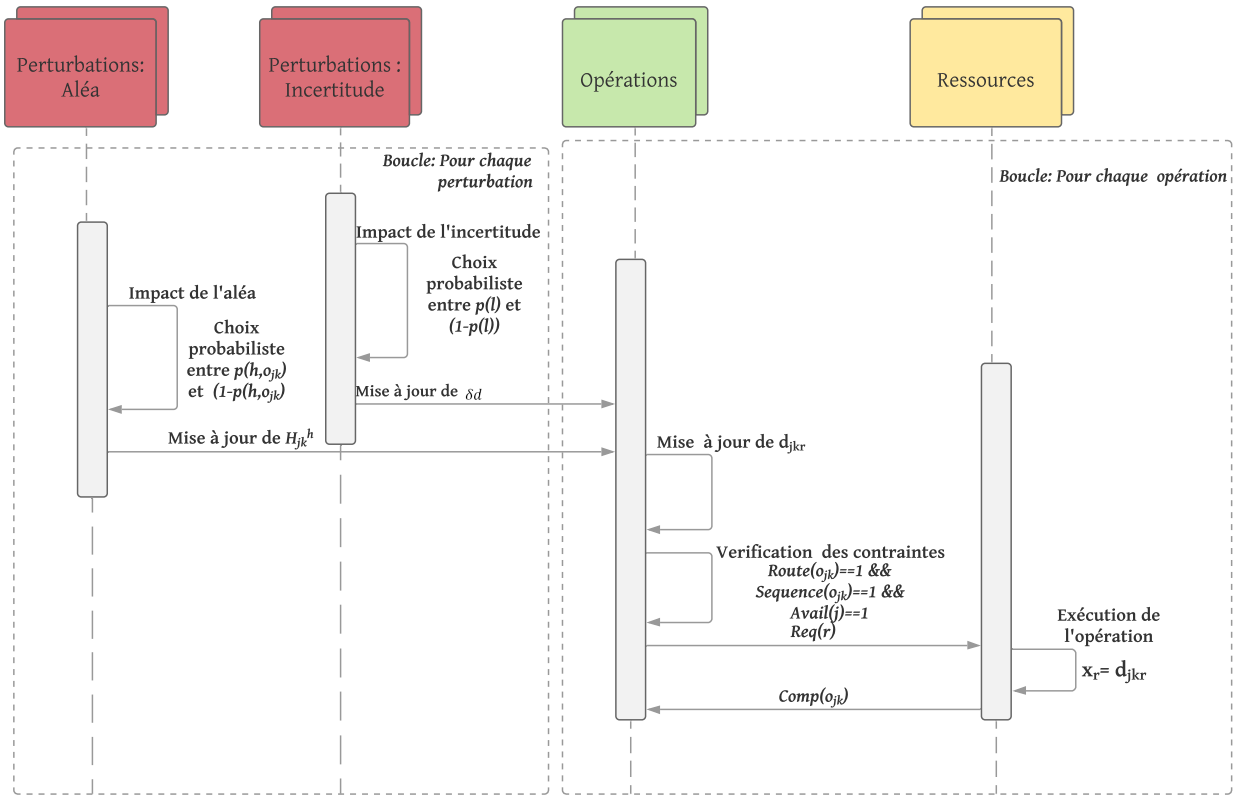


FIGURE 3.17 – Simulation de l'évolution des modèles proposés

quand le patron d'incertitude est pris en compte (Figure 3.19). Pour que le graphe d'états reste lisible, une seule opération est considérée et l'intervalle d'incertitude permet seulement 3 possibilités. Le graphe d'états résultant de la prise en compte des deux patrons incertitude et aléa est finalement le combinatoire des deux graphes présentés.

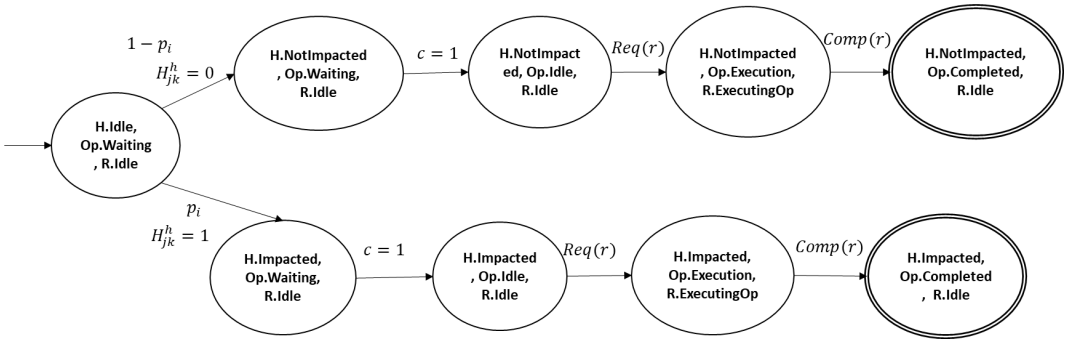


FIGURE 3.18 – Graphe d'états des patrons : Aléa(H), Opération(Op), Ressource(R)

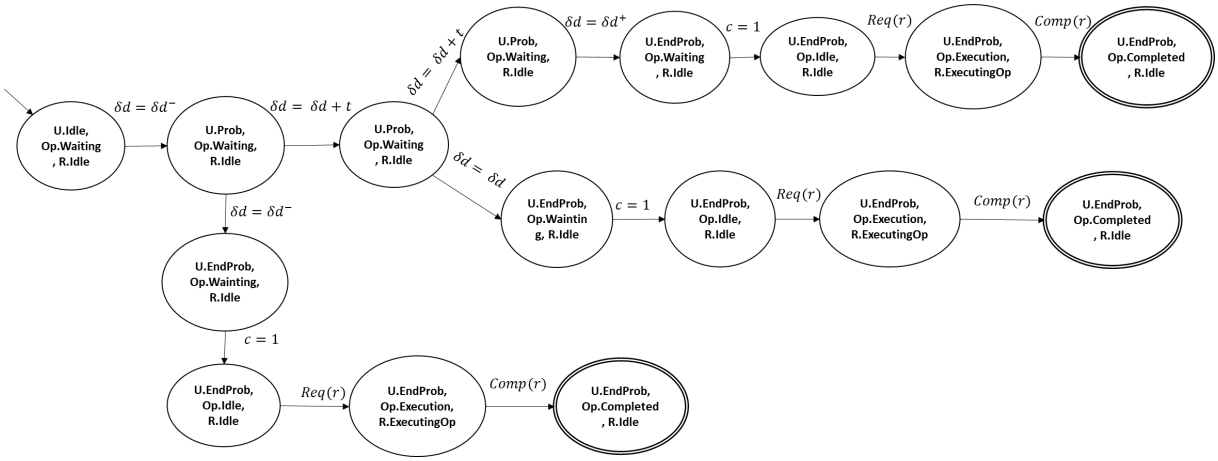


FIGURE 3.19 – Graphe d'états patrons : Incertitude(U), Opération(Op), Ressource(R)

3.4 Évaluation du niveau de service

Afin d'évaluer le niveau de service sur les modèles précédemment proposés, la spécification du niveau de service doit être modélisée et une approche d'évaluation doit être proposée.

Comme défini dans le chapitre 2, le niveau de service RL d'un ordonnancement déterministe s soumis à des perturbations $Pert$ est la probabilité que sa durée totale C_{max} respecte la deadline définie $\tilde{d}$: $RL(s, Pert, \tilde{d}) = Pr(C_{max}(s, Pert) \leq \tilde{d})$.

Les questions auxquelles doit répondre l'étape d'évaluation sont les suivantes :
— Comment traduire la spécification de la robustesse dans un langage reconnu par les ATS ?
— Quelle approche adopter pour calculer RL ?
Pour atteindre l'objectif d'évaluation, il est donc impératif de formuler la spécification de la robustesse pour que celle-ci soit vérifiée sur les patrons d'automates ATS définis dans l'étape de modélisation. Il est également nécessaire de déterminer la méthode de vérification choisie pour exécuter la spécification. À la fin de l'étape d'évaluation, la question de l'outil d'implémentation permet de répondre à la faisabilité du processus d'évaluation permettant son utilisation.

3.4.1 Modélisation de la propriété d'analyse du niveau de service

La modélisation doit permettre de décrire non seulement le comportement du système mais également les différentes propriétés que le système doit satisfaire. La logique temporelle [Aziz et al., 1995] permet de prendre en compte les propriétés réelles des systèmes dont la propriété du temps.

La logique temporelle est une extension de la logique booléenne ayant comme opérateurs (et : $\wedge$, ou : $\vee$, non : $\neg$). Cette logique est enrichie avec des opérateurs temporels [Michel, 1984]. La logique temporelle permet de vérifier si à un moment donné, la propriété du système est satisfaite ou non. Cette logique permet d'exploiter le séquencement des événements qui peuvent se produire dans un système. Elle permet donc d'exprimer la propriété : *À chaque fois que l'événement a est observé, l'événement b est observé auparavant.* La logique temporelle ne manipule jamais le temps explicitement, elle ne permet donc pas d'exprimer la notion du temps que ce

soit en secondes, heures ou jours. Pour quantifier l'évolution des événements, les observations sont utilisée. Par exemple, au lieu de dire que dans 3 secondes j'observe l'événement a, la logique temporelle permet d'exprimer que dans 3 observations l'événement a va se produire.

Pour représenter les propriétés temporelles d'un système, trois types de logique sont possibles LTL (*Linear Time Logic*), CTL^* et CTL (*Computation Tree Logic*).

Toutes les logiques présentées sont liées entre elles (Figure 3.20). En effet, la logique temporelle CTL^* est connue pour être la plus expressive en termes de propriétés à vérifier. Cette capacité rend cette logique très coûteuse en termes d'implémentation. Les logiques LTL et CTL sont donc incluse dans CTL^* et permettent une implémentation moins coûteuse. La logique $PCTL$ est l'extension probabiliste de la logique CTL.

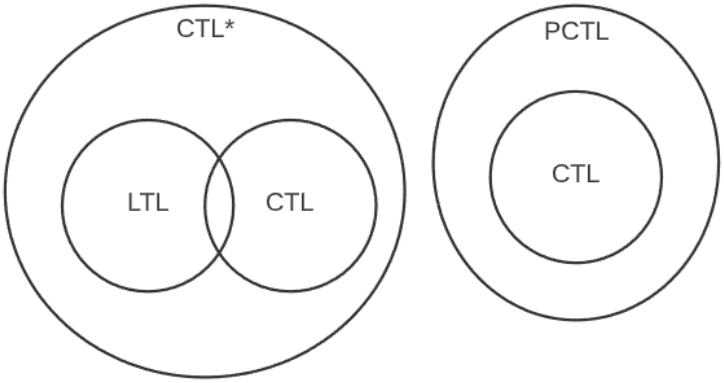


FIGURE 3.20 – Hiérarchie des logiques temporelles

3.4.1.1 Logique CTL

Dans le cadre de cette thèse, la logique CTL est retenue car elle permet de vérifier deux types de propriétés sur l'ensemble des chemins possibles d'évolution :

- Propriété d'états : permet de répondre à la question : *Dans quel état s, la propriété est Vrai ?*
- Propriété de chemins : permet de répondre à la question : *Quel est la validité d'une partie ou de tous les chemins à partir d'un état donné ?*. Un chemin est défini par une séquence finie ou infinie d'états.

3.4.1.1.1 Quantificateurs de chemin

Les propriétés de chemin permettent d'exprimer l'observabilité d'un événement ou d'une séquence au long d'une exécution. Cette exécution permet d'exprimer le chemin. Plus précisément, un chemin $\omega = s_0 s_1 s_2 \dots s_n \dots$ est une séquence d'états s du système [Bardin, 2008].
Les opérateurs temporels liés à un chemin sont **X** (next,suivant), **F** (future, dans le futur), **G**(globally, globallement) et **U** (until, jusqu'à). Pour une propriété atomique a associée à des états observés du système, les opérateurs temporels permettent d'exprimer les propriétés suivantes.

— Opérateur **X** : **X***a* signifie que *a* est vrai dans l'état suivant le long du chemin (Figure 3.21)

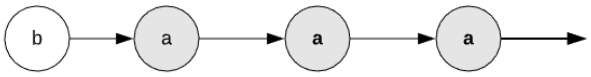


FIGURE 3.21 – Opérateur **X**

— Opérateur **F** : **F***a* signifie que *a* est vrai dans au moins un état du chemin (Figure 3.22). Cet opérateur permet d'exprimer l'accessibilité d'un état précis.

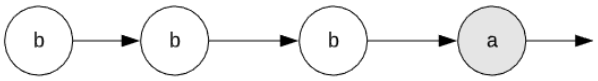


FIGURE 3.22 – Opérateur **F**

— Opérateur **G** : **G***a* signifie que *a* est vrai dans tout les états du chemin (Figure 3.23). L'opérateur **G** permet d'exprimer l'invariance des états du système. En le combinant avec l'opérateur **F**, il permet d'exprimer la vivacité tout au long d'un chemin (**G**($a \rightarrow$**F** b))

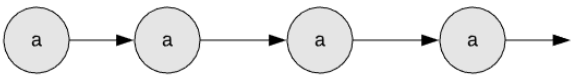


FIGURE 3.23 – Opérateur **G**

— Opérateur **U** : a**U**b signifie que *a* est vrai jusqu'à ce que *b* devient vrai dans un état du chemin (Figure 3.23)

3.4.1.1.2 Opérateurs temporels

Les opérateurs présentés permettent d'exprimer une propriété sur le long d'un chemin donnée offrant une vision linéaire du temps en le considérant fixe et dans le futur. En réalité, le temps peut être représenté comme une structure arborescente où à chaque état, plusieurs chemin sont possibles pour atteindre un état. Cette nouvelle vision permet donc d'exprimer des propriétés d'états [Bardin, 2008].

Deux opérateurs permettent de décrire les propriétés d'états, ils sont nommée quantificateurs de chemins :

— Le quantificateur **A** : **A**ϕ permet d'exprimer que tous les chemins partant d'un état connu satisfont la propriété ϕ,
— Le quantificateur **E** : **E**ϕ permet d'exprimer qu'il existe, au moins, un chemin partant de l'état courant qui satisfait la propriété ϕ.

En associant les quantificateurs de chemins et les opérateurs de chemins (**X,F,G,U**), plusieurs expressions sont possibles, parmi elles, nous retrouvons :

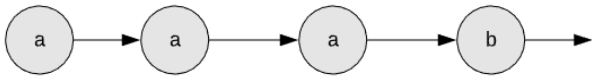


FIGURE 3.24 – Opérateur **U**

— Expression *"éventuellement"* **EF***a* : À partir de l'état courant s, il existe un chemin partant de s qui permet d'atteindre a (Figure 3.25).

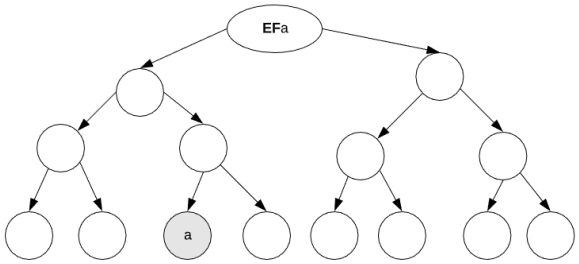


FIGURE 3.25 – Expression *EF*

— Expression *"toujours"* **AF***a* : À partir de l'état courant s, tous les chemins partant de s finissent par atteindre a (Figure 3.26).

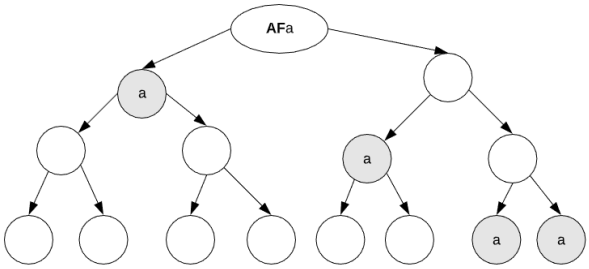


FIGURE 3.26 – Expression *AF*

Exemple 7. *Pour exprimer la situation qu'une ressource va tomber en panne un jour. En CTL, la propriété est exprimée par* ***EF****panne. Cette propriété se traduit par, il existe (au moins) un chemin, où l'état de la ressource sera panne. Pour exprimer la situation qu'une ressource est toujours disponible en CTL la propriété est* ***AG***¬*panne. Cette propriété se traduit par, pour tout les chemins et dans tous les états la ressource n'est pas en panne.*

3.4.1.2 Logique PCTL

La logique $PCTL$ est une extension de la logique CTL, c'est-à-dire qu'elle est basée sur la même typologie de propriétés et qu'elle permet l'utilisation de tout les opérateurs définis pour la logique CTL. La logique $PCTL$ permet d'enrichir les propriétés vérifiées avec des opérateurs probabilistes. Le langage $PCTL$ est souvent utilisée pour la vérification des chaînes de Markov à temps discret ($DTMC$). L'opérateur probabiliste considéré en $PCTL$ est P s'intégrant dans

les opérateurs pouvant être utilisé dans des formules d'états. La propriété probabiliste définie par cette logique est $P_{\ p}[\Psi]$, cette propriété exprime le fait que le chemin Ψ est vrai avec une probabilité $\sim p$, le symbole exprime les opérateurs mathématique de comparaison ($<, >, \leq, \geq$). Dans le cas de cette expression Ψ est exprimée en CTL. En effet, la propriété $PCTL$ cherche à calculer la probabilité du chemin permettant d'atteindre un état qui satisfait une propriété a.

Pour pouvoir exécuter cette propriété, toutes les probabilités du chemin ψ sont cumulées pour obtenir une probabilité globale du chemin à partir d'un état s ($Prob(s, \psi)$) ensuite cette probabilité est comparée avec la valeur de p ($Prob(s, \psi) \sim p$?). La vérification de cette propriété implique donc que tous les chemins exprimée en $PCTL$ soient mesurables.

À partir de cette hypothèse, [Kwiatkowska et al., 2007] ont introduit un nouvelle propriété en $PCTL$ permettant le calcul de la probabilité d'un chemin ($P_{=?}[\Psi]$), avec Ψ un chemin exprimée en CTL. La question exprimé au début de cette section peut donc être formulée par la propriété 3.10). Cette propriété permet d'exprimer la question : *quelle est la probabilité $P_{=?}$ que le système arrive éventuellement dans l'état s ?*. Dans le cas de la propriété (3.10), l'opérateur P permet de calculer la probabilité que la la propriété $Q = EFs$ soit satisfaite.

$$P_{=?}[EFs] \tag{3.10}$$

3.4.1.3 Modélisation de la spécification du niveau de service en $PCTL$

Pour formuler la propriété, considérons un état globale A signifiant que tous les modèles d'opérations de l'atelier sont dans leur localité marquée **Completed**. La question formulé est donc : *Quelle est la probabilité qu'il existe un chemin arrivant un jour dans l'état s avec une durée inférieure ou égale à deadline $\tilde{d}$?*

En utilisant la logique PCTL, cette question est exprimée par la propriété (3.11).

$$P =?[EF \text{ “}All\ operations\ o_{jk}\ are\ Completed\ before\ \tilde{d}\text{”}] \tag{3.11}$$

En réalité, le Makespan $C_{max}(s, Pert)$ est calculé par la valeur de l'horloge globale (clk). L'état $s =$ “*All operations o_{jk} are Completed before $\tilde{d}$*” est une expression PCTL pour : $(C_{max}(s, Pert) \leq \tilde{d})$.

Suite à la modélisation de l'atelier et de la propriété d'analyse du niveau de service, il est maintenant possible d'évaluer un niveau de service pour un atelier donné en utilisant le model-checking probabiliste.

3.4.2 Model-checking probabiliste

Pour rappel, un Model-cheker vérifie s'il existe un état s du modèle satisfaisant la propriété $M, s \models \Phi$? (Figure 3.27). Le processus de Model-Checking est souvent automatisé permettant une rapidité dans l'exécution de la vérification. À l'issue de la vérification, un Model-Cheker permet d'obtenir un contrexemple dans le cas où la propriété n'est pas satisfaite. Ces caractéristiques font du Model-cheking un bon candidat pour vérifier les systèmes complexes en prenant en compte leur propriétés temporelles.

Comme nous l'avons montré dans le chapitre 1, le Model-Checking stochastique est le plus adapté pour ce type de vérification. En effet, le Model-Cheker doit permettre de parcourir l'arbre

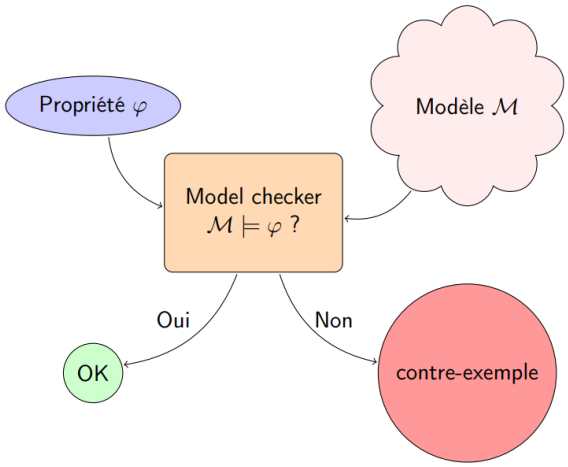


FIGURE 3.27 – Principe du Model Cheking

de transitions probabilistes et de calculer les probabilités liées aux chemins [Barbot, 2014]. Plusieurs méthodes et outils existent permettent le Model-Checking probabiliste Dans la littérature, les méthodes de Model-Checking probabiliste sont catégorisées en deux types : la vérification de modèle numérique (NMC) et la vérification de modèle statistique (SMC) [Barbot, 2014].

Les deux méthodes de Model-Cheking offrent des avantages et inconvénients (Table 3.5).

— Le Model Checking statistique consomme moins de mémoire. En effet, la simulation de chemins est nécessite une faible utilisation de la mémoire. Alors que pour le NMC, la mémoire pose souvent un problème de saturation à cause de l'analyse numérique effectuée.

— Du moment que la sémantique de la distribution de probabilité est disponible, alors le SMC peut être utilisé pour analyser tout système utilisant cette distribution.

— Le parallélisme est possible en utilisant le SMC, comme toutes les simulations sont indépendantes, ce qui n'est pas le cas pour le NMC.

— Le temps de vérification de la propriété est plus faible en utilisant le SMC. En effet, la génération de chemins ne nécessite pas la construction de toute l'arborescence d'évolution. Dans le cas du NMC, le calcul de probabilité nécessite la connaissance de tout l'arbre d'évolution afin de construire les matrices de probabilités.

NMC	*SMC*
- Saturation de mémoire	+ Faible besoin en mémoire
- Explosion combinatoire	+ Pas d'explosion combinatoire
+ Précision des résultats	- Approximation des résultats

TABLE 3.5 – Model-Checking : NMC Versus SMC

En conclusion, les deux types de Model-Cheking permettent d'atteindre des objectifs différents. En effet, si le système à traiter n'est pas volumineux et que la précision des résultats est nécessaire , alors le Model-Checking numérique est un bon candidat. Dans le cas où le système à vérifier est complexe et que l'approximation des résultats est acceptable, alors le Model-Checking statistique reste la meilleur méthode .

Dans le cas de l'étape d'évaluation, le système à vérifier est un système de production conte-

nant plusieurs éléments et ayant une taille réelle. L'utilisation du NMC n'est donc pas possible. Dans notre cas, la valeur de probabilité du niveau de robustesse n'est pas contrainte par la précision. Il suffit donc de trouver un degré de confiance optimal pour obtenir des résultats fiables. Le Model-Checking statistique sera donc privilégié.

Le Model-Chekling statistique génère différents chemins d'exécution et vérifie, après chaque exécution, la satisfaction d'une propriété pour donner les résultats statistiques associés. Cela permet d'éviter l'explosion combinatoire et est ensuite adapté à la vérification de systèmes réels [Ballarini et al., 2011].

Une méthode de résolution du SMC est basée sur la simulation. En effet,, une variable aléatoire X est définie, cette variable prend la valeur de 1 sur un chemin qui satisfait la propriété définie et 0 sinon. Ainsi cette variable suit une loi de Bernouilli, permettant ainsi de calculer son espérance $E(X)$.

Plus précisément, l'algorithme par simulation de Monte Carlo permet de simuler un nombre de chemins, noté N. Sur ce nombre de chemins simulés, l'algorithme permet de compter les chemins satisfaisant la propriété. Par la suite, une estimation de la probabilité est obtenue par le ratio du nombre de chemins à succès (i.e. satisfaisant la propriété définie) sur le nombre total de chemins simulés.

La question à laquelle doit faire face le SMC basée sur la simulation Monte Carlo est quel : *Quel est le nombre N de simulations nécessaires pour obtenir une estimation de probabilité correcte ?*

Pour calculer le nombre, [Nimal, 2010] a comparé trois méthodes. La première basée sur le calcul de l'intervalle de confiance, la deuxième basée sur le calcul de l'intervalle de confiance asymptotique et la dernière basée sur le Model-Checking approximatif.
La méthode par intervalle de confiance (CI) consiste à déterminer les limites de l'intervalle d'estimation de la probabilité avec une précision spécifiée ϵ et un niveau de confiance α tel que, la probabilité RL_i appartient à l'intervalle $[RL_i - \epsilon; RL_i + \epsilon]$ avec une probabilité de $(1 - \alpha)$.

La méthode d'intervalle de confiance asymptotique (ACI) est basée sur le même principe de calcul d'intervalle de probabilité, la seule différence est dans la méthode de calcul de ce dernier. Pour les deux méthodes, le nombre de chemins simulé N dépend de la largeur de l'intervalle de confiance, il n'est donc pas défini à l'avance.

La méthode de Model-Checking approximatif (AMC), permet de fixer un nombre de simulation souhaité N. À partir du N défini, l'intervalle de probabilité est calculé permettant de décider de diminuer le temps de vérification en diminuant N et de l'augmenter si N augmente.

Le comparatif de ces trois méthodes permet de définir la méthode par intervalle de confiance CI comme étant la méthode optimale pour le calcul de l'intervalle de probabilité. En effet, la méthode ACI, ne permet qu'une approximation de l'intervalle de confiance, elle est donc moins fiable que CI. Alors que pour AMC, le fait de devoir fixer le nombre de simulations à faire mène souvent à une surestimation de N. Cette surestimation va nécessiter un temps de résolution plus long. Dans le cas contraire, la sous-estimation de N engendre un intervalle de probabilité non fiable [Nimal, 2010].

Dans la section suivante, les modèles ATS et la propriété de robustesse sont implémentés afin d'exécuter le processus d'évaluation pour un évaluer un ordonnancement déterministe. La méthode de modélisation et d'évaluation ont été réalisées de façon à ce que le décideur n'ait pas à modéliser le problème d'évaluation pour chaque changement de paramètre. En effet, pour le décideur, le processus d'évaluation est une boîte noire ayant comme entrées les paramètres du problèmes donnés par le décideur et pour sortie le niveau de service calculé.

3.5 Implémentation du processus d'évaluation

Dans ce chapitre, nous avons présenté les deux étapes de réalisation du processus d'évaluation de la robustesse. Pour utiliser le processus d'évaluation, les deux étapes de réalisation sont implémentées dans un outil adapté. L'outil choisi doit permettre la modélisation des automates temporisés stochastiques (ATS) pour pouvoir implémenter les patrons de l'étape de modélisation. L'outil doit également permettre le Model-Checking statistique afin de traiter la propriété représentée en $PCTL$. Nous avons identifié deux outils ayant ces caractéristiques : PRISM [Kwiatkowska et al., 2002] et UppAal SMC [David et al., 2015].

Les deux outils permettent l'implémentation des modèles du langage ATS ainsi que le Model-Cheking statistique. Deux points de divergences sont néanmoins constatés. D'un point de vue modélisation, l'outil PRISM permet une représentation textuelle des modèles ATS, ce qui peut générer quelques problèmes de modélisation. L'outil UppAal SMC est basé sur une représentation graphique des modèles permettant une implémentation plus rapide. D'un point de vue Model-Checking, les deux outils intègrent la méthode SMC pour la vérification des propriétés exprimées en $PCTL$. Néanmoins, PRISM utilise la méthode ACM pour le calcul de l'intervalle de confiance, alors que UppAal SMC utilise la méthode CI pour le calcul de l'intervalle de confiance.

Dans ce cas, nous avons préféré l'outil UppAal SMC. Ce dernier est celui qui correspond le mieux pour implémenter les modèles et de la propriété de robustesse pour permettre la mesure de l'intervalle de confiance du niveau de service.

3.5.1 Implémentation du processus d'évaluation dans UppAal SMC

UppAal est une boîte à outils pour la vérification des systèmes en temps réel représentés par un réseau d'automates temporisés enrichis par des variables entières, des données structurées et la synchronisation. La première version de cet outil a été lancée en 1995, il est depuis en développement constant par l'Université d'Uppsala et l'Université d'Aalborg. UppAal est un outil utilisé avec succès dans des études de cas allant des protocoles de communication aux applications multimédias.

UppAal SMC vient répondre aux besoins de représentation des systèmes complexes ayant un comportement dynamique et prenant en compte les caractéristiques stochastiques des systèmes (Annexe B). Cette version de l'outil prend en compte la modélisation à l'aide des automates temporisés stochastiques. Pour vérifier les modèles ATS, UppAal SMC intègre le Model-Checking statistique pour permettre l'évaluation de propriétés exprimées dans la logique $PCTL$.

3.5.1.1 Implémentation des patrons du processus

Pour implémenter les patrons définis du processus d'évaluation, nous commençons par la déclaration des éléments nécessaires de nos patrons. En effet, dans l'onglet déclaration, toutes les variables sont déclarées. Non seulement les variables présentées dans les tables 3.2 et 3.3 sont présentées dans cette partie mais également les paramètres de l'ordonnancement s_i qui vont permettre l'instanciation des patrons. L'ordonnancement s_i est représenté par des tables contenant les informations de ce dernier comme par exemple dans déclaration (Figure B.2) la table représente les jobs existant dans l'atelier de production et permet de déclarer qu'ils sont initialement disponibles. Les tables permettent également la représentation des variables liées aux perturbations. En effet, pour chaque perturbation considérée (aléa ou incertitude), la table permet de définir les valeurs des paramètres. Pour chaque perturbation, une table de la taille du nombre d'opérations permet de définir pour chaque opération la valeur que la variable va initialement prendre. Par exemple, pour une perturbation de type aléa, les variables pour chaque opération sont : la probabilité d'occurrence de l'aléa pour cette opération ($p(h, o_{jk})$) et le booléen de l'impact de l'aléa sur l'opération (H^h_{jk}).

Par la suite, chaque patron défini dans la phase de modélisation du processus d'évaluation est intégré dans l'outil UppAal SMC. Cet outil permet une représentation graphique proche de celle utilisé pour décrire les patrons *ATS*. Tout d'abord, les patrons d'ordonnancement (opération et ressource) sont implémentés à l'aide de deux templates de l'outil (Figure 3.28), ensuite les patrons de perturbations (aléa et incertitude) sont introduits à leur tour dans deux autres templates (Figure 3.29). Ce type de modélisation facilite l'ajout ou le retrait d'une perturbation. En effet, pour l'ajout d'une perturbation, le patron associé au type de cette dernière est dupliqué à l'aide de l'outil. La perturbation peut donc être déclarée dans l'onglet de *Déclaration*. Si la perturbation n'est plus prise en compte, il suffit de supprimer le template et les paramètres associés ou bien tout simplement enlever le template de la perturbation dans la *Composition du système*. La deuxième méthode permet d'un côté de ne pas modifier la déclaration globale du problème et de l'autre côté, elle permet d'évaluer différentes perturbations sans pour autant modifier les templates et la déclaration.

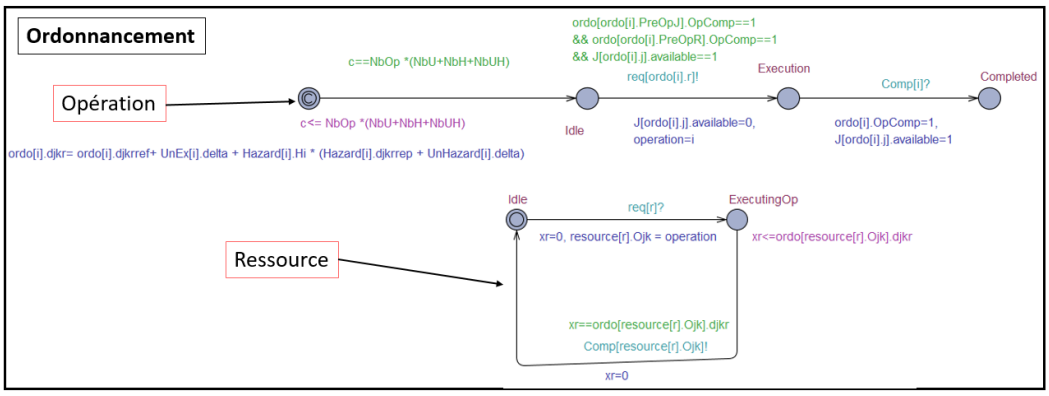


FIGURE 3.28 – Implémentation des patrons d'ordonnancements dans UppAal SMC

A travers la fenêtre de simulation les patrons sont instanciés (Figure 3.30). Cette fenêtre permet voir tout les modèles du problème considéré.

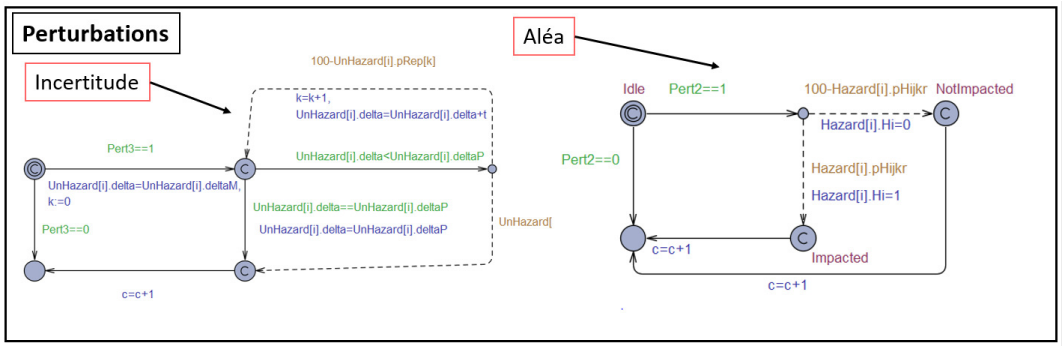


FIGURE 3.29 – Implémentation des patrons de perturbation dans UppAal SMC

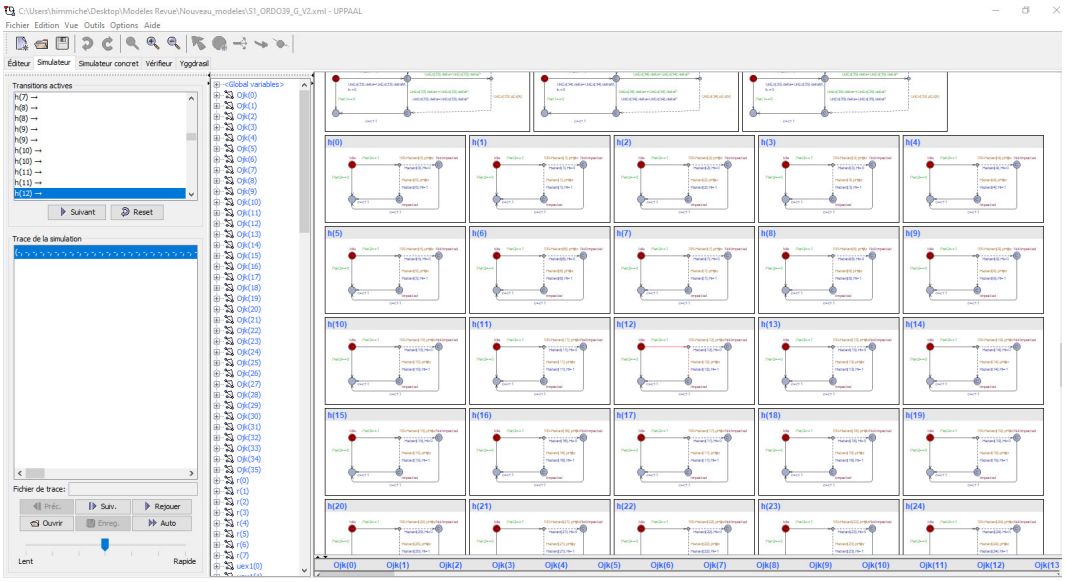


FIGURE 3.30 – Instanciation des patrons sur UppAal SMC

3.5.1.2 Implémentation de la propriété d'évaluation

L'outil UppAal SMC nous permet d'implémenter la propriété d'évaluation définie en $PCTL$ (Expression 3.11). Dans l'onglet *Vérifieur*, la propriété est retranscrite (Expression 3.12).

$$Pr[\leq \tilde{d}](<> forall(i : int[1, NbOp - 1])Operation(i).Completed) \tag{3.12}$$

En effet, $Pr[\leq \tilde{d}]$ est équivalent à l'opérateur $Pr =?[]$. Cette expression permet de prendre en compte la deadline au début de l'expression. La deuxième partie de l'expression $(<> forall(i : int[1, NbOp - 1])Operation(i).Completed)$ permet d'exprimer que pour tout les modèles d'opérations instanciés la localité marquée **Completed** est atteinte. Cette deuxième partie correspond à l'état s exprimée dans (Expression 3.11).

Pour vérifier la propriété, UppAal SMC permet de régler des paramètres statistiques suivants :

— $-\delta$ et $+\delta$: Lors de la vérification d'une hypothèse de la forme $Pr(\Phi) \geq \theta$, l'algorithme effectue les tests de deux hypothèses. Il s'agit de 1) $H_0 : Pr(\Phi) \geq \theta + \delta+$ et 2) H_1 :

$Pr(\Phi) \leq \theta - \delta-$. Ces paramètres définissent la région d'indifférence.

— α et β : α et β sont utilisés pour la vérification des hypothèses. La probabilité d'accepter H_1 au lieu de H_0 est α et inversement pour β. Dans le cas de l'évaluation des probabilités, α est également utilisé et c'est alors la probabilité d'être en dehors de l'intervalle de résultat de la probabilité permettant ainsi de définir les limites de l'intervalle de confiances. Pour la comparaison des probabilités, l'utilisation de α et β est la même que pour les tests d'hypothèses.

— ϵ est l'incertitude pour les évaluations de probabilités. L'outil évalue une certaine probabilité μ et produit le résultat $[\mu - \epsilon, \mu + \epsilon]$. Ce paramètre est donc la précision de la valeur de probabilité calculé par le Model-Checker.

— u_0 et u_1 sont les limites inférieure et supérieure utilisées dans la comparaison des probabilités. Comme pour les tests d'hypothèses, l'algorithme teste deux hypothèses : H_0 : $\frac{Pr(\Phi_1)}{Pr(\Phi_2)} \geq u_1$ et H_1 : $\frac{Pr(\Phi_1)}{Pr(\Phi_2)} \leq u_0$. Ces paramètres définissent la région d'indifférence pour comparer les probabilités.

— Paramètres de l'histogramme : ces paramètres permettent de fixer la largeur de l'histogramme généré dans le cas d'une évaluation de probabilité.

— Traces de résolution : Lors du calcul d'une simulation à l'aide de la requête de simulation, l'outil filtre les données à la volée et retient les points qui se distinguent par une certaine résolution lorsqu'ils sont tracés sur un écran. Ce paramètre contrôle la largeur maximale du tracé en pixels.

— Pas de discrétisation : Cette étape est utilisée pour l'intégration lorsque des systèmes hybrides sont utilisés dans le modèle.

Dans notre cas, les deux paramètres à régler sont α et ϵ. En effet, α permet de définir le risque que la probabilité calculée se trouvent en dehors de l'intervalle de confiance. Par défaut, ce risque est fixé à 5%. Le paramètre ϵ permet de fixer la largeur de l'intervalle de confiance, il est également fixé par défaut à 5%. Avec les valeurs par défaut, la vérification de la propriété permet d'obtenir un intervalle de confiance à 95% avec une largeur de 95% également (i.e. $[\mu - 0.05, \mu + 0.05]$ avec μ se trouve dans l'intervalle de confiance avec 95% de chance).

À partir de cette définition de l'étude de cas ainsi que les données d'entrée disponibles, nous appliquons le processus d'évaluation de la robustesse en commençant par la partie instanciation des modèles puis l'évaluation de la robustesse. Dans ce qui suit, les deux étapes de l'approche sont illustrées.

3.5.1.3 Illustration de la mesure du niveau de service

L'implémentation du cas d'étude adopté tout au long de ce chapitre permet d'évaluer la robustesse de l'ordonnancement s (Figure 3.3) en considérant l'ensemble de perturbations $Pert = \{u_1^{ex}, h^1, u_1^{h_1}\}$. La deadline à satisfaire $\tilde{d}$ prend, dans ce cas, la valeur suivante $\tilde{d} = 110\% C_{max}^{ref}(s)$. En considérant le Makespan de s, $C_{max}^{ref}(s) = 38\ UT$ la deadline considérée est donc $\tilde{d} = 42$. Dans ce cas, le décideur autorise une marge de 10% sur la durée totale de l'ordonnancement.

Lors de l'exécution du processus d'évaluation, le premier résultat du Model-Checker est l'intervalle de confiance du niveau de service RL. Pour chaque ordonnancement évalué, l'intervalle obtenu se présente sous la forme suivante : $[Average(RL) - 0,02, Average(RL) + 0,02]$. Nous considérons que la valeur du niveau de service est alors $RL = Average(RL)$. La deuxième information obtenue est le nombre d'exécutions nécessaires pour respecter l'intervalle de confiance pour chaque programme.

Ordonnancement	C_{max}^{ref}	$\tilde{d}$	CI of RL	RL_i	$Time(s)$
s	38	42	$[0.69, 0.71]$	70%	2,872

TABLE 3.6 – Résultats de l'évaluation de la robustesse

A la fin de l'exécution du processus d'évaluation (Table 3.6), le niveau de service obtenu est de 70%. Ce qui indique que l'ordonnancement permet d'absorber les perturbations en respectant la deadline à 70%.

3.6 Conclusion

Ce chapitre a présenté le processus d'évaluation permettant de mesurer le niveau de service et ainsi d'évaluer la performance de la robustesse d'un ordonnancement sous perturbations. Le processus est instancié en fonction des données d'entrée fourni par le décideur (l'ordonnancement a évaluer s, l'ensemble de perturbations à prendre en compte $Pert$ et la deadline à satisfaire $\tilde{d}$), le niveau de service mesuré par le processus est destiné au décideur pour qu'il puisse l'interpréter suivant ces besoins et contraintes. Le processus est constitué de deux étapes : la modélisation et l'évaluation. Dans l'étape de modélisation, une structure basée sur le concept du Plug and Play est proposée. La modélisation du problème de l'ordonnancement sous perturbations est basée sur des patrons de modèles d'automates temporisés stochastiques. L'approche permet ainsi l'instanciation en fonction des paramètres de l'atelier et de perturbations sans efforts supplémentaires de modélisation. Dans l'étape d'évaluation, la spécification du niveau de service est exprimée sous forme d'une propriété PCTL permettant ainsi d'effectuer du Model-Checking statistique.Pour exécuter le processus d'évaluation, les deux étapes de modélisation et d'évaluation ont été implémenté sur l'outil UppAal SMC permettant la modélisation des ATS et le Model-Checking statistique. Pour illustrer la faisabilité de l'évaluation de la robustesse par le biais du processus proposé, l'approche a été illustrée tout le long du chapitre sur une étude de cas d'un atelier Job Shop Flexible.

Chapitre 4

Analyse des performances de l'approche d'évaluation

Sommaire

4.1 Introduction 82
4.2 Généricité du processus d'évaluation 82
4.2.1 Généricité vis-à-vis de l'atelier de production 82
4.2.2 Généricité vis-à-vis des perturbations 84
4.3 Sensibilité du processus d'évaluation 87
4.3.1 Méthodologie 88
4.3.2 Sensibilité aux paramètres du Model-Checker 91
4.3.3 Sensibilité du processus aux paramètres d'entrée 93
4.4 Passage à l'échelle 100
4.4.1 Construction du plan d'expériences 101
4.4.2 Exécution du plan d'expérience 101
4.4.3 Analyse des résultats du passage à l'échelle 101
4.4.4 Passage à l'échelle : synthèse 103
4.5 Conclusion 104

4.1 Introduction

L'approche d'évaluation de la robustesse présentée dans le chapitre précédent permet d'évaluer le niveau de service d'un ordonnancement soumis à des perturbations. Ce chapitre a pour objectif d'évaluer cette approche selon deux points de vue. Le premier s'intéresse à la généricité de l'approche et le deuxième s'intéresse à ses performances et leur sensibilité par rapport aux paramètres d'entrée.

4.2 Généricité du processus d'évaluation

Un des arguments initiaux de la thèse pour la définition et la mise en œuvre d'une approche adaptée pour l'ordonnancement robuste basée sur les modèles et outils des Systèmes à Èvènements Discrets (SED) reposait sur le constat que les méthodes existantes principalement basées sur les modèles et outils de la Recherche Opérationnelle étaient souvent dédiées à un problème d'ordonnancement spécifique (type d'atelier considéré, perturbations considérées). Ainsi, l'objectif est ici d'évaluer dans quelle mesure, l'approche proposée dans le chapitre 3 est générique. Deux questions sont abordées :

— L'approche est-elle générique vis-à-vis du type d'atelier de production considéré ?
— L'approche est-elle générique vis-à-vis des perturbations considérées ?

4.2.1 Généricité vis-à-vis de l'atelier de production

L'approche que nous avons proposée permet d'évaluer le niveau de service d'un ordonnancement s soumis à un ensemble de perturbations $Pert$ et devant satisfaire une deadline $\tilde{d}$. Cet ordonnancement fourni par le décideur doit être exécuté dans un atelier de production et satisfait donc les contraintes associées. Il est donc légitime de se poser la question de la capacité de notre approche à pouvoir prendre en compte des ordonnancements quel que soit le type d'atelier associé. La littérature permet notamment d'identifier quatre types d'atelier [Pinedo, 2012] : Job Shop, Flow Shop, Open Shop et Hybrid Shop. Nous allons montrer dans cette section comment notre proposition reste valable quel que soit ce type d'atelier.

En pratique, ce point de généricité est pris en compte dans l'étape de modélisation et plus précisément dans le patron de l'opération (Figure 3.8a). En effet, le prédicat $Route(o_{jk}) == 1$ dans la garde de la première transition permet de traduire les contraintes associées à tout type d'atelier. Il s'agira juste d'instancier ce prédicat de manière cohérente suivant l'atelier considéré. Les règles d'instanciation sont définies dans la Table 4.1 adaptée à partir des premiers travaux sur la génération d'ordonnancement [Marangé et al., 2016]. Par ailleurs, le prédicat $Avail(j) == 1$ garantit que deux opérations d'un même job ne peuvent être exécutés en même temps, ce qui est une contrainte vraie quel que soit le type d'atelier.

Pour les ateliers de type Job Shop et Flow Shop, il existe des contraintes de précédence dans la gamme de chaque job (la seule différence entre les deux ateliers réside dans le fait que pour le Flow Shop, tous les jobs suivent le même trajet dans l'atelier). Un ordonnancement admissible s satisfait donc ces exigences. Le prédicat $Route(o_{jk2}) == 1$ consiste donc à vérifier que l'opération o_{jk1} précédant o_{jk2} dans la gamme du job j a été réalisée entièrement avant de pouvoir lancer l'exécution de l'opération o_{jk1}. C'est $OpComp(o_{jk1})$ qui permet de vérifier que l'opération o_{jk1} a bien été réalisée (voir patron de l'opération Figure 3.8a). Ainsi $Route(o_{jk2})$ est

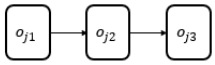
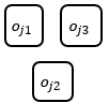
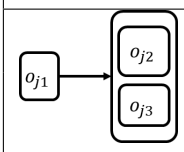
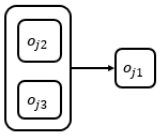
Job Shop ou Flow Shop (impliquant des contraintes de précédence entre opérations)			
	Prédicat pour o_{j1}	Prédicat pour o_{j2}	Prédicat pour o_{j3}
	$Route(o_{j1}) := 1$	$Route(o_{j2}) := OpComp(o_{j1})$	$Route(o_{j3}) := OpComp(o_{j2})$
Open Shop (sans contraintes de précédence)			
	Prédicat pour o_{j1}	Prédicat pour o_{j2}	Prédicat pour o_{j3}
	$Route(o_{j1}) := 1$	$Route(o_{j2}) := 1$	$Route(o_{j3}) := 1$
Hybrid Shop de type 1			
	Prédicat pour o_{j1}	Prédicat pour o_{j2}	Prédicat pour o_{j3}
	$Route(o_{j1}) := 1$	$Route(o_{j2}) := OpComp(o_{j1})$	$Route(o_{j3}) := OpComp(o_{j1})$
Hybrid Shop de type 2			
	Prédicat pour o_{j1}	Prédicat pour o_{j2}	Prédicat pour o_{j3}
	$Route(o_{j1}) := OpComp(o_{j2})\&OpComp(o_{j3})$	$Route(o_{j2}) := 1$	$Route(o_{j3}) := 1$

TABLE 4.1 – Règles d'instanciation des prédicats suivant le type d'atelier

égal à $OpComp(o_{jk1})$. L'atelier Flow Shop est un cas particulier du Job Shop où tous les jobs passent par les mêmes ressources.

Pour un atelier du type Open Shop, il n'y a pas de contrainte de précédence entre opérations d'une même gamme. Les opérations peuvent donc être exécutées dans n'importe quel ordre avec la seule contrainte de non-chevauchement des opérations du même job (garantie par le prédicat $Avail(j) == 1$). Donc pour un atelier de type Open Shop, le prédicat $Route(o_{jk}) == 1$ doit toujours être vrai donc $Route(o_{jk}) := 1$ pour chaque opération o_{jk}.

Le dernier type d'atelier est le Hybrid Shop. Ce type d'atelier est une combinaison des autres types d'atelier. Ainsi, les contraintes de précédence des opérations varient d'une opération à une autre. Deux types génériques d'Hybrid Shop peuvent être identifiés. Pour l'Hybrid Shop de type 1, l'opération o_{j1} doit d'abord être exécutée, avant les deux opérations o_{j2} et o_{j3} qui elles peuvent être exécutées dans un ordre quelconque. Dans le cas de l'Hybrid Shop de type 2, les opérations o_{j2} et o_{j3} peuvent être exécutées dans un ordre quelconque mais doivent être précédées de l'opération o_{j1}.

Les règles présentées dans la table 4.1, peuvent évidemment être combinées. La généricité de notre approche vis-à-vis du type d'atelier (parmi les types classiques) est donc satisfaite.

4.2.2 Généricité vis-à-vis des perturbations

Les perturbations prises en compte dans l'approche d'évaluation sont introduites par le décideur. Il est donc naturel de se poser la question de la généricité de l'approche vis-à-vis des perturbations. Cette question de généricité peut être déclinée suivant trois considérations :

- Les perturbations sont prises en compte comme des fluctuations des durées opératoires. Ces fluctuations étant modélisées comme des variables aléatoires suivant une loi de probabilité connue, est-ce que l'approche permet de prendre en compte toute loi de probabilité ?
- Les perturbations à prendre en compte dans $Pert$ étant définies par le décideur, l'approche est-elle générique vis-à-vis du type de perturbation ?
- Le décideur peut vouloir prendre en compte plusieurs perturbations en même temps. L'approche permet-elle la combinaison de perturbations ?

Ces trois considérations sont traitées dans cet ordre dans les prochaines sections.

4.2.2.1 Généricité vis-à-vis des lois de probabilités considérées

Comme présenté dans le chapitre 3, les perturbations sont prises en compte comme des fluctuations δd sur les durées opératoires d_{jkr}. Ces fluctuations sont modélisées comme des variables aléatoires avec une loi de probabilité connue. Dans le patron d'incertitude (figure 3.13), nous avons proposé une modélisation discrète de la distribution de probabilité considérée par le biais du poids de probabilité discret $p(l)$. La proposition 1 démontrée dans le chapitre 3, permet de calculer l'ensemble des $p(l)$ associé à une incertitude en fonction de la fonction de répartition $F_X(\delta d)$ de la loi de probabilité (formule rappelée dans l'équation (4.1)).

$$\begin{cases} p(0) = \frac{F_X(\delta d^- + t) - F_X(\delta d^-)}{F_X(\delta d^+ + t) - F_X(\delta d^-)} \\ p(l) = \frac{F_X(\delta d^- + (l+1)t) - F_X(\delta d^- + lt)}{F_X(\delta d^- + lt) - F_X(\delta d^- + (l-1)t)} \times \frac{p(l-1)}{1 - p(l-1)} \qquad for\ l \geq 1 \end{cases} \tag{4.1}$$

La démonstration présentée dans le chapitre 3 reste indépendante de la fonction de répartition considérée. Ainsi nous pouvons conclure que notre approche est générique vis-à-vis des lois de probabilités.

4.2.2.2 Généricité vis-à-vis du type de perturbations considérées

L'objectif de la structure de modélisation proposée est de permettre au décideur la liberté de traiter l'ensemble de perturbations qui convient à son objectif d'évaluation de la robustesse et aux caractéristiques stochastiques de son atelier de production. L'approche que nous proposons est basée sur deux patrons de perturbations instanciables.

Afin d'évaluer la capacité de notre approche de modélisation à intégrer de nouveaux scénarios de perturbations, nous nous intéressons au benchmark proposé dans la littérature par [Trentesaux et al., 2013]. Ce benchmark à l'intérêt de ne pas être défini par nous et d'offrir un ensemble de scénarios de perturbation très variés. Il s'intéresse à une plateforme de production du type Job Shop flexible. Il nous permet de nous mettre dans la situation où le décideur dispose d'une étude de l'atelier avec une description des perturbations qui peuvent éventuellement se produire durant l'ordonnancement.

Ce benchmark est un ensemble de quinze scénarios différents de perturbations (Tableau. 4.2). À partir de la description fournie pour chaque scénario, nous avons interprété et exploité les perturbations considérées. Sur les quinze scénarios traités, nous pouvons en intégrer dix dans notre approche et ainsi les évaluer. Les cinq autre scénarios sont des scénarios dans lesquels un ré-ordonnancement est nécessaire, ce qui n'entre pas dans le cadre de notre approche puisque cela nécessite une approche réactive.

Scénario	Description	Interprétation	Intégration
Ps_1	Redémarrage du système de production et attente connue	Aléa h impliquant une fluctuation constante sur la première opération de chaque machine	Pour toute opération $o_{j_r k_r}$ définie comme la première opération à exécutée sur la machine r : $d_{j_r k_r r} = d^{ref}_{j_r k_r r} + H^h_{j_r}.delay$ et $p(h, o_{j_r k_r}) = 100\%$
Ps_2	Une nouvelle opération peut être exécutée sur une ressource	Ne peut être traitée avec l'approche car nécessitant un ré-ordonnancement	
Ps_3	Une nouvelle ressource est ajoutée au système de production	Ne peut être traitée avec l'approche car nécessitant un ré-ordonnancement	
Ps_4	Sur une période connue, une ressource est impactée par une usure d'outil dû entraînant une augmentation de la durée des tâches	Aléa de type usure avec une durée d'allongement des tâches.	Pour toute opération o_{jk} exécutée sur la ressource usée r_i dans l'horizon concerné : $d_{jkr_i} = d^{ref}_{jkr_i} + H^h_{jk}.d^{ref,h}_{jkr_i}$ and $p(h, o_{jk}) = 100\%$
Ps_5	À un moment donné, un job urgent apparaît	Ne peut être traitée avec l'approche car nécessitant un ré-ordonnancement	
Ps_6	Un problème de qualité permet de refaire un job	Aléa liée au contrôle de qualité sur le job j	Pour toute les opérations du job j : $d_{jkr} = d^{ref}_{jkr} + H^h_{jk}.d^{ref,h}_{jkr}$ avec $d^{ref}_{jkr} = -d^{ref}_{jkr}$ et $p(h, o_{jk}) = 100\%$
Ps_7	Le système de convoyeur est en maintenance	Aléa sur la ressource du convoyeur	Pour toute les opérations exécutée sur le convoyeur r_i : $d_{jkr_i} = d^{ref}_{jkr_i} + H^h_{jk}.d^{ref,h}_{jkr_i}$
Ps_8	Système nécessitant un réapprovisionnement	Aléa impactant une opération donnée liée à la rupture de réapprovisionnement sur la ressource r_i générant un retard.	Une seule opération o_{jk} est impactée : $d_{jkr_i} = d^{ref}_{jkr_i} + H^h_{jk_i}.d^{ref,h}_{jkr_i}$ avec $p(h, o_{jk}) = 100\%$
Ps_9	Panne de la ressource r_i	Aléa impactant une des opérations exécutées sur r_i	Pour la première opération exécutée sur r_i : $d_{jkr_i} = d^{ref}_{jkr_i} + H^h_{jk}.d^{ref,h}_{jkr_i}$ and $p(h, o_{jk}) = 100\%$
Ps_{10}	Panne de la ressource critique r_i	Aléa liée à la ressource r_i générant une durée de maintenance	Même intégration que Ps_9
Ps_{11}	Problème de qualité sur une ressource r_i impactant 50% des opérations exécutées sur r_i	Aléa rajoutant un retard dû aux retouches pour améliorer la qualité.	Pour toute opération o_{jk} exécutée sur r_i : $d_{jkr_i} = d^{ref}_{jkr_i} + H^h_{jk}.d^{ref,h}_{jkr_i}$ and $p(h, o_{jk}) = 50\%$
Ps_{12}	Une ressource r_i devient indisponible suite à une défaillance	Aléa impactant les opérations exécutées sur r_i et générant une durée de maintenance	Même intégration que Ps_9
Ps_{13}	Arrivée d'un nouveau job après la fin du dernier	Ne peut être traitée avec l'approche car nécessitant un ré-ordonnancement	
Ps_{14}	Mise en arrêt du système de production après la fin de la dernière opération du premier job	Aléa impactant toute les opérations exécutées après le premier job j_1.	Pour les opérations traitées après les opérations du job j_1 : $d_{jkr} = d^{ref}_{jkr} + H^h_{jk}.d^{ref,h}_{jkr}$ avec $p(h, o_{jk}) = 100\%$
Ps_{15}	Tout le système de production est mis en pause après la fin du du quatrième job	Aléa impactant les opérations traitées après le job j_4.	Même intégration que Ps_14

TABLE 4.2 – Scénarios de perturbations interprétés du benchmark de [Trentesaux et al., 2013]

Par exemple, le premier scénario Ps_1 défini considère que le système de production doit redémarrer au début de l'horizon de production avec un temps de retard connu. l'interprétation de ce scénario permet de considérer une perturbation de type Aléa. En effet, l'occurrence du redémarrage peut être interprétée comme étant un aléa ayant une probabilité d'occurrence $p(h, o_{jk}) = 100\%$ pour toutes les opérations o_{jk} qui sont exécutées en premier sur chaque ressource r. Le redémarrage du système va générer une durée *delay* indépendante des opérations représentant le retard engendré par cet aléa.

Pour le scénario Ps_{11}, les auteurs décrivent un problème de qualité sur une ressource r_i impliquant un délai supplémentaire de retouche pour 50% des opérations affectées à la ressource. Cette description est interprétée comme étant l'occurrence d'un aléa lié à un problème de qualité ayant une probabilité d'occurrence égale à $p(h, o_{jk}) = 50\%$ pour toute les opérations o_jk qui sont exécutées sur r_i. Et cette retouche induit une durée de retouche connue.

Nous remarquons que la majorité des scénarios définis par le benchmark peut être interprétée comme un aléa générant un retard connu sur un ensemble d'opérations connu.

Les résultats de cette interprétation du benchmark confirment la capacité de notre approche à intégrer des scénarios de perturbations pour autant que celles-ci puissent être interprétées comme une fluctuation de la durée opératoire et ne nécessitent pas de ré-ordonnancement.

4.2.2.3 Généricité vis-à-vis de la combinaison de perturbations

La structure de l'approche proposée permet non seulement de prendre en compte différents types de perturbations, mais aussi de combiner une ou plusieurs perturbations. Cela signifie que le décideur peut choisir d'évaluer la robustesse d'un ordonnancement face à la combinaison de différents types de perturbations. Il est alors possible d'instancier les patrons de perturbations autant de fois que nécessaire pour considérer l'ensemble des perturbations voulu par le décideur (Figure 4.1).

Pour illustrer le principe de combinaison des perturbations, nous considérons les six scénarios suivants :

- Scénario 1 : ce scénario permet de considérer une seule perturbations de type incertitude sur la durée d'exécution. L'équation de d_{jkr} devient donc : $d_{jkr} = d^{ref}_{jkr} + \delta d^{u_1^{ex}}_{jkr}$.
- Scénario 2 : ce scénario permet de considérer la perturbations de type aléa qui est la panne ressource. L'expression de d_{jk} devient donc : $d_{jkr} = d^{ref}_{jkr} + H^{h_1}_{jk}.d^{ref,h_1}_{jkr}$
- Scénario 3 : ce scénario combine deux perturbations de type aléa, la panne ressource et la rupture de réapprovisionnement. En combinant ces deux aléas l'expression de d_{jkr} devient : $d_{jkr} = d^{ref}_{jkr} + H^{h_1}_{jk}.d^{ref,h_1}_{jkr} + H^{h_2}_{jk}.d^{ref,h_2}_{jkr}$.
- Scénario 4 : ce scénario combine deux perturbations de type incertitude, l'incertitude sur la durée d'exécution et l'incertitude sur la demande. L'expression de d_{jkr} dans ce cas devient : $d_{jkr} = d^{ref}_{jkr} + \delta d^{u_1^{ex}}_{jkr} + \delta d^{u_2^{ex}}_{jkr}$
- Scénario 5 : ce scénario combine les deux types de perturbations aléas et incertitude. Dans ce scénario, l'incertitude liée à l'aléa. L'expression de d_{jkr} devient : $d_{jkr} = d^{ref}_{jkr} + \delta d^{u_1^{ex}}_{jkr} + H^{h_1}_{jk}.\left(d^{ref,h_1}_{jkr} + \delta d^{u_1^{h_1}}_{jkr}\right)$.
- Scénario 6 : ce scénario permet de combiner l'ensemble des perturbations et ainsi d'obtenir une expression de d_{jkr} globale.

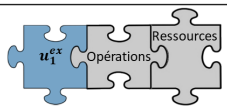
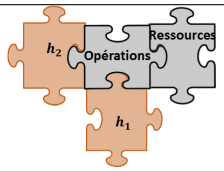
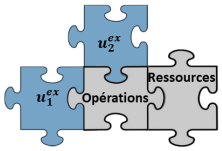
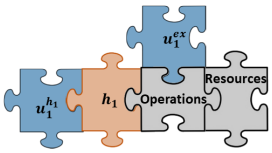
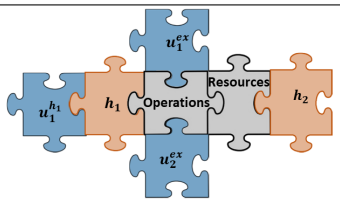
ID Scénario	Construction de Pert	Expression de d_{jkr}
Scénario 1: Incertitude sur la durée d'exécutions		• $d_{jkr} = d_{jkr}^{ref} + \delta d_{jkr}^{u_1^{ex}}$
Scénario 2: Panne ressource		• $d_{jkr} = d_{jkr}^{ref} + H_{jk}^{h_1}.d_{jkr}^{ref,h_1}$
Scénario 3: Panne ressource, Rupture de réapprovisionnement		• $d_{jkr} = d_{jkr}^{ref} + H_{jk}^{h_1}.d_{jkr}^{ref,h_1} + H_{jk}^{h_2}.d_{jkr}^{ref,h_2}$
Scénario 4: Incertitude sur la durée d'exécutions, Incertitude sur la demande		• $d_{jkr} = d_{jkr}^{ref} + \delta d_{jkr}^{u_1^{ex}} + \delta d_{jkr}^{u_2^{ex}}$
Scénario 5: Incertitude sur la durée d'exécutions, Panne ressource, Incertitude sur la durée de maintenance,		• $d_{jkr} = d_{jkr}^{ref} + \delta d_{jkr}^{u_1^{ex}} + H_{jk}^{h_1}.(d_{jkr}^{ref,h_1} + \delta d_{jkr}^{u_1^{h_1}})$
Scénario 6: Combinaison de toutes les perturbations		• $d_{jkr} = d_{jkr}^{ref} + (\delta d_{jkr}^{u_1^{ex}} + \delta d_{jkr}^{u_2^{ex}}) +$ $(H_{jk}^{h_1}.(d_{jkr}^{ref,h_1} + \delta d_{jkr}^{u_1^{h_1}}) + H_{jk}^{h_2}.d_{jkr}^{ref,h_2})$

FIGURE 4.1 – Scénario de combinaison de plusieurs perturbations

Chaque scénario considéré permet de combiner différentes perturbations, et de les intégrer dans notre approche par l'instanciation de d_{jkr} (Figure 3.15).

4.3 Sensibilité du processus d'évaluation

Nous avons étudié dans la section précédente la capacité de notre approche à être générique vis-à-vis de certains paramètres. Dans un souci de mise en œuvre, il est également important d'analyser les performances de notre approche en terme de résolution. Les performances concernent le temps de résolution et le nombre de simulations. Afin d'analyser les performances du processus, nous avons identifié trois questions qui nous semblent essentielles :

1. Quel est la sensibilité de ces performances aux paramètres de contrôle du Model-Checker ?
2. Quel est la sensibilité de ces performances aux paramètres d'entrée du problème d'évaluation ?
3. Quel est l'impact de la taille de l'atelier de production sur les performances du processus ?

Les questions formulées permettent d'analyser les capacités de résolution du processus d'évaluation et de déterminer ses limites d'utilisation.
La première question permet d'analyser les effets des paramètres du Model-Checker sur les performances du processus d'évaluation.
La deuxième question permet d'établir un rapport de cause à effet entre les paramètres d'entrée du processus et les performances de résolution de ce dernier. L'objectif de cette analyse est d'identifier les paramètres ayant un impact non négligeable sur les performances du processus d'évaluation. La dernière question traite du passage à l'échelle. L'évaluation de l'impact de la taille de l'atelier permettra ainsi d'envisager une application dans un contexte industriel.

Cette section est structurée de la façon suivante. Nous présentons tout d'abord la méthodologie adoptée et ses différentes étapes. Nous détaillons ensuite l'outillage utilisé pour exécuter cette méthodologie. Enfin nous traitons les résultats des expérimentations pour chacune des questions.

4.3.1 Méthodologie

Le processus d'évaluation détaillé dans le Chapitre 3 permet de calculer le niveau de service d'un ordonnancement soumis à des perturbations. Les éléments d'entrée du processus sont l'ordonnancement s, les perturbations $Pert$ et la deadline $\tilde{d}$. L'ordonnancement s est défini, entre autre, par deux choses : la taille de l'atelier associé et la règle d'ordonnancement qui a permis d'aboutir à s. La taille de l'atelier est le paramètre permettant de traiter la question du passage à l'échelle. Les autres paramètres permettent le traitement de la question de sensibilité du processus d'évaluation. Dans le processus d'évaluation, des paramètres de contrôle sont identifiés. En effet, pour effectuer l'évaluation du niveau de service, il est nécessaire au préalable de régler les paramètres du Model-Checker. À la sortie du processus d'évaluation, nous pouvons analyser deux performances. Pour calculer le niveau de service, un nombre de simulation $NbSim$ est exécuté pour atteindre l'intervalle de confiance (voir section Model-Checking Chapitre 3). De plus, l'exécution de ces simulations nécessite un temps de résolution $Time$. Ces deux éléments sont donc considérés pour l'analyse des performances de résolution du processus d'évaluation (Figure 4.2). Et lorsqu'on parle de sensibilité, il s'agit bien d'évaluer la sensibilité de ces deux performances aux paramètres d'entrée présentés précédemment.

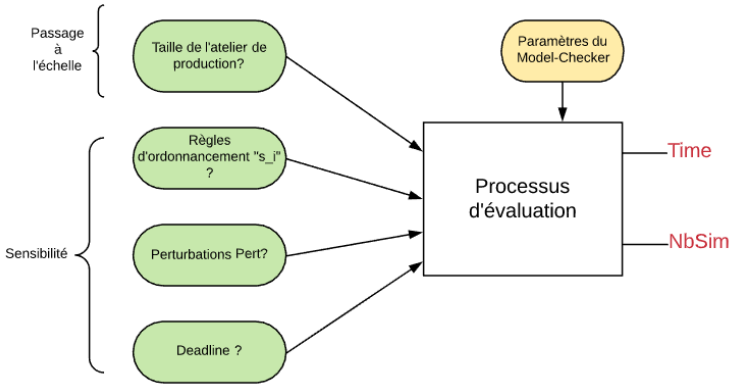


FIGURE 4.2 – Paramètres de l'expérimentation

Pour mener à bien cette expérimentation, une méthode basée sur le principe des plans d'expériences est définie. La méthode des plans d'expériences permet d'organiser la phase d'ex-

périmentation tout en optimisant le nombre d'expériences nécessaires [Goupy, 2006]. Un plan d'expériences permet d'établir un lien entre une réponse Y et des facteurs X_i ($Y = F(X_i)$). Dans notre cas, deux réponses nous intéressent ($Y_1 = Time$, $Y_2 = NbSim$) et les facteurs sont les différents paramètres d'entrée du processus d'évaluation.

Cette méthode d'expérimentation est basée sur trois phases.

1. Construction du plan d'expériences à partir des paramètres disponibles à l'entrée du processus d'évaluation. Pour ce faire, les facteurs d'entrées du plan d'expérience ainsi que les réponses que l'on souhaite observer sont définis. Ensuite, le nombre et le contenu des niveaux des facteurs sont établis. À la fin de cette première étape, plusieurs hypothèses sont émises pour mener à bien la suite de l'expérimentation.
2. Exécution du plan d'expériences. Tout d'abord, la génération des expériences permet de déterminer le nombre d'expériences à exécuter et de s'assurer de l'homogénéité des expériences. Par la suite, les expériences sont implantées dans le processus d'évaluation. Les résultats des expériences sont récupérés à la fin de cette étape.
3. Analyse des résultats. Faire une analyse de l'impact des facteurs sur les caractéristiques de performance. Dans un premier temps, une analyse graphique des résultats va permettre de mettre en évidence les impacts existants. Afin de prouver statistiquement les résultats graphiques, une analyse statistique permettra d'amener des conclusions pertinentes à la question de l'impact des paramètres d'entrée sur les réponses ($Time$ et $NbSim$).

4.3.1.1 Construction du plan d'expériences

La construction du plan d'expériences est une étape effectuée en fonction de la question traitée. Pour traiter le passage à l'échelle, le plan d'expériences à construire dépend de la taille de l'atelier de production. Donc les facteurs seront liés à l'atelier de production considérés. Dans le cas de la sensibilité, le plan d'expériences est construit en fonction des paramètres d'entrée du problème d'évaluation ($Pert$, $\tilde{d}$, s et les paramètres du model-checker).

La première étape pour construire le plan d'expériences est l'identification des facteurs pouvant impacter la performance du processus d'évaluation, à savoir le temps de résolution et le nombre de simulations. Un recensement des paramètres du problème est important. Nous avons décidé de décomposer cette analyse en trois questions définissant trois familles de facteurs et autant de plans d'expériences à réaliser :

— la sensibilité par rapport aux paramètres du model-checker
— la sensibilité par rapport aux paramètre d'entrées du processus : la règle d'ordonnancement pour s, les perturbations $Pert$ et la deadline $\tilde{d}$
— le passage à l'échelle par rapport à la taille de l'atelier (sensibilité par rapport à l'atelier)

La deuxième étape est l'identification des différents niveaux que peuvent prendre les facteurs. Un niveau désigne une valeur quantitative ou qualitative que le facteur peut prendre. Dans la majorité des plans d'expériences, les facteurs ont un même nombre de niveaux, ce qui permet d'établir un plan d'expérience factoriel du type n^k avec n le nombre de niveaux et k le nombre de facteurs.

4.3.1.2 Exécution du plan d'expériences

Pour exécuter les plans d'expériences identifiés (un plan pour chacune des questions), il a été nécessaire de développer un outillage adapté. L'objectif de l'outillage est, tout d'abord, de

faciliter l'exécution de l'expérimentation et ainsi permettre l'automatisation du processus d'évaluation pour n expériences.

Le processus d'expérimentation (Figure 4.3) permet de visualiser les besoins d'outillage pour cette phase. En effet, au début d'une expérimentation les deux informations nécessaires sont les ordonnancements à évaluer ainsi que les perturbations à prendre en compte. À partir de ces informations, un plan d'expériences peut être défini. L'étape suivante est donc l'exécution du processus d'évaluation de la robustesse pour chaque expérience du plan généré. A la fin de l'évaluation du plan, les résultats peuvent être récupérés dans un fichier adapté. L'outillage utilisé doit permettre la reproductibilité et la répétabilité de ce processus. Le programme a comme entrée des fichiers Excel généré à l'aide d'algorithmes développés pour l'occasion et il produira à la fin de son exécution un fichier Excel contenant les résultats de l'expérimentation. Afin d'instancier les modèles en fonction des données du fichier Excel effectuée sur UppAal SMC, le programme devra générer des fichiers de type XML.

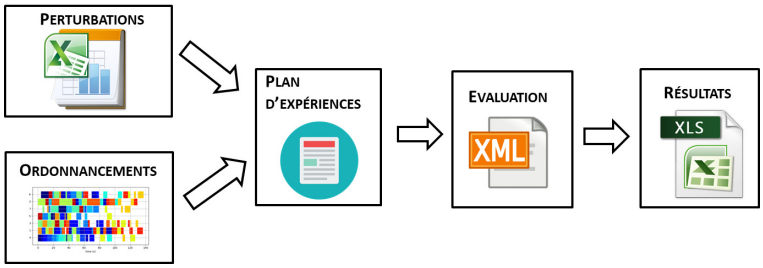


FIGURE 4.3 – Processus d'expérimentation

Pour tester la sensibilité à l'ordonnancement d'entrée s, nous avons défini une méthode de génération d'ordonnancement déterministe qui permet de fournir plusieurs ordonnancements s correspondant à des règles d'ordonnancement prédéfinies.

Trois éléments sont donc récupérés à partir du processus d'expérimentation, le premier est le diagramme de Gantt de l'ordonnancement généré et le deuxième est un fichier de sortie (.out) contenant les informations de l'ordonnancement (date de début et de fin de chaque opération et l'allocation des opérations aux ressources). Le dernier fichier contient les résultats de l'exécution du processus d'évaluation.

4.3.1.3 Analyse des résultats

Pour analyser les résultats des exécutions des plans d'expériences, deux types d'outil d'analyses sont utilisés : les outils graphiques et les outils statistiques. En effet, pour visualiser les effets des facteurs sur les performances de l'approche, le graphique des effets principaux est utilisé. Ce graphique permet de visualiser l'effet d'un facteur sur une réponse en comparant les moyennes suivant le niveau des facteurs. Cette comparaison permet de savoir si cette différence génère une incidence sur la réponse. Les graphiques des effets principaux permet d'afficher, pour chaque facteur, la moyenne de chaque niveau. Ces moyennes sont connectées par une ligne. Par exemple, dans (Figure 4.4), le graphique présente les effets de deux facteurs (engrais, emplacement) sur la croissance d'une plante. Ce graphique suggère que les deux facteurs ont un impact sur la réponse (puisque la réponse varie suivant le niveau du facteur). Le graphique des effets principaux donne une information purement qualitative qu'il convient de valider à l'aide d'un

outil statistique comme l'analyse de la variance.

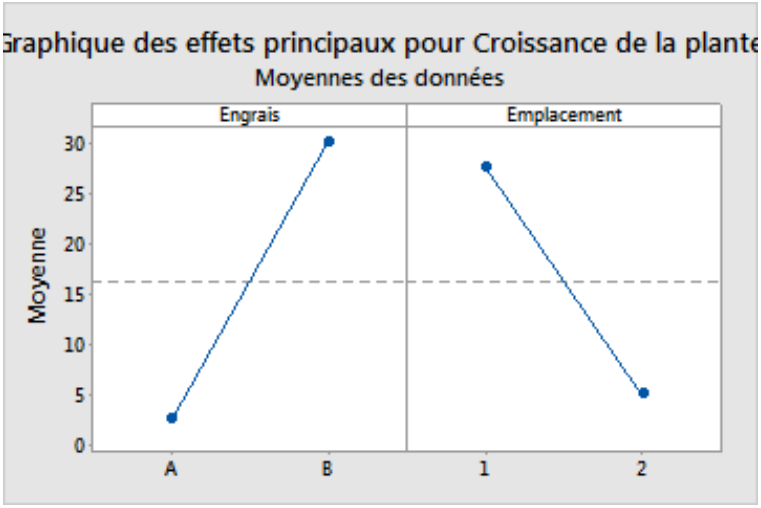


FIGURE 4.4 – Exemple du graphique des effets principaux (Source : Minitab)

L'analyse de la variance (ANAVAR) permet de faire une analyse statistique pour vérifier si les facteurs ont un impact réel sur la sortie étudié. Plus précisément en comparant les moyennes, deux hypothèses sont vérifiées. L'hypothèse nulle permet de considérer que les moyennes sont égale alors que l'hypothèse alternative considère que les moyennes diffèrent. Pour reprendre l'exemple de la figure 4.4, l'hypothèse nulle serait celle qui consisterait à dire que la croissance moyenne de la plante pour l'emplacement 1 est la même que la croissance moyenne de la plante pour l'emplacement 2 (le facteur emplacement n'a donc pas d'effet). L'hypothèse alternative est celle qui correspond à des croissances moyennes différentes selon l'emplacement (le facteur emplacement a donc un effet). Ainsi l'analyse de la variance est un complément nécessaire au graphique des effets puisqu'elle permet de démontrer statistiquement un effet.

Les résultats d'une analyse de la variance renvoient notamment une probabilité p pour chaque facteur étudié qu'il convient d'associer à un seuil α (classiquement fixé à 0,05) pour interpréter les résultats :

— Si $p < \alpha$ alors l'hypothèse nulle est rejetée (le facteur a un effet sur la réponse),
— Sinon l'hypothèse nulle est acceptée (le facteur n'a pas d'effet).

4.3.2 Sensibilité aux paramètres du Model-Checker

La phase d'expérimentation cherche à déterminer les éléments impactant les performances du processus d'évaluation. Pour que cet objectif soit atteint dans les meilleures conditions, la question de l'indépendance des résultats face au Model Checker doit être traitée.
Le Model Checker statistique développé par UppAal SMC permet de régler un ensemble de paramètres d'entrée statistique en fonction de la propriété à vérifier.

Les deux paramètres statistiques qui nous intéressent sont notés α et ϵ. En effet, le paramètre α, permet le calcul de l'intervalle de confiance du niveau de service. La valeur qui lui est donnée par défaut est $0,05$ ce qui équivaut à un intervalle de confiance à 95%. Le paramètre α permet de définir le risque que la valeur de probabilité se trouve en dehors de l'intervalle de confiance. Le paramètre ϵ permet quant à lui de définir la précision dans l'intervalle de confiance, plus

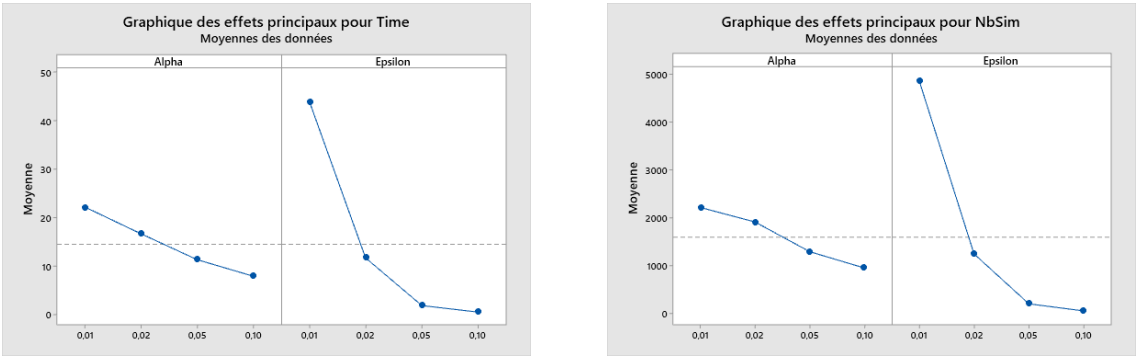
concrètement, ce paramètre permet d'indiquer la largeur de l'intervalle de confiance calculé. Par défaut, la valeur de ce paramètre est définie à $0,05$. Dans ce qui suit, nous cherchons à déterminer si le choix de ces deux paramètres à un effet sur la performance de l'approche dans l'objectif de s'assurer que le Model Checker ne biaise pas l'expérimentation mais aussi de trouver les valeurs optimales à attribuer aux paramètres.
Pour ce faire, nous procédons à un ensemble d'expérimentations en modifiant à chaque expérience la valeur de α et ϵ. L'expérimentation a été effectuée en se référant à l'exemple d'illustration [Giard, 2003] avec les paramètres considérés dans la phase d'illustration (Chapitre 3). À partir de ces paramètres, un plan d'expériences est construit en considérant α et ϵ comme étant des facteurs ayant 4 niveaux (Table 4.3). Les quatre niveaux définis permettent de considérer un panel de valeurs allant de $0,01$ à $0,1$, permettant de faire varier le risque α et la largeur de l'intervalle de confiance ϵ.

	Facteurs	
	α	ϵ
Niveaux des facteurs	$0,01$	$0,01$
	$0,02$	$0,02$
	$0,05$	$0,05$
	$0,1$	$0,1$

TABLE 4.3 – Niveaux des paramètres du Model Checker

Après la génération du plan d'expériences, les résultats sont analysés à l'aide des graphiques et outils statistiques détaillés dans la section précédente. Nous évaluons l'effet des facteurs α et ϵ sur les deux performances du processus $Time$ (le temps de résolution) et $NbSim$ (le nombre de simulations générées) à l'aide des graphiques des effets principaux.

À partir des graphiques des effets principaux (Figure 4.5), les deux facteurs ϵ et α semblent avoir un effet sur le temps de résolution ($Time$) et le nombre de simulations ($NbSim$) même si celui de α est plus faible.



(a) Effets principaux des paramètres du Model-Checker sur $Time$

(b) Effets principaux des paramètres du Model-Checker sur $NbSim$

FIGURE 4.5 – Effets des paramètres du Model-Checker sur les performances du processus

Ces analyses sont confirmées par l'analyse de la variance (Figure 4.6). Les valeurs de p sont supérieures à 0,05 pour α (donc ce paramètre n'a pas d'influence sur le temps de résolution et le nombre de simulations générées) alors qu'elles sont nulles pour ϵ (donc ce paramètre à

une influence sur les deux réponses. On peut l'expliquer par le fait que ϵ fixe la largeur de l'intervalle de confiance et donc fixe la précision désirée. Plus la précision est fine plus le nombre de simulations pour atteindre cette précision va être grand tout comme le temps de résolution.

Analyse de la variance

Source	DL	SomCar ajust	CM ajust	Valeur F	Valeur de p
Alpha	3	465,5	155,17	1,67	0,242
Epsilon	3	4915,6	1638,54	17,61	0,000
Erreur	9	837,3	93,03		
Total	15	6218,4			

(a) ANAVAR pour $Time$

Analyse de la variance

Source	DL	SomCar ajust	CM ajust	Valeur F	Valeur de p
Alpha	3	3971258	1323753	1,99	0,187
Epsilon	3	60623254	20207751	30,33	0,000
Erreur	9	5995582	666176		
Total	15	70590093			

(b) ANAVAR pour $NbSim$

FIGURE 4.6 – Analyses de la variance en fonction des paramètres ϵ et α

Pour conclure, l'analyse effectuée a pu démontrer que seule la précision de l'intervalle de confiance ϵ a un effet sur le temps de résolution et le nombre de simulations. Cette expérience permet de constater que si le besoin de précision des résultats est important pour le décideur alors l'augmentation de $Time$ et $NbSim$ est inévitable.

Pour la suite des expérimentations, nous choisissons de fixer $\alpha = 0,02$ et $\epsilon = 0.01$. Ce qui permet de construire un intervalle de confiance à 98% avec une précision d'intervalle de calcul de 1%. Autrement dit, nous partons du principe que le décideur va considérer qu'un niveau de service est différent d'un autre à l'ordre du % (en dessous de 1% de différence, deux niveaux de services sont considérés comme identiques). Et nous n'avons que 2% de risque que le niveau de service calculé ne soit pas correct (c'est-à-dire soit en dehors de l'intervalle de confiance).

4.3.3 Sensibilité du processus aux paramètres d'entrée

L'objectif ici est d'analyser si les paramètres d'entrées (règle d'ordonnancement s, $Pert$, et $\tilde{d}$) ont un impact sur les performances du processus d'évaluation.

4.3.3.1 Construction du plan d'expériences

Pour construire le plan d'expériences, l'atelier de production considéré est celui adapté de [Giard, 2003] présenté dans le chapitre 3.

Le facteur lié à l'ordonnancement s permet d'étudier l'impact de la variabilité des ordonnancements d'entrée sur les performances du processus d'évaluation. L'idée est d'évaluer si les caractéristiques de l'ordonnancement (ressources inoccupées ...) peuvent influencer le temps de résolution et le nombre de simulations. Pour ce faire, nous avons considéré trois règles de génération qui permettent d'obtenir des ordonnancements s_i avec des caractéristiques différentes. Chacune de ces règles définit un niveau pour le facteur lié à l'ordonnancement (voir table 4.4).

Les perturbations définies dans le problème d'évaluation permettent d'identifier plusieurs facteurs. Le premier facteur est la combinaison des perturbations à prendre en compte dans $Pert$. Pour analyser l'effet de la variabilité des perturbations sur les performances du processus d'évaluation, nous avons défini trois combinaisons de perturbations (Table 4.5).

Le deuxième facteur lié aux perturbations concernent les caractéristiques des perturbations considérées. Les caractéristiques associées aux incertitudes (u^{ex} et u^h) sont : les fluctuations

Facteur	Niveau	Règle d'ordonnancement	Logique
s_i	s_1	*capa_ max*	Priorité au job avec une durée d'exécution totale la plus grande ($max(\sum_j d^{ref}_{jkr})$)
	s_2	*capa_ restante_ max*	Priorité au job ayant une somme de durées d'exécution restante la plus grande
	s_3	*default_ sched*	Priorité au premier job de la liste des jobs

TABLE 4.4 – Facteur lié à la règle d'ordonnancement s_i

Facteur	Niveau	Scénario de $Pert_i$			Interprétation
		u^{ex}	h	u^h	
$Pert_i$	$Pert_1$	1	0	0	Incertitude sur la durée d'exécution des opérations
	$Pert_2$	0	1	0	Panne ressource
	$Pert_3$	1	1	1	Incertitude sur la durée d'exécution des opérations avec panne ressource et incertitude sur la durée de réparation

TABLE 4.5 – Facteur lié à la combinaison de perturbations de perturbations $Pert$

		$\delta d^{u^{ex}}_{jkr}$	$p(l)$	$\delta d^{u^h}_{jkr}$	$p(h, o_{jk})$
Niveaux des facteurs	1	$+/-10\% d^{ref}_{jkr}$	$expon$	$+/-10\% d^{ref,h}_{jkr}$	1%
	2	$+/-25\% d^{ref}_{jkr}$	$unif$	$+/-25\% d^{ref,h}_{jkr}$	3%
	3	$+/-40\% d^{ref}_{jkr}$	$norm$	$+/-40\% d^{ref,h}_{jkr}$	5%

TABLE 4.6 – Facteurs liés à $Pert$

($\delta d^{u^{ex}}_{jkr}$ et $\delta d^{u^h}_{jkr}$) et la distribution de probabilité associée au calcul de $p(l)$. Pour l'aléa h, le paramètre à considérer est la probabilité d'occurrence $p(h, o_{jk})$. Les niveaux associés à ces paramètres sont donnés dans la table 4.6.

- $\delta d^{u^{ex}}_{jkr}$: est la variation liée à l'incertitude sur la durée d'exécution de l'opération, cette variation est exprimée en pourcentage. Trois niveaux de variation sont définis (10%, 25% et 40%), ces niveaux permettent de traiter une amplitude différente de la variation, allant d'une amplitude minimale à une amplitude maximale.
- $p(l)$: est la probabilité liée aux incertitudes (sur les durées d'exécution ou liée à l'aléa). Définir les niveaux de $p(l)$ va permettre de traiter différents types de distributions de probabilité qui sont associé aux variations. Dans le cadre de cette expérimentation, la distribution considérée est la même pour toutes les incertitudes. Les trois distributions considérées sont celles traitées habituellement dans la littérature, à savoir la distribution exponentielle ($expon$), l'uniforme ($unif$) et normale ($norm$).
- $\delta d^{u^h}_{jkr}$: est la variation liée à l'incertitude sur la durée de référence de l'aléa. Cette variation impacte la durée considérée pour la perturbation d'aléa. Les trois niveaux définis pour ce facteur sont exprimés en pourcentage et permettent, comme pour le premier facteur, de traiter une plage d'amplitude allant d'un minimum 10% à un maximum de 40% passant par un niveau intermédiaire de 25%.
- $p(h, o_{jk})$: est la probabilité liée à l'impact d'un aléa sur une opération. Cette probabi-

lité est la même pour toutes les opérations de l'ordonnancement. Elle est définie par un pourcentage allant de 1% à 5% avec un niveau intermédiaire de 3%.

Le troisième et dernier facteur est la deadline $\tilde{d}$ que le décideur souhaite satisfaire. Pour faire varier ce facteur avec des valeurs les moins arbitraires possibles, nous avons décidé de fixer des niveaux qui sont fonction du Makespan de l'ordonnancement déterministe. Ainsi $\tilde{d}$ est défini comme $\tilde{d} = \tilde{d}_{perc} \times C^{ref}_{max}(s)$. Le pourcentage $\tilde{d}_{perc}$ permet de savoir quelle est la marge ajoutée à la durée de l'ordonnancement d'entrée afin d'absorber les perturbations traitées. Les trois niveaux fixés pour $\tilde{d}_{perc}$ sont donnés dans la table 4.7.

	Niveaux du facteur		
$\tilde{d}_{perc}$	110%	120%	130%

TABLE 4.7 – Facteur lié à $\tilde{d}$

4.3.3.2 Exécution et analyse des résultats du plan d'expérience

Les facteurs étant parfaitement définis ainsi que leurs niveaux, il est possible maintenant d'exécuter un plan d'expériences complet (Table 4.8).

ID	S_i	Scénario	δd^{ref}_{jkr}	$p(l)$	$\delta d^{ref,h}_{jkr}$	$p(h, o_{jk})$	$\tilde{d}$
1...	Table 4.4	Table 4.5	Table. 4.6				Table 4.7

TABLE 4.8 – Plan d'expériences pour l'analyse de sensibilité

Suite à l'exécution du plan d'expérience, nous avons pu remarquer que les résultats étaient identiques pour $Time$ et $NbSim$. C'est-à-dire, qu'un facteur ayant un effet sur $Time$ a également un effet sur $NbSim$. En fait une analyse de régression linéaire a permis de montrer que ces deux réponses évoluent linéairement ensemble.

Ainsi, dans ce qui suit, nous pourrons nous permettre d'analyser les résultats uniquement pour $Time$, puis de partager les conclusions avec $NbSim$.

4.3.3.3 Sensibilité à la règle d'ordonnancement

Les règles d'ordonnancement identifiées pour ce facteur permettent de générer trois ordonnancements différents. Pour chaque expérience, trois ordonnancements sont générés suivant les trois règles définies (*capa_ max*, *capa_ restante_ max*, *default_ sched*).

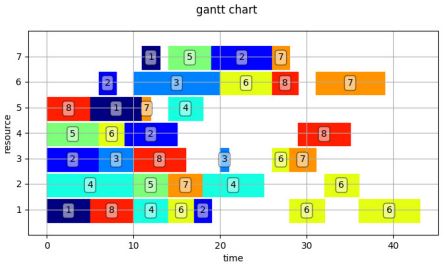


FIGURE 4.7 – Exemple d'ordonnancement s_1 : règle *capa_ max*

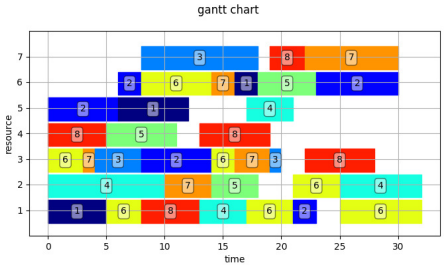


FIGURE 4.8 – Exemple d'ordonnancement s_2 : règle *capa_ restante_ max*

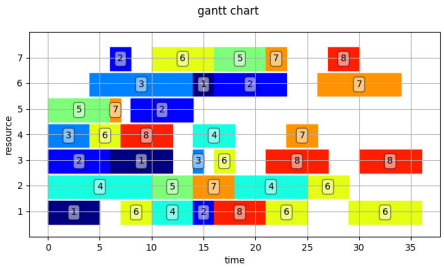


FIGURE 4.9 – Exemple d'ordonnancement s_3 : règle *default_ sched*

Ces trois règles génèrent des ordonnancements ayant différents séquencements des opérations sur les ressources. Cette différence est visible sur les exemples présentées dans les Figures 4.7, 4.8 et 4.9. En effet, ces exemples permettent de constater que chaque ordonnancement a plus au moins de temps libre sur les ressources. L'objectif de l'analyse de ce facteur est donc d'établir si cette différence entre les trois règles influence les performances du processus d'évaluation, à savoir le temps de résolution $Time$ et le nombre de simulations $NbSim$.

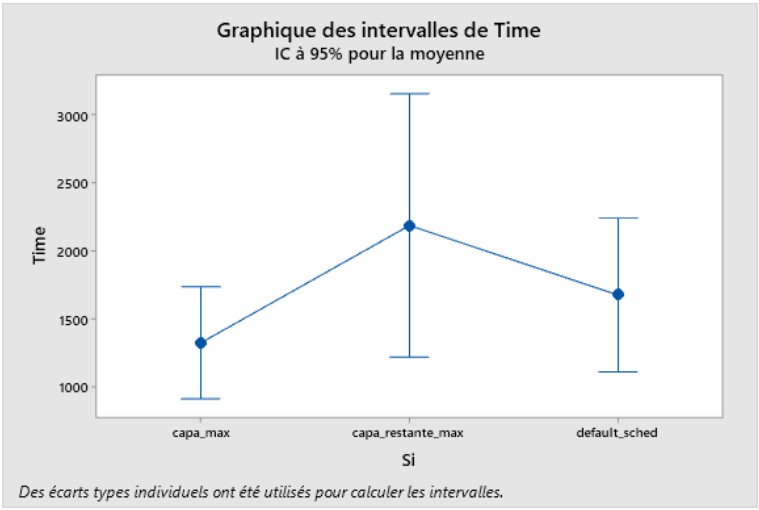


FIGURE 4.10 – Impact de la règle d'ordonnancement sur $Time$

À partir de l'analyse statistique des résultats, nous pouvons conclure que la règle d'ordonnancement utiliser pour générer l'ordonnancement s n'a pas d'impact sur les performances du processus (figure 4.10). Nous avons de plus effectué une analyse de la variance permettant de démontrer statistiquement ce résultat.

4.3.3.4 Sensibilité à la combinaison des perturbations *Pert*

Les trois ensembles de perturbations traités pour le facteur $Pert_i$ représentent les différents scénarios de combinaison des perturbations considérées. En effet, le premier ensemble $Pert_1$ introduit une perturbation de type incertitude, le deuxième ensemble $Pert_2$ traite la perturbation de type aléa alors que le troisième ensemble $Pert_3$ combine ces deux types.

L'analyse statistique effectuée permet de constater que l'ensemble des perturbations considéré a un effet sur le temps de résolution et le nombre de simulations. Cet effet est visualisé sur les graphes des effets principaux (Figure 4.11). Pour conforter ce résultat, une analyse de la variance a été effectuée (Figure 4.12). La probabilité p associée à $Pert_i$ est nulle, donc cela confirme que ce paramètre a un effet.

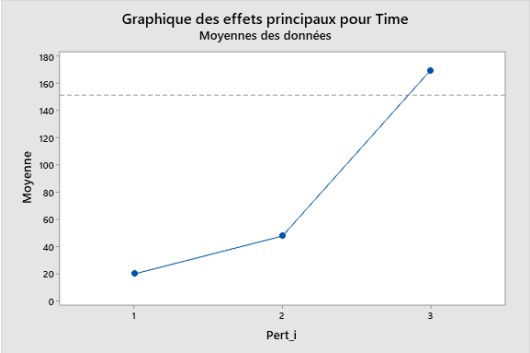


FIGURE 4.11 – Effets de $Pert_i$ sur la performance $Time$

Analyse de la variance

Source	DL	SomCar ajust	CM ajust	Valeur F	Valeur de p
Pert_i	2	645493	322746	27,24	0,000
Erreur	276	3270402	11849		
Total	278	3915895			

FIGURE 4.12 – Analyse de la variance en fonction des combinaisons de perturbations

4.3.3.5 Sensibilité aux paramètres de perturbations

Pour analyser l'effet des facteurs liés aux paramètres de perturbations, nous nous intéressons à l'ensemble de perturbations $Pert_3$. Ce scénario combine tous les types de perturbations permettant ainsi de considérer la totalité des facteurs. L'analyse statistique des paramètres de perturbations permet de générer le graphique des effets principaux sur les quatre facteurs qui nous intéressent, à savoir les variabilités liées aux incertitudes ($\delta d^{u^{ex}}_{jkr}$ et $\delta^{u^h}_{jkr}$), la distribution de probabilité suivie par les incertitudes ($p(l)$) et la probabilité d'occurrence de l'aléa ($p(h, o_{jk})$). Ce graphique permet de détecter quels sont les paramètres ayant un effet visible sur les performances du processus d'évaluation. Tous les facteurs ont un effet sur la performance du temps de résolution (Figure 4.13). Pour conforter ces remarques, une analyse de la variance est effectuée (Figure 4.14). Cette analyse permet finalement d'écarter le facteur liée aux incertitudes sur la durée de réparation ($\delta d^{u^h}_{jkr}$) des facteurs à effet. Les autres facteurs ont un effet sur les performances du processus.

D'après le graphique des effets principaux, il semblerait que les deux paramètres les plus

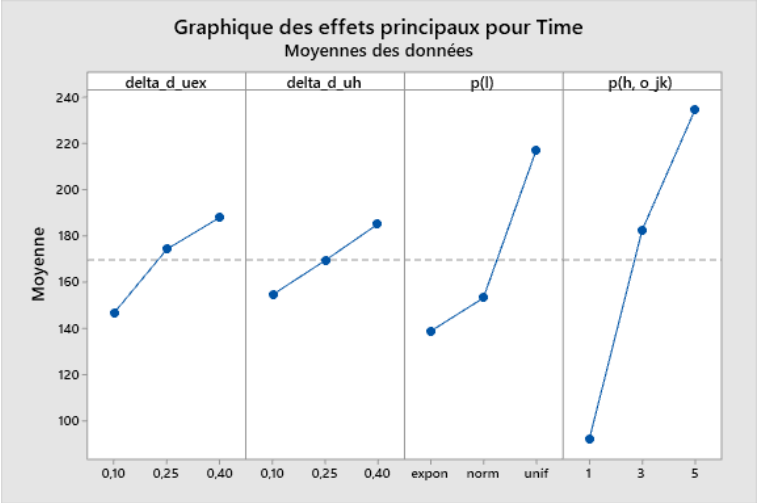


FIGURE 4.13 – Effets principaux des paramètres de perturbations sur *Time*

Analyse de la variance

Source	DL	SomCar ajust	CM ajust	Valeur F	Valeur de p
delta_d_uex	2	71817	35909	4,23	0,016
delta_d_uh	2	37642	18821	2,22	0,111
p(l)	2	282824	141412	16,65	0,000
p(h, o_jk)	2	845529	422764	49,78	0,000
Erreur	234	1987364	8493		
Inadéquation de l'ajustement	72	110775	1539	0,13	1,000
Erreur pure	162	1876589	11584		
Total	242	3225175			

FIGURE 4.14 – Analyse de la variance en fonction des paramètres de perturbations

impactant sont le type distribution ($p(l)$) et la probabilité d'occurrence de l'aléa $p(h, o_{jk})$. Nous analysons ces deux facteurs plus finement dans ce qui suit.

4.3.3.6 Sensibilité au facteur $p(l)$

Le facteur $p(l)$ permet de considérer différentes distributions de probabilités suivies par les incertitudes u^{ex} et u^h. Ce facteur a un effet sur les performances du processus d'après l'analyse de la variance effectuée. Les graphiques des intervalles (Figure 4.15) permet de montrer que les niveaux 1 et 2 se chevauchent, ils ont donc globalement le même impact sur le temps de résolution. En revanche le troisième niveau est clairement discriminant. Il semblerait donc que la loi uniforme implique une augmentation discriminante du temps de résolution.

4.3.3.7 Sensibilité au facteur $p(h, o_{jk})$

Le facteur $p(h, o_{jk})$ permet de définir trois niveaux pour la probabilité d'occurrence de l'aléa h. Cette probabilité est le facteur ayant le plus d'influence visible sur les performances du processus d'évaluation. En effet, les graphiques des intervalles permettent de conforter cette hypothèse et d'identifier la nature cet impact. Les intervalles de confiances des niveaux définis sont distincts et ne présentent aucun chevauchement ce qui implique que le passage d'un niveau est discriminant pour le temps de résolution *Time* (Figure 4.16). Il est donc notable que l'augmentation du facteur

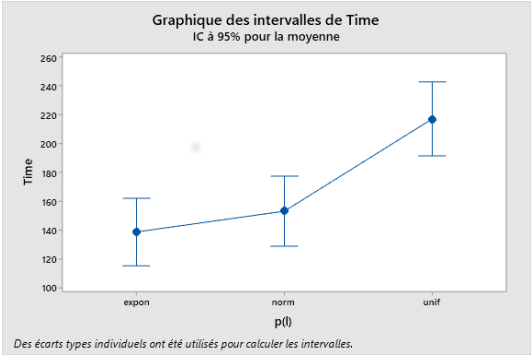


FIGURE 4.15 – Impact de $p(l)$ sur $Time$

$p(h, o_{jk})$ détériore les performances du processus d'évaluation en nécessitant plus de temps de résolution et en générant un nombre de simulations plus élevé.

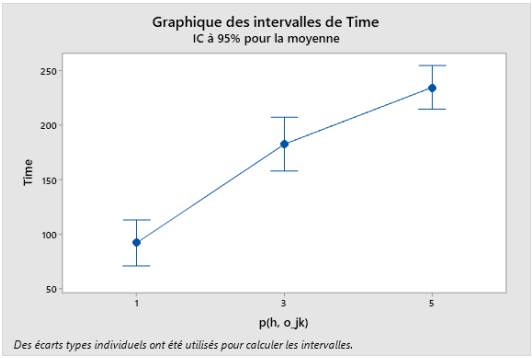


FIGURE 4.16 – Impact de $p(h, o_{jk})$ sur $Time$

4.3.3.8 Sensibilité à la deadline $\tilde{d}$

La deadline $\tilde{d}$ est un facteur ayant un effet apparent sur les performances du processus. En effet, les graphiques des effets principaux (Figure 4.17) permettent de déduire que l'effet de $\tilde{d}$ est statistiquement significatif sur les performances du processus (confirmé par l'analyse de la variance de la figure 4.18).

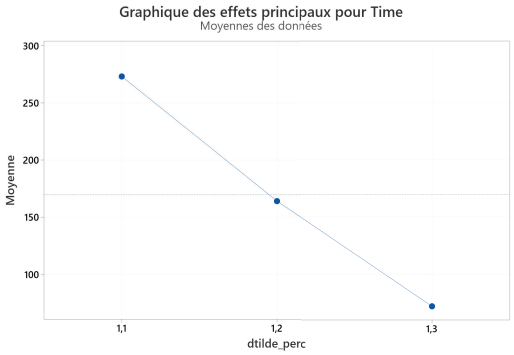


FIGURE 4.17 – Effets principaux pour les paramètres de la deadline

Analyse de la variance

Source	DL	SomCar ajust	CM ajust	Valeur F	Valeur de p
dtilde_perc	2	1642270	821135	124,50	0,000
Erreur	240	1582905	6595		
Total	242	3225175			

FIGURE 4.18 – Analyse de la variance en fonction de la deadline

4.3.3.9 Sensibilité aux paramètres d'entrée : synthèse

À l'issue de l'analyse effectuée sur les paramètres du problème d'évaluation, nous pouvons déduire différentes informations essentielles sur les performances du processus d'évaluation. Le premier étant l'interaction entre les différentes performances. En effet, nous avons pu montrer que l'évolution de $Time$ par rapport à $NbSim$ est linéaire.

La deuxième conclusion qui émane de l'analyse effectuée est l'impact de la règle d'ordonnancement sur les performances du processus. En effet, nous avons constaté que ce facteur n'a pas d'influence sur le temps de résolution La troisième remarque effectuée suite à cette analyse concerne le type de scénario de perturbations considéré. Les scénarios étudiés sont progressifs allants du traitement d'une seule perturbation à la combinaison de plusieurs types de perturbations (Incertitudes et Aléas). En augmentant le nombre de perturbations (scénario $Pert_3$ défini dans le tableau 4.5), le temps de résolution et le nombre de simulations sont amenés à augmenter. Le décideur doit donc prendre en considération que pour traiter une combinaison de plusieurs perturbations, les performances du processus d'évaluation risquent de se détériorer.

L'analyse des facteurs liés aux paramètres de perturbations permet de conclure que pour les incertitudes, la variabilité de la durée δ n'a pas d'effet statistiquement significatif sur les performances de l'approche, en revanche la distribution de probabilité nécessaire pour le calcul de $p(l)$ a un effet sur les performances du processus. Cet effet est lié à la discrétisation de la distribution et au nombre d'itérations que génère cette dernière. Pour l'aléa, la probabilité d'occurrence a un effet considérable sur les performances du processus. En effet, plus la valeur de cette probabilité augmente plus $Time$ et $NbSim$ augmentent. Cette détérioration est expliquée par l'augmentation du nombre d'opérations impactées par l'aléa. En effet, en augmentant $p(h, o_{jk})$, le nombre d'opérations pouvant être affectées par l'aléa augmente.

La dernière conclusion concerne la deadline qui a un impact sur les performance de résolution. La table 4.9 permet de faire une synthèse des effets qui ont été identifiés précédemment.

s	Scénario	δd_{jkr}^{ref}	$p(l)$	$\delta d_{jkr}^{ref,h}$	$p(h, o_{jk})$	$\tilde{d}$
non	oui	oui	oui	oui	non	oui

TABLE 4.9 – Synthèse des effets des facteurs

4.4 Passage à l'échelle

La question du passage à l'échelle permet de définir les limites de l'approche en ce qui concerne la taille du problème qui peut être traitée par notre approche. En effet, répondre à la question du

passage à l'échelle permet de déterminer les limites de performances de l'approche afin d'évaluer son applicabilité dans un environnement réel pour traiter des problèmes industriels.

4.4.1 Construction du plan d'expériences

Pour analyser le passage à l'échelle, l'expérimentation consiste à appliquer le processus d'évaluation sur des ordonnancements provenant d'ateliers de production de tailles différentes. Pour construire un panel d'expérimentation qui englobe un maximum de configuration de taille, nous avons choisi d'utiliser l'étude de cas illustrée dans le chapitre 3 de [Giard, 2003].

L'échelle d'un atelier de production peut être mesurée par les caractéristiques suivantes : le nombre de ressources de l'atelier (NbR), le nombre de jobs (NbJ) à exécuter, et le nombre d'opérations pour chaque job j ($NbOp^j$). Les niveaux de chacune de ces caractéristiques sont définis à partir de l'atelier de production utilisé pour l'illustration de la faisabilité du processus d'évaluation [Giard, 2003]. Le niveau 1 d'une caractéristique correspond à celui de l'atelier adapté de [Giard, 2003] et utilisé dans le chapitre 3. Le niveau 2 d'une caractéristique consiste à doubler la valeur définie dans le niveau 1 (on double les ressources, on double les jobs ...). Le niveau 3 consiste à doubler la valeur définie dans le niveau 2. Voir la Table 4.10 pour le détail de ces niveaux. On peut noter ici que le nombre d'opérations total à exécuter $NbOp$ varie entre 35 et 560.

	NbJ	$NbOp^j$	NbR
Niveaux des facteurs	8	1([3; 7])	7
	16	2([6; 14])	14
	32	3([12, 28])	28

TABLE 4.10 – Facteurs liés à la taille de l'atelier de production

4.4.2 Exécution du plan d'expérience

Afin de mener à bien l'expérimentation pour répondre à la question du passage à l'échelle, nous fixons la valeur de $\tilde{d}_{perc}$ de telle sorte qu'elle permette d'obtenir un niveau de service supérieur ou égal à 80%.

Les autres paramètres prennent leur valeur dans les mêmes conditions que pour l'analyse de la sensibilité aux paramètres d'entrée.

4.4.3 Analyse des résultats du passage à l'échelle

L'exécution du plan d'expérience permet d'obtenir un fichier de résultats contenant, pour chaque expérience, le temps de résolution du processus ($Time$) et le nombre de simulations effectuées par le model checker ($NbSim$) .

Les résultats de l'expérimentation permettent de déterminer le temps de résolution et le nombre de simulation en fonction de la taille de l'atelier. La figure 4.19 permet de visualiser la répartition du temps de résolution pour ce plan d'expérience. Nous remarquons ainsi que la valeur médiane se situe à moins de 10 minutes (561 secondes). Cela signifie que la moitié des expériences est résolue en moins de 10 minutes. De plus 75 de la population est résolu en moins de 2500 secondes, ce qui reste inférieur à l'heure. Les points extrêmes (nécessitant une grande durée) sont considérés comme des points aberrants.

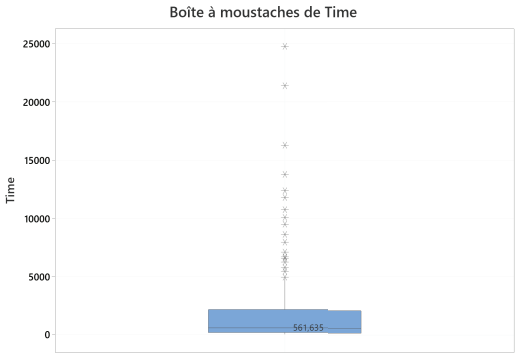


FIGURE 4.19 – Répartition du temps de résolution

Dans ce qui suit, les trois facteurs principaux (NbJ, $NbOp^j$ et NbR) sont analysés. Ces trois facteurs ont un effet sur les performances du processus d'évaluation. Les graphes des effets principaux permettent de voir que les trois facteurs ont un impact visible sur le temps de résolution $Time$ (ce qui est confirmé par l'analyse de la variance associée figure 4.21).

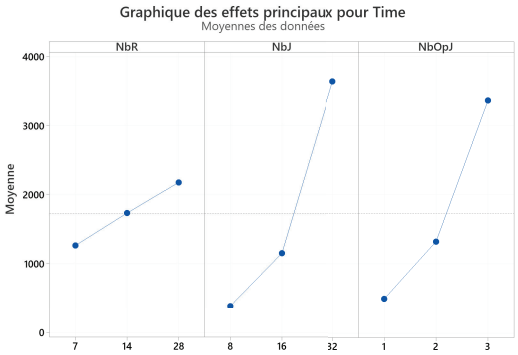


FIGURE 4.20 – Effets de la taille de l'atelier

Analyse de la variance

Source	DL	SomCar ajust	CM ajust	Valeur F	Valeur de p
NbR	2	33748515	16874257	2,93	0,055
NbJ	2	467052457	233526229	40,56	0,000
NbOpJ	2	352918897	176459448	30,65	0,000
NbR*NbJ	4	20792275	5198069	0,90	0,463
NbR*NbOpJ	4	20206158	5051540	0,88	0,478
NbJ*NbOpJ	4	186169309	46542327	8,08	0,000
Erreur	224	1289537067	5756862		
Inadéquation de l'ajustement	8	23501275	2937659	0,50	0,855
Erreur pure	216	1266035792	5861277		
Total	242	2370424679			

FIGURE 4.21 – Analyse de la variance en fonction de la taille de l'atelier

Dans un premier temps, la remarque générale est que lorsque la taille de l'atelier augmente, le temps de résolution et le nombre de simulations augmentent. Pour comprendre l'impact de la taille de l'atelier sur les performances de l'approche, nous définissons un nouveau paramètre : la charge potentielle de l'atelier. Celui-ci est défini comme le rapport entre le nombre d'opérations à exécuter dans l'atelier ($NbOp$) et le nombre de ressources disponibles (NbR).

Le ratio $NbOp/NbR$ impacte considérablement le temps de résolution. En effet le graphique

des effets de la Figure 4.22 suggère que le temps de résolution augmente exponentiellement avec la charge de l'atelier. L'analyse de la variance de la figure 4.23 permet de vérifier que ce facteur a un effet important.

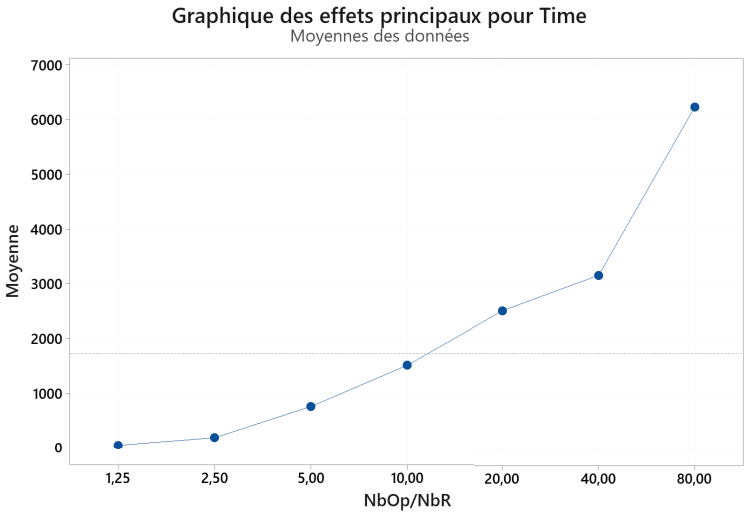


FIGURE 4.22 – Effets du ratio $NbOp/NbR$ sur $Time$

Analyse de la variance

Source	DL	SomCar ajust	CM ajust	Valeur F	Valeur de p
NbOp/NbR	6	413160956	68860159	8,30	0,000
Erreur	236	1957263723	8293490		
Total	242	2370424679			

FIGURE 4.23 – Analyse de la variance pour $NbOp/NbR$

4.4.4 Passage à l'échelle : synthèse

Pour la question du passage à l'échelle, nous nous sommes intéressés aux effets des facteurs liés aux paramètres d'entrée de l'atelier de production lié à l'ordonnancement s.
Les performances analysées sont le temps de résolution du processus et le nombre de simulations.

Au vu des analyses effectuées précédemment, une première conclusion globale serait que l'approche est fortement sensible à la taille de l'atelier de production. L'évolution du temps de résolution est exponentielle en fonction de la taille de l'atelier impliquant un grand nombre de jobs NbJ, un grand nombre d'opérations par job $NbOP^j$ et un grand nombre de ressources NbR. En dépit de cette évolution, le temps de résolution maximal se rapprochant de 15 heures est celui de l'expérience ayant un maximum de nombre de jobs et d'opérations par job, à savoir 560 opérations au total, devant être exécutées sur un minimum de ressources, à savoir 7 ressources. Ce résultat implique que le processus d'évaluation arrive à calculer le niveau de service pour une construction d'atelier pessimiste (charge maximale des ressources).
À partir de cette conclusion, la réponse à la question du passage à l'échelle est apportée. En

effet, le processus d'évaluation développé permet de traiter des tailles importantes d'atelier de production.

La distribution du temps de résolution est concentrée sur la moyenne de 50 minutes (comme indiqué dans la figure 4.19), ce qui implique que le temps maximal constaté n'est pas représentatif de la majorité des expériences mais plutôt des expériences à configurations pessimistes. La distribution du nombre de simulations est plus étendue sur l'intervalle de l'histogramme. En dépit de cette constatation, la majorité des expériences génèrent un nombre de simulations autour de la moyenne 8252 simulations.

4.5 Conclusion

L'objectif de chapitre était d'évaluer les performances du processus d'évaluation présenté au chapitre 3 selon trois points de vue : généricité, sensibilité et passage à l'échelle. Nous avons ainsi pu démontrer que le processus d'évaluation est générique vis-à-vis du type d'atelier, du type de perturbations et des combinaisons de perturbations. De plus, des expérimentations basées sur des plans d'expériences ont permis d'évaluer la sensibilité du processus aux paramètres d'entrée. Cela a permis d'identifier les paramètres critiques vis-à-vis des performances du processus (temps de résolution et nombre de simulations). Enfin un autre plan d'expériences a permis d'évaluer la capacité du processus d'évaluation à supporter un passage à l'échelle concernant la taille de l'atelier en entrée du processus. Nous avons pu démontrer que le temps de résolution augmente de manière exponentielle avec la taille de l'atelier mais surtout avec la charge moyenne de l'atelier. Cependant les temps de résolution même pour des problèmes de très grande taille (32 jobs, 560 opérations et 32 ressources) restent acceptable dès lors que l'on est dans une évaluation a priori. Cela permet donc d'envisager l'utilisation de l'approche d'évaluation de la robustesse dans le cadre d'un processus d'aide à la décision pour traiter des cas de taille industrielle.

Chapitre 5

Du processus d'évaluation vers l'aide à la décision

Sommaire

5.1 Introduction 105
5.2 Ordonnancement robuste sur une nouvelle ligne de production . . . 106
5.2.1 Choix de l'ordonnancement permettant d'absorber la plus grande déviation sur les durées opératoires 108
5.2.2 Détermination de l'ordonnancement le plus robuste 108
5.2.3 Détermination d'une date de livraison malgré les perturbations 109
5.2.4 Identification de la ressource critique à surveiller 110
5.3 Optimisation robuste pour l'atelier d'emballage 110
5.4 Conclusion 114

5.1 Introduction

Le processus d'évaluation proposé dans le cadre de cette thèse permet d'évaluer la robustesse d'un ordonnancement déterministe s soumis à des perturbations $Pert$ et respectant une deadline $\tilde{d}$. Il permet au décideur d'évaluer le niveau de service $RL(s, Pert, \tilde{d})$. À travers ce calcul, plusieurs informations peuvent être exploitées par le décideur dans un cadre industriel. Dans un cadre industriel, l'objectif est de satisfaire la commande du client tout en respectant ses besoins. L'ordonnancement permet d'organiser le fonctionnement de l'atelier de production. Les risques de retard dûs aux perturbations dans un horizon d'ordonnancement peuvent ainsi impacter la qualité de la commande client. L'objectif de ce chapitre est d'explorer l'utilisation du processus d'évaluation dans le cadre d'un scénario industriel. Le scénario présenté permet d'illustrer des situations de décision pouvant se produire dans un atelier de production. Pour résoudre ces situations, plusieurs problèmes de robustesse identifiés dans le chapitre 2 peuvent être utilisés. La finalité de ces problèmes est d'obtenir des informations précises afin d'aider le décideur dans l'atelier de production.

Le scénario présenté est celui d'une entreprise composée d'un atelier de production chargé de l'assemblage de produits et d'un atelier d'emballage permettant de finaliser la commande du client (Figure 5.1). Dans l'atelier d'assemblage deux lignes de production pré-existantes assurent la production de produits commercialisés par l'entreprise. Dans une logique de développement, l'entreprise décide d'ouvrir une troisième ligne d'assemblage pour prendre en charge de nouvelles

commandes client. La ligne de production est de type Flow Shop avec deux ressources et 3 nouveaux jobs (représentant 3 nouveaux produits). Le problème d'ordonnancement lié à cette application est notée dans la littérature $F2||C_{max}$ (notations expliquées dans l'Annexe B) (F pour flow shop, 2 pour le nombre de ressources et C_{max} pour le critère d'optimisation le Makespan). L'atelier d'emballage est un atelier de deux machines parallèles non liées (chaque machine est différente). Le problème d'ordonnancement correspondant est noté $R2||C_{max}$ (R pour machines parallèles non liées, 2 pour le nombre de machines et C_{max} pour le critère d'optimisation).

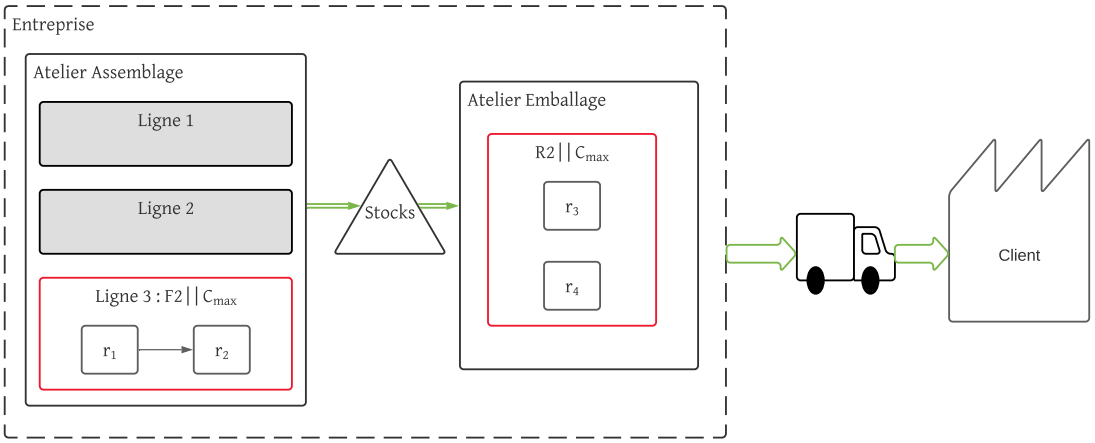


FIGURE 5.1 – Illustration du scénario industriel

5.2 Ordonnancement robuste sur une nouvelle ligne de production

Nous nous intéressons tout d'abord à la nouvelle ligne d'assemblage. Cette ligne Flow Shop contient deux ressources r_1 et r_2. L'ensemble de jobs considéré J contient 3 jobs et chaque job est défini par une paire d'opérations o_{jk} appartenant à l'ensemble $O_j^J = \{o_{j1}, o_{j2}\}$. Chaque opération doit être exécutée sur les ressources r en respectant une durée d'exécution de référence d_{jkr}^{ref} établie par le service de conception des produits. Les durées d'exécution de référence sont inscrites dans la Table 5.1.

Job	Opération	d_{jkr}^{ref} $(heures)$
1	o_{11}	12
1	o_{12}	12
2	o_{21}	4
2	o_{22}	16
3	o_{31}	20
3	o_{32}	8

TABLE 5.1 – Données de l'atelier Flow Shop

À partir de ces données du problème, six ordonnancements déterministes (correspondant aux

6 permutations possibles dans l'ensemble des 3 jobs) peuvent être déterminés (Figure 5.2). Parmi ces ordonnancements, l'ordonnancement optimal pour le problème déterministe correspond à l'ordonnancement s_3 qui a un makespan de 44 heures.

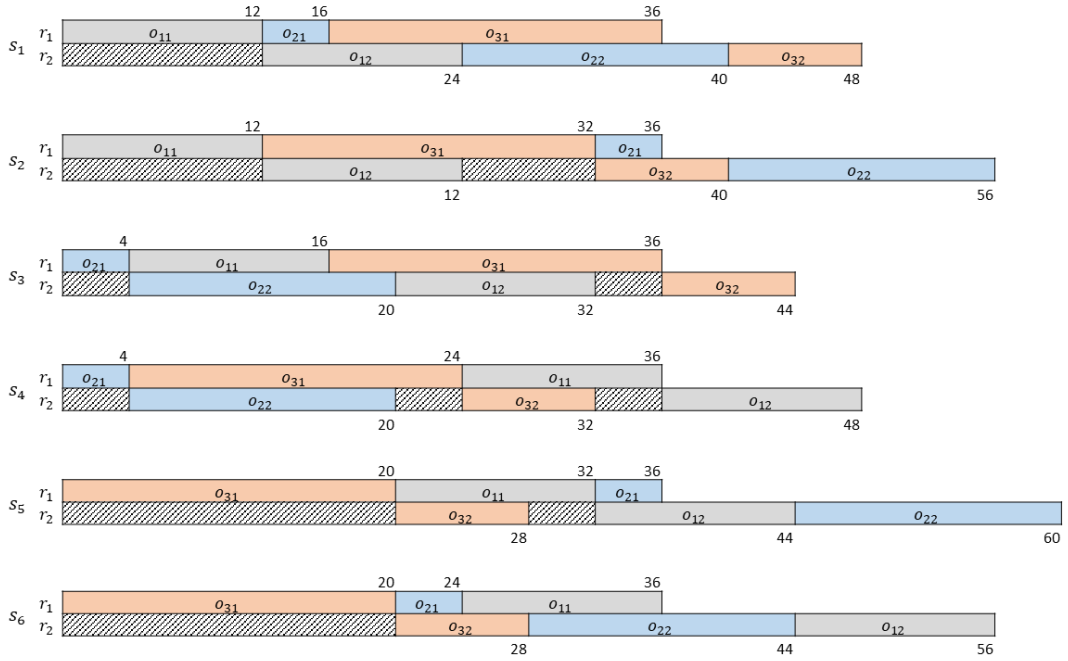


FIGURE 5.2 – Ordonnancements déterministes générés à partir de $F2||C_{max}$

Dans ce chapitre, quatre scénarios d'usage permettent d'exploiter le processus d'évaluation. Chacun de ces scénarios représente une situation de décision pour le décideur. Nous allons montrer comment la bonne utilisation du processus d'évaluation permet d'apporter des informations pertinentes pour le décideur afin de l'aider à faire le bon choix. Les scénario sont décrits de la façon suivante :

1. À la mise en service de la nouvelle ligne de production, le décideur ne dispose pas d'un historique de données permettant de modéliser de manière fiable les incertitudes sur les durées opératoires et notamment leur impact sur la variation par rapport aux durées de référence. Le décideur souhaite choisir parmi des ordonnancements d'entrée celui qui permet d'absorber la plus grande déviation sur la durée de référence.
2. Après six mois de service, un historique de données de production permet maintenant d'avoir des informations sur les durées opératoires réalisées. Un modèle fiable des incertitudes sur les durées opératoires peut donc être obtenu. Le décideur souhaite choisir maintenant l'ordonnancement avec le meilleur comportement face à ces perturbations.
3. Les produits fabriqués sur la ligne de production sont ensuite conditionnés par l'atelier d'emballage. Le décideur souhaite fournir au responsable de l'atelier d'emballage une date de livraison réaliste de ces produits malgré les perturbations sur l'ordonnancement qu'il a choisi d'exécuter afin que celui-ci puisse planifier sa production.
4. Par ailleurs le décideur souhaite planifier la maintenance à réaliser sur sa ligne de production et notamment identifier la machine critique à surveiller. L'objectif pour lui est d'identifier la machine (connaissant sa probabilité de panne) qui impacte le plus la capacité de l'ordonnancement exécuté à satisfaire la durée prévisionnelle.

Ces quatre scenario sont formalisés dans le tableau 5.6, et les réponses remontées au décideur pour chaque problème y sont données également.

5.2.1 Choix de l'ordonnancement permettant d'absorber la plus grande déviation sur les durées opératoires

Lors de la mise en place d'une nouvelle ligne de production, il peut être intéressant d'avoir une visibilité sur les perturbations pouvant être absorbées par un ordonnancement afin de choisir celui à exécuter. En effet, pour être certains de faire face à un maximum d'imprévus en attendant d'avoir un retour d'expérience sur la ligne, le décideur va choisir un ordonnancement pouvant absorber un maximum de perturbations, même si sa durée n'est pas la meilleure. Le décideur s'intéresse ici aux incertitudes pouvant impacter les durées d'exécution des opérations. L'objectif d'identifier et d'implémenter l'ordonnancement permettant d'absorber un ensemble maximal de perturbations $\overline{Pert}$ tout en satisfaisant un seuil de robustesse défini RL_{lim} et une deadline $\tilde{d}$. Ainsi l'obtention de cet ensemble maximal de perturbations revient à exécuter une analyse de la stabilité ($EV4$) définie dans le chapitre 2).

Dans le cas de ce scénario, le décideur fourni les paramètres d'entrée du problème avec les spécifications suivantes :

- La deadline à prendre en compte est fixée à $\tilde{d} = 51\ h$, ce qui correspond à une journée de production supplémentaire de 7 heures sur le C_{max} optimal (défini par s_3 et égal à $44h$).
- L'analyse de la stabilité est effectuée sur les ordonnancement ayant un C_{max}^{ref} inférieur à 51, à savoir s_1, s_3 et s_4 (voir Figure 5.2).
- Le niveau de service minimum est fixé à $RL_{lim} = 95\%$
- La perturbation à analyser, ici l'incertitude sur les durées d'exécution des opérations $Pert = \{u_1^{ex}\}$ est définie comme $d_{jkr} = d_{jkr}^{ref} + \delta d_{jkr}^{U_1^{ex}}$ avec $\delta d_{jkr}^{U_1^{ex}} \in \left[-\delta \times d_{jkr}^{ref}\ ;\ +\delta \times d_{jkr}^{ref}\right]$. Avec δ qui représente un pourcentage de déviation induit par l'incertitude et qui est identique à toutes les opérations.

Ainsi $Pert$ peut être mesuré par δ et trouver $\overline{Pert}$ revient à chercher la plus grande valeur de δ. Suite à l'application de l'algorithme $EV4$ avec ces paramètres d'entrée, le décideur récupère cette valeur pour chaque ordonnancement (voir table 5.2). L'ordonnancement s_3 absorbe une déviation de 37% alors que les 2 autres ordonnancements absorbe seulement 12%, pour une même deadline et un même niveau de service demandé. Le décideur prend évidemment la décision d'appliquer l'ordonnancement s_3.

$\tilde{d}$	Ordonnancement	δ
$\tilde{d} = 51$	s_1	12%
	s_3	37%
	s_4	12%

TABLE 5.2 – Analyse de la stabilité des ordonnancements

5.2.2 Détermination de l'ordonnancement le plus robuste

Après plusieurs mois de service des données ont été historiées sur les temps d'exécutions des opérations permettant de construire un modèle fiable des perturbations. À partir de ce retour d'expérience, le décideur souhaite éventuellement remettre en cause le choix de s_3 afin d'exécuter l'ordonnancement le plus robuste face aux perturbations réelles. Dans ce cas, le décideur souhaite

connaître le niveau de service de tous les ordonnancements déterministes (Figure 5.2) avec les perturbations observées $Pert$ afin de choisir le meilleur.

Pour analyser le niveau de service ($EV1$ du chapitre 2, le décideur fournit les éléments suivants :

— L'ensemble de perturbations $Pert = \{u_1^{ex}\}$ (modélisée comme dans la section précédente) avec une fluctuation δ égale à $+/-20\%$ (i.e. $d_{jkr} = d_{jkr}^{ref} + / - 20\% \times d_{jkr}^{ref}$).
— La deadline est fixée à 110% par rapport la durée totale de l'ordonnancement déterministe (i.e. $\tilde{d} = 110\% * C_{max}^{ref}(s)$).

Ordonnancement	C_{max}^{ref}	$\tilde{d}$	RL
s_1	48	53	97%
s_2	56	61	95%
s_3	44	48	93%
s_4	48	53	97%
s_5	60	66	97%
s_6	56	61	94%

TABLE 5.3 – Résultats de l'évaluation du niveau de service pour 6 ordonnancements

Suite à l'analyse du niveau de service, le décideur obtient pour chaque ordonnancement les niveaux de service résumés dans le tableau 5.3. Trois ordonnancements { s_2, s_3, s_6} sont rejetés par le décideur car leur niveau de service est inférieur ou égal à 95%. Pour les 3 autres { s_1, s_4, s_5} ayant un niveau de service équivalent à 97%, le décideur pourra choisir celui dont le Makespan C_{max}^{ref} est le plus faible, ici $48UT$ afin d'avoir un bon compromis entre la durée totale de l'ordonnancement et son niveau de service. Ici s_1 et s_4 ont le même compromis. Cependant s_4 permet de mieux gérer les encours de production (les dates de fin des jobs pris un par un sont meilleures). Le décideur choisit donc d'exécuter s_4.

5.2.3 Détermination d'une date de livraison malgré les perturbations

Suite à la détermination de l'ordonnancement robuste (s_4), le décideur doit informer l'atelier d'emballage de la date de livraison des produits. Cette date de livraison doit être la plus juste possible pour éviter de créer du stock en amont de l'atelier d'emballage ou à l'inverse de créer une pénurie de travail dans celui-ci. Pour l'ordonnancement s_4, le décideur souhaite savoir quelle deadline $\underline{\tilde{d}}$ il peut promettre au responsable de l'atelier d'emballage malgré les perturbations. Afin de déterminer cette deadline, il suffit d'effectuer une analyse de la sensibilité identifié dans le chapitre 2 par le problème d'évaluation EV_3.

Les paramètres d'entrée du problème sont donc :

— L'ordonnancement s_4 (Figure 5.2).
— La perturbation : l'incertitude sur les durées d'exécution des opérations avec les mêmes paramètres que précédemment ($Pert = \{u_1^{ex}\}$ et $d_{jkr} = d_{jkr}^{ref} + / - 20\% \times d_{jkr}^{ref}$).
— Le niveau de service à garantir $RL_{lim} = 90\%$.

L'application du problème EV_3 permet de trouver que $\underline{\tilde{d}} = 53$ pour s_4.C'est la deadline minimale qui malgré les perturbations permet de respecter le niveau de service RL_{lim}.

5.2.4 Identification de la ressource critique à surveiller

Toujours pour l'ordonnancement s_4, le décideur souhaite identifier la machine critique à surveiller afin d'aider à fixer la maintenance préventive. Pour cela, le processus d'évaluation est utilisé en ayant les paramètres d'entrée suivants :

— L'ordonnancement d'entrée s_4,
— La perturbation prise en compte : panne sur une ressource $Pert = \{h_1\}$,
— Probabilité d'occurrence de h_1 : $p(h_1, o_{jk}) = 1/3$ pour la machine tombant en panne
— le temps de réparation : $d_{jkr}^{ref,h_1} = 3\ h$
— La deadline considérée : $\tilde{d} = 53$

Les résultats obtenus dans le tableau 5.4 sont obtenus en fonction de la ressource pouvant subir une panne. Si c'est la ressource r_1 qui peut tomber en panne on obtient $RL(r_1)$ et si c'est la ressource r_2, on obtient $RL(r_2)$. On voit bien ici que lorsque la ressource r_2 est susceptible de tomber en panne le niveau de service associé à l'ordonnancement s_4 chute à 75% par rapport à la ressource r_1. Il conviendrait ainsi de réfléchir à une maintenance préventive sur cette machine afin de réduire sa probabilité de défaillance.

Ordonnancement	Criticité ressource	
	$RL(r_1)$	$RL(r_2)$
s_4	96%	75%

TABLE 5.4 – Identification de la ressource critique

5.3 Optimisation robuste pour l'atelier d'emballage

Dans l'atelier d'emballage (Figure 5.1), les produits provenant de l'atelier de production sont conditionnés sur deux ressources parallèles r_3 et r_4. Un problème de génération de la solution d'ordonnancement se pose. Pour le décideur, il est question de proposer une solution d'ordonnancement optimisant le Makespan C_{max}. L'optimisation de cet atelier n'est pas la seule préoccupation du décideur, en effet la position de cette ligne la rend sensible aux perturbations . L'emballage étant la dernière action sur le produit avant la finalisation de la commande client, toute perturbation peut engendrer des retards et nuire au bon fonctionnement de l'entreprise. La robustesse de l'ordonnancement implémentée sur cette ligne est donc un critère à prendre en compte. Pour combiner l'optimisation et l'analyse du niveau de service, le décideur cherche à trouver l'ordonnancement optimal qui garantie un niveau de service fixé.

Parmi les problèmes de robustesse définis dans le chapitre 2, le premier problème d'optimisation correspond au besoin du décideur. En effet, les problèmes d'optimisation permettent de trouver une solution d'ordonnancement qui respectent la performance de la robustesse et optimisent le critère du Makespan.
Chaque problème d'optimisation permet de faire appel au problème d'évaluation. Dans le cadre de cette thèse, nous considérons que pour résoudre le problème d'optimisation, l'algorithme doit faire appel à une étape d'évaluation. À titre d'exemple, le problème de l'optimisation de la solution a pour objectif de trouver une solution d'ordonnancement ayant, non seulement, un Makespan optimal, mais également qui satisfait un seuil de niveau de service donné (voir OP_1 dans Chapitre 2).

Au cours de ces travaux de thèse, une collaboration entre différents groupes de travail du GDR MACS a été lancée. Plus précisément, une collaboration entre le groupe de travail spécialisé SED (GT SED) et le groupe de travail spécialisé en ordonnancement (GT BERMUDES) a été initiée. Au cours de ces travaux communs, une approche hybride a été proposée. Elle consiste à combiner une approche d'optimisation RO et l'approche que nous proposons pour l'évaluation de la robustesse. Cette collaboration a aboutit à un chapitre d'ouvrage [Marangé et al., 2020].

L'approche de résolution du problème d'optimisation OP_1 repose sur le couplage des modèles de recherche opérationnelle souvent utilisés dans l'optimisation robuste [Bertsimas and Sim, 2004] avec le processus d'évaluation reposant sur les modèles de systèmes à événements discrets.

Ici, le problème d'ordonnancement consiste à trouver l'affectation de N jobs (avec une seule opération d'emballage) sur l'une ou l'autre des deux machines parallèles. Le programme mathématique permettant de résoudre le problème $R2||C_{max}$ sans perturbations s'écrit sous la forme suivante :

$$\begin{cases} \textit{Minimise } C_{max} & \\ s.t. & \\ \sum_{r=1}^{2} s_{jr} = 1 & \forall j \in \{1,\dots,N\} \\ \sum_{j=1}^{N} d_{jr} s_{jr} \leq C_{max} & \forall r \in \{1,2\} \\ C_{max} \geq 0 & \\ s_{jr} \in \{0,1\} & \forall (j,k) \in \{1,\dots,N\} \times \{1,2\} \end{cases}$$

d_{jr} est la durée du job j sur la machine r (il n'y a qu'une seule opération donc l'indice k n'est pas utile ici). s_{jr} est une variable binaire qui vaut 1 si le job j est affectée à la machine r et 0 sinon. La première contrainte permet de vérifier que le job est exécuté sur une et une seule machine. La deuxième contrainte permet de vérifier que la durée totale de l'ordonnancement (C_{max}) est supérieure ou égale à la durée de réalisation de chaque machine. Les deux dernières contraintes permettent de fixer le domaine de variation des variables.

Nous considérons maintenant que la durée d_{jr} est incertaine ($d_{jr} = d_{jr}^{ref} + \delta d_{jr}$) et qu'il existe une variable aléatoire ξ_{jr} qui prend ses valeurs dans $[-1\ ;\ 1]$, telle que :

$$\delta d_{jr} = \xi_{jr} \Delta d_{jr}$$

avec Δd_{jr} connu.

La prise en compte de ces incertitudes dans le modèle initial revient au programme mathématique suivant :

$$\begin{cases} \textit{Minimise } C_{max} & \\ s.t. & \\ \sum_{r=1}^{2} s_{jr} = 1 & \forall j \in \{1,\dots,N\} \\ \sum_{j=1}^{N} d_{jr}^{ref} s_{jr} + \sum_{j=1}^{N} \xi_{jr} \Delta d_{jr} s_{jr} \leq C_{max} & \forall r \in \{1,2\} \\ C_{max} \geq 0 & \\ s_{jr} \in \{0,1\} & \forall (j,k) \in \{1,\dots,N\} \times \{1,2\} \end{cases}$$

Afin de résoudre ce type de problème, l'approche de [Bertsimas and Sim, 2004] consiste à construire des solutions capables de résister raisonnablement à l'incertitude des paramètres tout en garantissant un degré de protection (RL_{lim}) voulu par le décideur. Elle s'oppose à l'approche de [Kouvelis and Yu, 1997] ou à d'autres approches très conservatives qui considéreraient uniquement le pire cas (c'est-à-dire résoudre le programme précédent en fixant xi_{jr} à 1 pour toutes les durées).

Appliquer l'approche de [Bertsimas and Sim, 2004] à notre problème revient à fixer pour chaque ressource r un paramètre Ω_r de telle sorte que l'équation (5.1) soit vérifiée.

$$P\left(\sum_{j=1}^{N} \xi_{jr}\Delta d_{jr}s_{jr} \leq \max_{\sum_{j=1}^{N}|\xi_{jr}|\leq\Omega_r}\left(\sum_{j=1}^{N} \xi_{jr}\Delta d_{jr}s_{jr}\right)\right) \geq RL^i_{lim} \tag{5.1}$$

Puis à résoudre le programme mathématique défini par l'équation (5.2).

$$\begin{cases} \textit{Minimise } C_{max} & \\ s.t. & \\ \sum\limits_{r=1}^{2} s_{jr} = 1 & \forall j \in \{1,\ldots,N\} \\ \sum\limits_{j=1}^{N} d^{ref}_{jr}s_{jr} + \max\limits_{\sum_{j=1}^{N}|\xi_{jr}|\leq\Omega_r}\left(\sum\limits_{j=1}^{N} \xi_{jr}\Delta d_{jr}s_{jr}\right) \leq C_{max} & \forall r \in \{1,2\} \\ C_{max} \geq 0 & \\ s_{jr} \in \{0,1\} & \forall (j,k) \in \{1,\ldots,N\}\times\{1,2\} \end{cases} \tag{5.2}$$

En pratique, fixer les valeurs de Ω_r pour que l'équation 5.1 soit satisfaite n'est pas un problème facile. [Bertsimas and Sim, 2004] ont démontré que si les variables aléatoires ξ_{jr} sont indépendantes et symétriques alors les valeurs de Ω_r peuvent être déterminées analytiquement. Une telle hypothèse est difficilement soutenable dans un contexte industriel. Nous avons montré dans [Marangé et al., 2020] comment, les outils développés dans le chapitre 3 peuvent être utilisés afin de vérifier pour un couple (Ω_1, Ω_2) donné et une solution s trouvée par le programme linéaire (5.2), si l'équation (5.1) est satisfaite ou non. Ainsi de manière itérative, il est possible de tendre vers une solution s qui : (i) minimise le C_{max} tout en (ii) garantissant un niveau de service fixé par le décideur. Cela permet donc de résoudre le problème OP_1.

La méthode de résolution proposée repose sur l'itération des trois modules suivants (Figure 5.3).

1. Module de recherche opérationnelle (RO) : à partir de l'initialisation des paramètres de l'atelier de production et des paramètres de perturbations, un modèle MILP robuste est formulé en fixant la valeur du couple (Ω_1, Ω_2). Pour la première itération, ce couple est fixé à $(0, 0)$ (cela revient à étudier le problème déterministe). Le modèle est ensuite introduit dans un solveur pour obtenir la solution d'ordonnancement optimale prenant en compte les paramètres de robustesse. La solution s générée est ensuite envoyée vers le deuxième module.
2. Module des systèmes à événements discrets (SED) : Compte tenu du systèmes de production et de la solution s proposée par le module RO, le processus d'évaluation peut être paramétré et la deadline $\tilde{d}$ peut être identifiée. En effet, la solution s, les paramètres de perturbations $Pert$ ainsi que la deadline souhaitée sont renseignés à l'entrée du processus d'évaluation.

L'ensemble de perturbations $Pert$ est défini suivant les données de l'atelier et la deadline $\tilde{d}$ est dans ce cas égal au Makespan de la solution ($\tilde{d} = C_{max}^{ref}$). Suite à l'exécution du processus d'évaluation, le niveau de service $RL(s, Pert, \tilde{d})$ est récolté.

3. Module de test de robustesse : En fonction du niveau de service obtenu et de la comparaison effectuée avec le seuil défini RL_{lim}, (i) si le seuil identifié par le décideur est atteint, alors le processus est arrêté et la solution s générée est retenue. La solution s est alors qualifiée de robuste ; (ii) Si le niveau de service n'atteint pas le seuil fixé par le décideur, les paramètres de robustesse (Ω_r) sont mis à jour et envoyés au premier module (RO) pour une nouvelle itération.

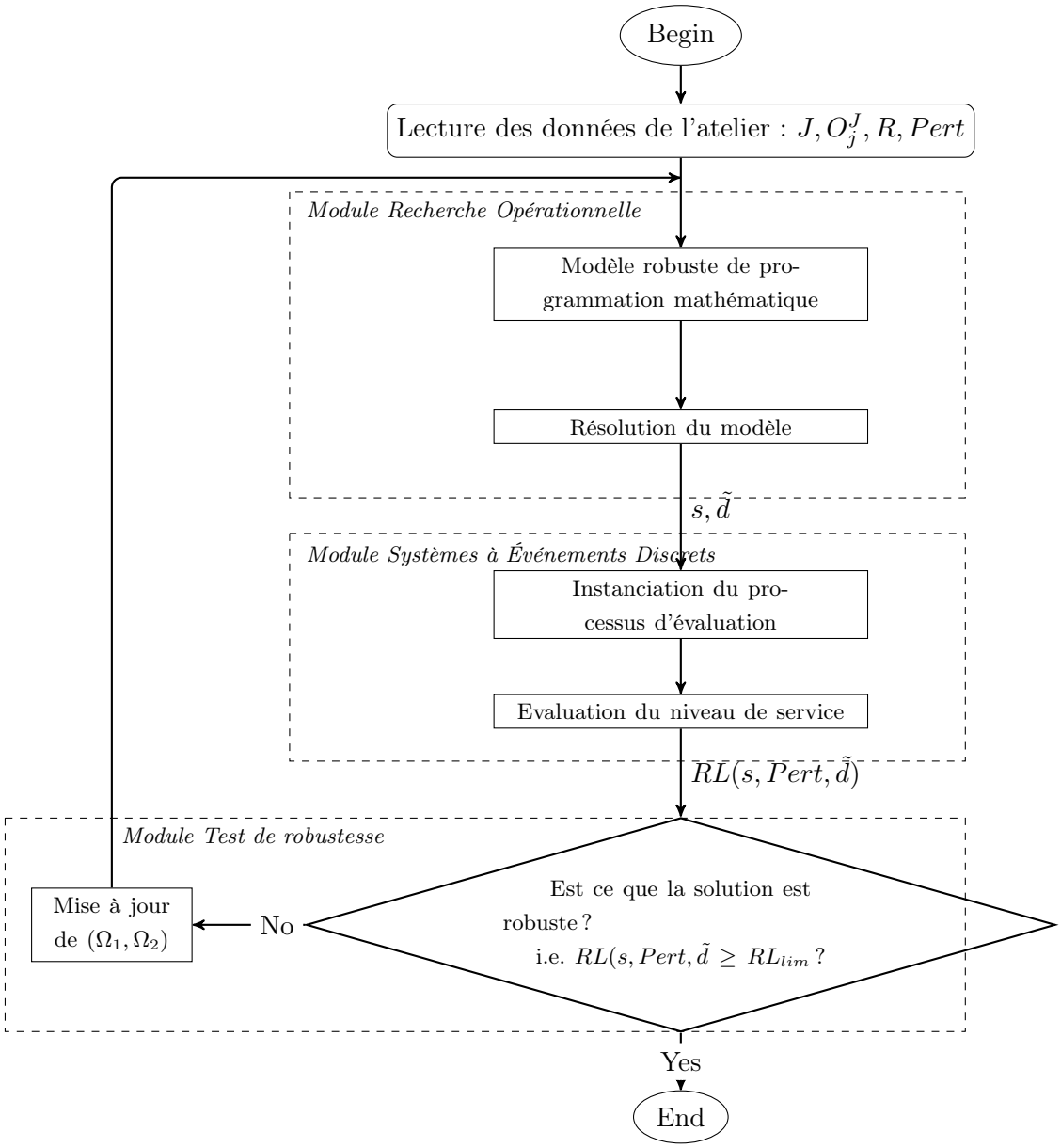


FIGURE 5.3 – Méthode hybride pour trouver la solution robuste [Marangé et al., 2020]

Pour résoudre le problème de l'optimisation de la robustesse, l'approche combinant l'optimisation et l'évaluation est utilisée. Dans le cas de cette application, l'atelier étudié est un atelier

machine	d_{1r}^{min}	d_{1r}^{max}	d_{2r}^{min}	d_{2r}^{max}	d_{3r}^{min}	d_{3r}^{max}	d_{4r}^{min}	d_{4r}^{max}	d_{5r}^{min}	d_{5r}^{max}
r_3	1	1	1	3	2	8	1	5	3	13
r_4	1	3	1	1	1	3	2	6	3	9
	d_{6r}^{min}	d_{6r}^{max}	d_{7r}^{min}	d_{7r}^{max}	d_{8r}^{min}	d_{8r}^{max}	d_{9r}^{min}	d_{9r}^{max}	d_{10r}^{min}	d_{10r}^{max}
r_3	1	3	2	6	1	3	3	5	1	1
r_4	4	6	1	5	1	7	1	3	1	1

TABLE 5.5 – Caractéristiques des jobs pour $R2||C_{max}$

à machines parallèles avec 10 jobs à produire (voir table 5.5). L'approche d'optimisation robuste proposée permet d'exécuter un algorithme d'optimisation puis le processus d'évaluation itérativement afin de trouver la solution d'ordonnancement robuste s qui permet de satisfaire le niveau de service RL_{lim} et de prendre en compte l'occurrence des perturbations $Pert$.

Les différentes itérations de l'approche hybride, explorent deux solutions alternatives. La première solution s_7 permet d'affecter les jobs j_1, j_4, j_6, j_7, j_8 à la ressource r_3 et d'affecter les jobs j_2, j_3, j_5, j_9 et j_{10} à la ressource r_4. La solution s_8 permet d'affecter les jobs j_1,j_4,j_6, j_7, j_8, j_{10} à la ressource r_3 et les jobs j_2,j_3 , j_5 et j_9 à la ressource r_4.
L'approche hybride permet de faire converger vers une solution avec un bon niveau de service sans trop dégrader le makespan. Le makespan de la solution déterministe optimale est égale $12UT$ associé à un niveau de service de 65%. Si le décideur accepte de dégrader ce makespan de 20%, ce qui veut dire que le makespan augmente à 14, le niveau de service atteint donc 90%.

Afin d'apporter une réponse efficace au problème d'optimisation nous avons proposé une méthode combinant la programmation mathématique robuste et des systèmes à événements discrets. Cela permet d'atteindre le niveau de service souhaité par le décideur.

5.4 Conclusion

Dans ce chapitre, le lien entre le processus d'évaluation présenté dans le cœur de cette thèse et le cadre industriel est établi et ce par le biais d'un scénario industriel. Ce scénario industriel définit plusieurs problèmes de décision qui sont traités (Table 5.6). En effet, ce scénario défini plusieurs situations nécessitant l'utilisation du processus d'évaluation et faisant appel à plusieurs problèmes de robustesse. La résolution de ces problèmes permet de répondre aux différents besoins du décideur dans l'atelier de production. À la fin de ce chapitre, nous proposons une situation où le décideur a besoin d'optimiser une ligne d'emballage importante pour le bon fonctionnement de la production de l'entreprise. Pour résoudre cette situation, une première approche hybride entre méthode d'optimisation et d'évaluation est proposé dans le cadre d'une collaboration entre les deux communauté RO et SED. Cette première approche permet d'explorer la possibilité d'utilisation du processus d'évaluation en combinaison avec les approches existantes dans l'objectif de proposer des solutions d'ordonnancement robustes et optimales.

Dans la première section, nous avons présenté quatre problèmes pouvant être intéressant d'un point de vue aide à la décision. La deuxième section permet d'explorer la combinaison de l'approche proposée de l'évaluation de la robustesse avec des approches d'optimisation existant dans la littérature et ce dans l'objectif de trouver une solution d'ordonnancement robuste.

D'autres cas d'utilisation peuvent être décrits en utilisant le processus d'évaluation, en effet dans le deuxième chapitre de la thèse, d'autres problèmes d'évaluation et d'optimisation sont identifiés. Il est également possible d'adapter l'utilisation du processus à d'autres besoins d'aide à la décision pouvant être extrait de données industrielles.

Situation	Question	Utilisation du processus d'évaluation	Résolution de la situation
Au t= 1 jour , l'ouverture d'une nouvelle ligne de production $F2\|\|C_{max}$	Quel est le maximum d'incertitudes qui peut être absorbé par l'ordonnancement ?	Problème d'analyse de la stabilité	$\overline{Pert} = \{u_1^{ex}(\delta = +/-37\%)\}$
Au t= 6 mois, données collectées et traitées sur la nouvelle ligne	Quel est l'ordonnancement robuste à implémenter dans cette situation ?	Analyse du niveau de service de plusieurs ordonnancement	Ordonnancement robuste = s_4 avec $RL(s_4, Pert, \tilde{d}) = 97\%$.
Les produits de la lignes sont utilisés par l'atelier d'emballage.	Quelle est la deadline minimale que je peux promettre à l'atelier aval ?	Analyse de la sensibilité de l'ordonnancement choisi	$\underline{\tilde{d}} = 53$ pour l'ordonnancement s_4
Implémentation de l'ordonnancement s_4 nécessite une surveillance du bon déroulement de la production	Quelle est la machine critique à maintenir ?	Identification des éléments critiques	ressource critique = r_2 pour s_4
Optimisation de la ligne d'emballage ($R2\|\|C_{max}$) pour garantir les performances de l'entreprise	Quel est l'ordonnancement optimal robuste que le décideur peut proposer ?	Approche hybride d'optimisation robuste (RO/SED)	Ordonnancement : s_8 avec $RL(s_8, Pert, \tilde{d}) = 90\%$ et $C_{max} = 14h$.

TABLE 5.6 – Utilisations du processus d'évaluation dans un contexte industriel

Conclusions et perspectives

Les travaux de thèse présentés dans ce manuscrit s'intéressent à la problématique de l'ordonnancement d'un atelier de production en considérant les perturbations. Dans le contexte de l'industrie 4.0 où les ateliers de production sont soumis à des contraintes de flexibilité et d'agilité, la prise en compte de l'impact des perturbations est primordiale. Les travaux rapportés dans ce manuscrit avaient justement pour objectif de traiter la question de l'ordonnancement sous perturbations. Une synthèse des contributions principales de ces travaux est proposée dans la figure 1 et détaillée chapitre par chapitre dans les paragraphes suivants.

Pour définir le périmètre scientifique, le chapitre 1 a établi un état de l'art autour de la question de l'ordonnancement sous perturbations. Cet état de l'art a montré que les approches existantes de recherche opérationnelle étaient souvent dédiées au type d'ateliers, de perturbations et dépendant des performances spécifiques à évaluer. Ces hypothèses sont difficiles à tenir dans le cadre de l'industrie 4.0, où les ateliers de production sont flexibles et agiles. Il a donc été montré qu'il était important d'avoir une approche générique et adaptable à tout type d'atelier et de perturbations. Ce chapitre a mis en avant que les approches SED stochastiques avaient un potentiel de modélisation et de résolution qui correspond à cet objectif.

Le chapitre 2 a proposé une formalisation des problèmes d'ordonnancement robuste en se basant sur le niveau de service $RL(s, Pert, \tilde{d})$ qui permet d'évaluer la probabilité que la durée d'exécution d'un ordonnancement s (son C_{max}) soumis à des perturbations $Pert$ satisfasse une deadline $\tilde{d}$. Il a été montré que le problème d'ordonnancement robuste était un problème multi-facettes où différents points de vue (stabilité, sensibilité, niveau de service) sont nécessaires pour l'étudier. Huit problèmes d'évaluation et d'optimisation de la robustesse ont été mis en avant, tous basés sur l'analyse du niveau de service. Cette formalisation offre un cadre d'étude au décideur confronté au choix d'un ordonnancement à exécuter dans son atelier face à des perturbations.

Le chapitre 3 a proposé une modélisation et une approche permettant d'évaluer le niveau de service. La modélisation basée sur les automates temporisés stochastiques proposée est modulaire et générique, car elle est décomposée en modules séparant le comportement de l'ordonnancement (patrons d'opérations, patrons de ressources) de celui des perturbations (patrons d'incertitudes, patrons d'aléas). Une traduction du niveau de service en langage formel a permis de définir une méthode d'évaluation basée sur la vérification formelle par Model checking statistique.

Suite à la proposition d'une approche d'évaluation du niveau de service, le chapitre 4 a permis d'évaluer les performances de celle-ci selon trois aspects : généricité, sensibilité et passage à l'échelle. Il a été montré dans ce chapitre que le processus d'évaluation est générique vis-à-vis du type d'atelier, et du type de perturbations pris en compte. Des plans d'expériences ont permis d'une part d'évaluer la sensibilité du processus aux paramètres d'entrée et d'identifier

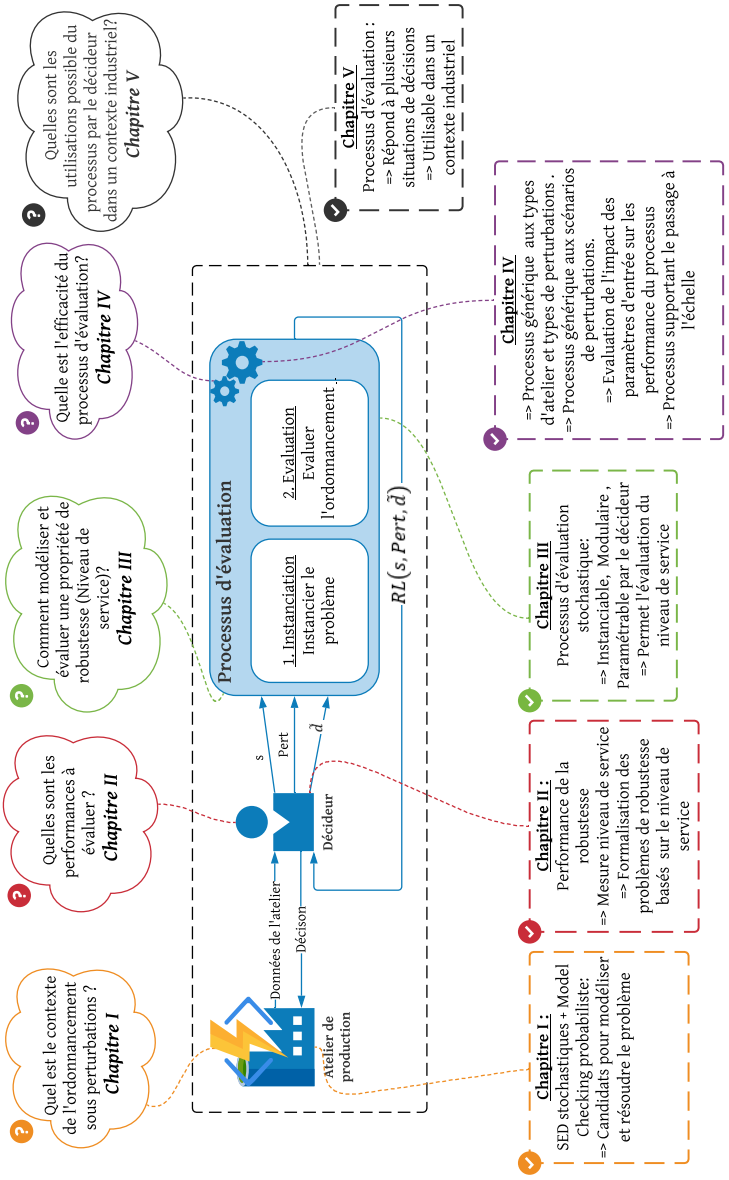


FIGURE 1 – Synthèse des contributions de la thèse

les paramètres critiques vis-à-vis des performances du processus (temps de résolution et nombre de simulations), et d'autre part d'évaluer la capacité du processus d'évaluation à supporter un passage à l'échelle. Ces analyses ont permis de conclure sur le potentiel d'application de cette approche dans un cadre industriel.

Le processus d'évaluation proposé au chapitre 3 est aussi un outil d'aide à la décision. Le chapitre 5 a montré comment la mise en œuvre de ce processus sur un scénario industriel permet de répondre à de multiples problèmes de décision en contexte perturbé. Ces problèmes de décision ont été traduits comme des instances des problèmes de robustesse définis et formalisés dans le chapitre 2 démontrant la pertinence de ce cadre d'étude. Puis l'application du processus

d'évaluation sur ces problèmes a permis de répondre aux questions initiales du décideur.

L'objectif général de cette thèse était de démontrer l'hypothèse de recherche que les SED étaient de bons candidats pour modéliser et évaluer les performances de robustesse d'un ordonnancement. L'approche de modélisation et d'évaluation, et l'analyse des performances de notre processus d'évaluation ont montré la faisabilité et l'applicabilité de cette hypothèse. De plus, tout au long du manuscrit différents ateliers, différentes perturbations et l'utilisation pour un scénario industriel ont démontré la généricité de notre approche et confirmé l'hypothèse de recherche émise dans l'introduction. De plus, ces résultats permettent de mettre en évidence des perspectives à plus ou moins long terme que nous exposons dans la suite.

Dans ce manuscrit, le processus d'évaluation proposé peut *a priori* être utilisé sur des problèmes de taille industrielle sans que nous ayons pu le démontrer sur un cas industriel réel. Il serait donc intéressant d'explorer le réel potentiel de ce processus dans le cadre d'une collaboration industrielle. Le contexte industriel permettrait de s'ouvrir sur d'autres questions de recherche qui correspondent à l'intérêt de l'utilisation du processus d'évaluation dans un atelier de production. La piste de plusieurs autres cas d'utilisation du processus dans un contexte industriel réel est également intéressante. Dans ce cadre, une collaboration industrielle a été lancée dans la suite des travaux de thèse afin d'aborder cette perspective rapidement.

Pour faciliter l'utilisation du processus d'évaluation, il devient nécessaire de mettre en place un outil ergonomique et intuitif afin de permettre à des décideurs non spécialistes des modèles SED d'utiliser le processus. L'objectif d'un tel outil est de développer une interface de communication avec le décideur afin que la modélisation du problème reste une boîte noire pour l'utilisateur. Ce dernier doit juste rentre les paramètres du problèmes. Dans les travaux précédents de notre équipe de recherche sur l'ordonnancement déterministe par la vérification formelle de modèles [Marangé et al., 2011, Marangé et al., 2016], un premier prototype logiciel a été développé [Pannequin et al., 2017]. L'objectif de ce premier logiciel est de générer un ordonnancement faisable en utilisant les modèle SED basée sur les automates temporisés. Des travaux d'évolution de cet outil on déjà été lancés afin d'intégrer nos derniers résultats. L'objectif est ainsi d'intégrer le processus d'évaluation et de proposer au décideur un outil qui non seulement génère un ordonnancement mais également évalue sa robustesse.

Enfin, les travaux de recherches présentés dans ce mémoire permettent de mettre en évidence des perspectives de recherche à long terme, notamment sur les hypothèses sous-jacentes à ces travaux qui pourraient être levées ou réduites. Nous avons fait l'hypothèse que les données de l'atelier de production avaient déjà été traitées en amont (notamment par des outils des "data sciences") afin d'en extraire les informations pour paramétrer les modèles d'ordonnancement et de perturbations. Une première perspective à long terme serait d'intégrer cet analyse dans le processus d'évaluation afin d'automatiser le paramétrage des modèles d'opérations, de ressources et de perturbations et leur mise à jour à partir des données remontant du terrain dans un contexte d'industrie 4.0. Une deuxième perspective qui s'inscrit dans ce cadre est d'étudier l'approximation de la discrétisation des modèles stochastiques des perturbations afin de garantir un niveau de précision élevé et de ne pas perdre de l'information entre le modèle continu et le modèle discret de la perturbation.

Dans la collaboration effectuée entre les deux groupes de travail GT SED et GT BERMUDES, une première approche hybride (optimisation RO / Evaluation SED) a été proposée. Pour permettre l'hybridation du processus d'évaluation avec la méthode d'optimisation robuste

de [Bertsimas and Sim, 2004], nous avons proposé un premier module d'interaction très simple. En effet, une perspective intéressante dans ce cas est d'améliorer ce module pour qu'il soit également adaptable à toute approche d'optimisation robuste et donc maintenir les caractéristiques de généricité du processus d'évaluation.

L'extension de l'utilisation du processus d'évaluation dans d'autres domaine que la production est également envisageable. En effet, l'ordonnancement sous perturbations est une problématique commune entre plusieurs domaines notamment celui de la santé ou encore dans la gestion de projet. L'évaluation de la performance de la robustesse pourrait donc être un atout considérable et apporter de l'information dans des secteurs sensibles aux perturbations. De plus, la structure du processus lui permet de s'adapter aux changements de problématique sans pour autant avoir un coût de modification important. À titre d'exemple, dans la gestion de projet, une opération est équivalente à une sous-tâche à faire, le job est une tâche entière à faire et la ressource peut être une ressource humaine ou informatique.

Bibliographie

[Abdeddaïm et al., 2006] Abdeddaïm, Y., Asarin, E., and Maler, O. (2006). Scheduling with timed automata. *Theoretical Computer Science*, 354(2) :272 – 300.

[Abdeddaïm and Maler, 2001] Abdeddaïm, Y. and Maler, O. (2001). Job-shop scheduling using timed automata ? In Berry, G., Comon, H., and Finkel, A., editors, *Computer Aided Verification*, pages 478–492, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Abdeddaïm and Masson, 2012] Abdeddaïm, Y. and Masson, D. (2012). Real-Time Scheduling of Energy Harvesting Embedded Systems with Timed Automata. *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2012 IEEE 18th International Conference on*, pages 31–40.

[Al-hinai and Elmekkawy, 2011] Al-hinai, N. and 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*, 132(2) :279–291.

[Allen et al., 2014] Allen, A., Caldwell, J., Heard, C., Sakiotis, I., and Tillinghast, D. (2014). Discrete event simulation for supporting production planning and scheduling decision in job shop facilities. In *MODSIM World 2014*, pages 444–448.

[Alur and Dill, 1994] Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235.

[Aubry, 2018] Aubry, A. (2018). *Systèmes industriels soumis à de la variabilité : vers un pilotage sous perturbations pour les systèmes de production holoniques.* Habilitation à diriger des recherches, Université de Lorraine.

[Aubry et al., 2009] Aubry, A., Rossi, A., and Jacomino, M. (2009). A generic off-line approach for dealing with uncertainty in production systems optimisation. *IFAC Proceedings Volumes*, 42(4) :1481 – 1486. 13th IFAC Symposium on Information Control Problems in Manufacturing.

[Aziz et al., 1995] Aziz, A., Singhal, V., Balarin, F., Brayton, R. K., and Sangiovanni-Vincentelli, A. L. (1995). It usually works : The temporal logic of stochastic systems. In Wolper, P., editor, *Computer Aided Verification*, pages 155–165, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Ballarini et al., 2011] Ballarini, P., Djafri, H., Duflot, M., Haddad, S., and Pekergin, N. (2011). Petri nets compositional modeling and verification of flexible manufacturing systems. In *2011 IEEE International Conference on Automation Science and Engineering*, pages 588–593.

[Barbot, 2014] Barbot, B. (2014). *Acceleration for statistical model checking.* Theses, École normale supérieure de Cachan - ENS Cachan.

[Bardin, 2008] Bardin, S. (2008). Introduction au model checking. *CEA, LIST, Safety Software laboratory.*

[Bause and Kritzinger, 2002] Bause, F. and Kritzinger, P. S. (2002). *Stochastic petri nets : an introduction to the theory*, volume 1. Friedrich Vieweg & Sohn Verlagsgesellschaft.

[Beck and Wilson, 2004] Beck, J. C. and Wilson, N. (2004). Job shop scheduling with probabilistic durations. In *Proceedings of the 16th European Conference on Artificial Intelligence*, page 652–656.

[Bertsimas and Sim, 2004] Bertsimas, D. and Sim, M. (2004). The Price of Robustness. *Operations Research*, 52(1) :35–53.

[Billaut et al., 2010] Billaut, J.-C., Moukrim, A., and Sanlaville, E. (2010). *Flexibility and Robustness in Scheduling*. ISTE-Wiley publishing,London.

[Birge and Louveaux, 1997] Birge, J. R. and Louveaux, F. (1997). *Introduction to stochastic programming*. Springer, New York.

[Briand et al., 2007] Briand, C., La, H. T., and Erschler, J. (2007). A robust approach for the single machine scheduling problem. *Journal of Scheduling*, 10(3) :209–221.

[Cai et al., 2009] Cai, X., Wu, X., and Zhou, X. (2009). Stochastic Scheduling Subject to Preemptive-Repeat Breakdowns with Incomplete Information. *Operations Research*, 57(5) :1236–1249.

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

[Castañeda et al., 2011] Castañeda, G. A. P., Aubry, J.-F., and Brinzei, N. (2011). Stochastic hybrid automata model for dynamic reliability assessment. *Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability*, 225(1) :28–41.

[Cherif et al., 2019] Cherif, G., Leclercq, E., and Lefebvre, D. (2019). Hybrid fms scheduling using t-tpn and beam search in uncertain environments. 1 :405–410.

[Chiola et al., 1993] Chiola, G., Marsan, M. A., Balbo, G., and Conte, G. (1993). Generalized stochastic petri nets : a definition at the net level and its implications. *IEEE Transactions on Software Engineering*, 19(2) :89–107.

[Daniels and Carrillo, 1997] Daniels, R. L. and Carrillo, J. E. (1997). β-Robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions*, 29(11) :977–985.

[Daoui and Lefebvre, 2017] Daoui, C. and Lefebvre, D. (2017). Control design for untimed petri nets using markov decision processes. *Operations Research and Decisions*, 27(4) :27–43.

[Dauzère-Pérès et al., 2010] Dauzère-Pérès, S., Castagliola, P., and Lahlou, C. (2010). *Service Level in Scheduling*, chapter 5, pages 99–121. John Wiley & Sons, Ltd.

[Davenport et al., 2014] Davenport, A., Gefflot, C., and Beck, C. (2014). Slack-based techniques for robust schedules. In *Sixth European conference on planning*.

[David et al., 2015] David, A., Larsen, K. G., Legay, A., Mikučionis, M., and Poulsen, D. B. (2015). Uppaal smc tutorial. *International Journal on Software Tools for Technology Transfer*, 17(4) :397–415.

[Dias and Ierapetritou, 2016] Dias, L. S. and Ierapetritou, M. G. (2016). Integration of scheduling and control under uncertainties : Review and challenges. *Chemical Engineering Research and Design*, 116 :98 – 113. Process Systems Engineering - A Celebration in Professor Roger Sargent's 90th Year.

[Dubois et al., 2003] Dubois, D., Fargier, H., and Fortemps, P. (2003). Fuzzy scheduling : Modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research*, 147(2) :231–252.

[Elyasi and Salmasi, 2012] Elyasi, A. and Salmasi, N. (2012). Stochastic scheduling with minimizing the number of tardy jobs using chance constrained programming. *Mathematical and Computer Modelling*, 57(5-6) :1154–1164.

[Feng et al., 2012] Feng, W., Zheng, L., and Li, J. (2012). The robustness of scheduling policies in multi-product manufacturing systems with sequence-dependent setup times and finite buffers. *Computers & Industrial Engineering*, 63(4) :1145–1153.

[Feng et al., 2016] Feng, X., Zheng, F., and Xu, Y. (2016). Robust scheduling of a two-stage hybrid flow shop with uncertain interval processing times. *International Journal of Production Research*, 54(12) :3706–3717.

[Galbraith, 1973] Galbraith, J. R. (1973). *Designing complex organizations*. Addison-Wesley Longman Publishing Co., Inc.

[Ghezail et al., 2010] Ghezail, F., Pierreval, H., and Hajri-Gabouj, S. (2010). Analysis of robustness in proactive scheduling : A graphical approach. *Computers & Industrial Engineering*, 58(2) :193 – 198.

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

[Goren and Sabuncuoglu, 2008] Goren, S. and Sabuncuoglu, I. (2008). Robustness and stability measures for scheduling : single-machine environment. *IIE Transactions*, 40(1) :66–83.

[Goupy, 2006] Goupy, J. (2006). *Plans d'expériences*. Ed. Techniques Ingénieur.

[Graham et al., 1979] Graham, R., Lawler, E., Lenstra, J., and Kan, A. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling : a Survey. *Annals of Discrete Mathematics*, 5 :287–326.

[Gyulai et al., 2017] Gyulai, D., Pfeiffer, A., and Monostori, L. (2017). Robust production planning and control for multi-stage systems with flexible final assembly lines. *International Journal of Production Research*, 55(13) :3657–3673.

[Harrison et al., 1996] Harrison, E. F., Frishammar, J., Tarter, C. J., and Hoy, W. K. (1996). A process perspective on strategic decision making. *Journal of Educational Administration*, 34(1) :212–228.

[Himmiche et al., 2018a] Himmiche, S., Aubry, A., Marangé, P., Duflot-Kremer, M., and Pétin, J.-F. (2018a). Using statistical-model-checking-based simulation for evaluating the robustness of a production schedule. In *Service Orientation in Holonic and Multi-Agent Manufacturing*, pages 345–357. Springer.

[Himmiche et al., 2017] Himmiche, S., Aubry, A., Marangé, P., and Pétin, J.-F. (2017). Modeling flexible workshops scheduling problems : evaluating a timed automata based approach vs milp. *IFAC-PapersOnLine*, 50(1) :1225–1230. 20th IFAC World Congress.

[Himmiche et al., 2018b] Himmiche, S., Marangé, P., Aubry, A., and Pétin, J.-F. (2018b). Robust production scheduling under machine failures-a des based evaluation approach. *IFAC-PapersOnLine*, 51(7) :271–276.

[Himmiche et al., 2019] Himmiche, S., Marangé, P., Aubry, A., and Pétin, J.-F. (2019). Approche et cadre de modélisation pour l'évaluation de l'impact de perturbations sur un ordonnancement. In *12ème Colloque sur la Modélisation des Systèmes Réactifs, MSR'19*.

[Ho, 1989] Ho, C.-J. (1989). Evaluating the impact of operating environments on mrp system nervousness. *International Journal of Production Research*, 27(7) :1115–1135.

[Ivanov et al., 2016] Ivanov, D., Dolgui, A., Sokolov, B., and Werner, F. (2016). Schedule robustness analysis with the help of attainable sets in continuous flow problem under capacity disruptions. *International Journal of Production Research*, 54(11) :3397–3413.

[Jensen, 2001] Jensen, M. T. (2001). Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures. *Applied Soft Computing*, 1(1) :35–52.

[Kádár et al., 2004] Kádár, B., Pfeiffer, A., and Monostori, L. (2004). Discrete event simulation for supporting production planning and scheduling decisions in digital factories. In *Proceedings of the 37th CIRP international seminar on manufacturing systems*, pages 444–448.

[Kall et al., 1994] Kall, P., Wallace, S. W., and Kall, P. (1994). *Stochastic programming*. Springer.

[Kemeny and Snell, 1976] Kemeny, J. G. and Snell, J. L. (1976). *Markov chains*. Springer-Verlag, New York.

[Kira et al., 1997] Kira, D., Kusy, M., and Rakita, I. (1997). A stochastic linear programming approach to hierarchical production planning. *Journal of the Operational Research Society*, 48(2) :207–211.

[Kouvelis and Yu, 1997] Kouvelis, P. and Yu, G. (1997). *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, Dordrecht.

[Kwiatkowska et al., 2002] Kwiatkowska, M., Norman, G., and Parker, D. (2002). Prism : Probabilistic symbolic model checker. In *Computer Performance Evaluation : Modelling Techniques and Tools*, pages 200–204, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Kwiatkowska et al., 2007] Kwiatkowska, M., Norman, G., and Parker, D. (2007). *Stochastic Model Checking*, pages 220–270. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Kwiatkowska et al., 2006] Kwiatkowska, M., Norman, G., Parker, D., and Sproston, J. (2006). Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, 29(1) :33–78.

[Larsen et al., 1997] Larsen, K. G., Pettersson, P., and Yi, W. (1997). Uppaal in a nutshell. *International journal on software tools for technology transfer*, 1(1-2) :134–152.

[Lawler and Wood, 1966] Lawler, E. L. and Wood, D. E. (1966). Branch-and-bound methods : A survey. *Operations Research*, 14(4) :699–719.

[Lefebvre and Daoui, 2020] Lefebvre, D. and Daoui, C. (2020). Control design for bounded partially controlled tpns using timed extended reachability graphs and mdp. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 50(6) :2273–2283.

[Lefebvre and Mejia, 2018] Lefebvre, D. and Mejia, G. (2018). Robust scheduling in uncertain environment with petri nets and beam search. *IFAC-PapersOnLine*, 51(11) :1077 – 1082. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[Li and Ierapetritou, 2008] Li, Z. and Ierapetritou, M. (2008). Process scheduling under uncertainty : Review and challenges. *Computers & Chemical Engineering*, 32(4) :715 – 727.

[Liu and Sahinidis, 1996] Liu, M. L. and Sahinidis, N. V. (1996). Optimization in process planning under uncertainty. *Industrial & Engineering Chemistry Research*, 35(11) :4154–4165.

[Marangé et al., 2016] Marangé, P., Aubry, A., and Pétin, J.-F. (2016). Ordonnancement d'ateliers à partir de patrons de modélisation basés sur des automates communicants. In *11th International Conference on Modeling, Optimization and Simulation*, Montréal, Canada.

[Marangé et al., 2020] Marangé, P., Lemoine, D., Aubry, A., Himmiche, S., Norre, S., Bloch, C., and Pétin, J.-F. (2020). *Coupling Robust Optimization and Model-Checking Techniques for Robust Scheduling in the Context of Industry 4.0*, pages 103–124. Springer International Publishing, Cham.

[Marangé et al., 2011] Marangé, P., Pétin, J.-F., Manceaux, A., and Gouyon, D. (2011). Contribution à la reconfiguration des systèmes de production : ordonnancement par recherche d'atteignabilité. *Journal Européen des Systèmes Automatisés (JESA)*, 45(1/3) :45–60.

[Mezgebe, 2020] Mezgebe, T. T. (2020). *Human-inspired algorithms for designing new control system in the context of factory of the future*. PhD thesis, Université de Lorraine.

[Michel, 1984] Michel, M. (1984). Algebre de machines et logique temporelle. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 287–298. Springer.

[Monostori et al., 2016] Monostori, L., Kádár, T., Bauerhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., and Ueda, K. (2016). Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2) :621–641.

[Mula et al., 2006] Mula, J., Poler, R., García-Sabater, J., and Lario, F. (2006). Models for production planning under uncertainty : A review. *International Journal of Production Economics*, 103(1) :271 – 285.

[Nimal, 2010] Nimal, V. (2010). *Statistical approaches for probabilistic model checking*. PhD thesis, University of Oxford.

[Palacios et al., 2016] Palacios, J. J., Puente, J., Vela, C. R., and González-Rodríguez, I. (2016). Benchmarks for fuzzy job shop problems. *Information Sciences*, 329 :736 – 752.

[Panek et al., 2006] Panek, S., Engell, S., and Stursberg, O. (2006). Scheduling and planning with timed automata. In Marquardt, W. and Pantelides, C., editors, *16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering*, volume 21 of *Computer Aided Chemical Engineering*, pages 1973 – 1978. Elsevier.

[Pannequin et al., 2017] Pannequin, R., Marangé, P., Aubry, A., and Pétin, J.-F. (2017). Tabs : un outil logiciel d'ordonnancement de la production basé sur les automates temporisés. In *11ème Colloque sur la Modélisation des Systèmes Réactifs, MSR 2017*.

[Peng et al., 2016] Peng, Z., Lu, Y., and Miller, A. (2016). Uncertainty analysis of phased mission systems with probabilistic timed automata. In *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–8.

[Pinedo, 2012] Pinedo, M. (2012). *Scheduling*, volume 5. Springer.

[Plateau and Atif, 1991] Plateau, B. and Atif, K. (1991). Stochastic automata network of modeling parallel systems. *IEEE Transactions on Software Engineering*, 17(10).

[Rahimi et al., 2018] Rahimi, M., Dumitrescu, E., and Niel, E. (2018). Multi-resource sharing scheduling considering uncontrollable environment. pages 500–507.

[Rossit et al., 2019] Rossit, D. A., Tohmé, F., and Frutos, M. (2019). Production planning and scheduling in Cyber-Physical Production Systems : a review. *International Journal of Computer Integrated Manufacturing*, 32(4-5) :385–395.

[Sahinidis, 2004] Sahinidis, N. V. (2004). Optimization under uncertainty : state-of-the-art and opportunities. *Computers and Chemical Engineering*, 28 :971–983.

[Schmidt, 1992] Schmidt, G. (1992). A Decision Support System for Production Scheduling. *Journal of Decision Systems*, 1(2-3) :243–260.

[Sevaux and Sörensen, 2004] Sevaux, M. and Sörensen, K. (2004). A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2(2) :129–147.

[Simon et al., 2018] Simon, E., Oyekan, J., Hutabarat, W., Tiwari, A., and Turner, C. (2018). Adapting petri nets to discrete event simulation for the stochastic modelling of manufacturing systems. *International Journal of Simulation Modelling*, 17(1) :5–17.

[Stewart et al., 1995] Stewart, W. J., Atif, K., and Plateau, B. (1995). The numerical solution of stochastic automata networks. *European Journal of Operational Research*, 86(3) :503 – 525.

[Subbiah and Engell, 2010] Subbiah, S. and Engell, S. (2010). Short-term scheduling of multi-product batch plants with sequence-dependent changeovers using timed automata models. In Pierucci, S. and Ferraris, G. B., editors, *20th European Symposium on Computer Aided Process Engineering*, volume 28 of *Computer Aided Chemical Engineering*, pages 1201–1206. Elsevier.

[Tan, 2002] Tan, B. (2002). Production control of a pull system with production and demand uncertainty. *IEEE Transactions on Automatic Control*, 47(5) :779–783.

[Trentesaux et al., 2013] Trentesaux, D., Pach, C., Bekrar, A., Sallez, Y., Berger, T., Bonte, T., Leitão, P., and Barbosa, J. (2013). Benchmarking flexible job-shop scheduling and control systems. *Control Engineering Practice*, 21(9) :1204 – 1225.

[Vaidya et al., 2018] Vaidya, S., Ambad, P., and Bhosle, S. (2018). Industry 4.0 - A Glimpse. *Procedia Manufacturing*, 20 :233–238.

[Vieira and Frazzon, 2020] Vieira, G. E. and Frazzon, E. M. (2020). Searching for Production Robustness Through Simulation-Based Scheduling Optimization. pages 351–362. Springer, Cham.

[V.J.Leon et al., 1994] V.J.Leon, V., Wu, D., and Storer, R. (1994). Robustness measure and robust scheduling for job shops. *IIE Transactions*, 26(5) :32–43.

[Xiong et al., 2013] Xiong, J., Xing, L.-n., and Chen, Y.-w. (2013). Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. *International Journal of Production Economics*, 141(1) :112–126.

[Yao and Cassandras, 2012] Yao, C. and Cassandras, C. G. (2012). Using infinitesimal perturbation analysis of stochastic flow models to recover performance sensitivity estimates of discrete event systems. *Discrete Event Dynamic Systems*, 22(2) :197–219.

[Zandieh and Motallebi, 2018] Zandieh, M. and Motallebi, S. (2018). Determination of production planning policies for different products in process industries : using discrete event simulation. *Production Engineering*, 12 :737–746.

Annexe A

Formulation d'un problème d'ordonnancement

La combinaison du type d'atelier de production, des contraintes à considérer et des objectifs d'optimisation possibles permet de définir un nombre infini de problèmes [Pinedo, 2012]. Pour faire face à l'évolution constante que connais le monde manufacturier, [Graham et al., 1979] ont proposé une classification des problèmes de robustesse. Cette notation est plus connue par les trois champs $\alpha|\beta|\gamma$. Les auteurs présentent le problème d'ordonnancement en utilisant les trois champs, α pour décrire l'environnement de l'atelier, β pour définir les contraintes liées aux caractéristiques des jobs et γ pour décrire le critère d'optimisation souhaité. A partir de ces champs, une panoplie de problèmes d'ordonnancement est déterminée. Cet effort de classification a permis l'utilisation massive de cette notation dans la littérature pour la résolution des problèmes d'ordonnancement de la production.

— **Champ** α :

Le champ α est lié à la configuration des ressources dans l'atelier, en d'autres termes il permet de définir le type d'atelier à prendre en considération. Ce champs se décompose en deux sous-champs α_1 et α_2 ($\alpha = \alpha_1\alpha_2$). Le premier sous champ permet de définir le type de la configuration des ressources alors que le deuxième sous champs permet de déterminer le nombre de ressources à l'entrée du problème. Le sous-champs α_1 peut prendre deux types de valeurs $\alpha_1 = \{P, Q, R\}$ ou $\alpha_1 = \{F, J, O\}$, ces deux valeurs permettent de prendre en compte les ateliers de type machine parallèle ou de configuration spécifique (**F**= flow shop, **J**= job shop, **O**= open shop). . Le sous-champs α_2 peut prendre un nombre arbitraire fixé de ressources $\alpha_2 = m$ ou un nombre non fixé $\alpha_2 = \varnothing$ (Table A.1)

.

— **Champ** β :

Le champ β définit les éventuelles caractéristiques d'un job. En effet, la notation de Graham permet de recenser un grand nombre de contraintes pouvant s'appliquer aux jobs (Table A.2).

— **Champ** γ :

Le champ γ permet d'exprimer le critère d'optimisation qui permet de résoudre le problème d'ordonnancement. Trois critères usuellement traités dans la littérature : le Makespan C_{max} qui permet d'optimiser le temps de complétion de l'ordonnancement, le retard totale T_{max} qui mesure le retard d'exécution et le retard algébrique L_{max} est la mesure du retard moyen d'exécution.

La notation proposée ici permet de formuler des problèmes d'ordonnancement en utilisant les trois champs. Par exemple la notation **JMPM|pmtn|** C_{max} permet d'exprimer un problème

Le champ $\alpha = \alpha_1 \varnothing$	
Valeur α_1	Description
$\varnothing$	Machine unique
P, Q, R	Machines parallèles : Identiques (**P**), Proportionnelles (**Q**) ou Non Reliées (**R**).
F, J, O, X	**F**= Atelier Flow Shop, **J**= Atelier Job Shop, **O**= Atelier Open Shop, **X**= Atelier Mixte
HF	Atelier Flow Shop Hybride
GO	Atelier Open Shop Général
GJ	Atelier Job Shop Général

TABLE A.1 – Notations des ateliers production

Le champ β	
Valeur β	Description
s_i	Les jobs ont des dates distinctes de début
r_i	Les jobs ont des dates distinctes de fin
d_i	Les jobs ont des dates d'échéances
$\tilde{d}_i$	Les jobs ont une deadline
split	Une opération peut être partagée et réalisée simultanément sur des ressources différentes
pmtn	Une opération peut être préemptée et repris après sur n'importe quelle ressource
over	Autorisation du chevauchement de deux opérations du même job
block	L'atelier a des stocks intermédiaires à capacité limité entre les machines
batch	Les opérations peuvent être regroupées dans des lots de production.
$p - batch$	Les opérations peuvent être regroupées dans des lots et sont exécutées en parallèle dans chaque lot
$s - batch$	Les opérations peuvent être regroupées dans des lots et sont exécutées en série dans chaque lot.
R_{sd}	Une machine peut avoir un temps de démontage qui dépend de la séquence d'opérations traitée sur cette machine
S_{sd}	Une machine peut avoir un temps de préparation avant le traitement d'une opération qui dépend de la séquence d'opérations traitée sur cette machine.
ORS	Opérations qui ont une contrainte de partage de ressources. Une opération doit être exécutée sur plusieurs ressources
$unavail_j$	Une machine peut avoir une ou plusieurs périodes d'indisponibilité connues à l'avance

TABLE A.2 – Contraintes des ateliers de productions

d'optimisation du Makespan dans le cas d'un atelier Job Shop flexible avec l'autorisation de la préemption des opérations. Un atelier flexible est un atelier ayant des ressources qualifiées pour l'exécution de plusieurs types d'opération.

Annexe B

UppAal SMC

Uppaal est une boîte à outils pour la vérification des systèmes en temps réel représentés par un réseau d'automates temporisés enrichis par des variables entières, des données structurées et la synchronisation. La première version de cet outil a été lancé en 1995, il est depuis en développement constant par l'Université d'Uppsala et l'Université d'Aalborg. UppAal est un outil utilisé avec succès dans des études de cas allant des protocoles de communication aux applications multimédias.
UppAal SMC vient répondre aux besoins de représenter les systèmes complexes ayant un comportement dynamique et prenant en compte les caractéristiques stochastiques des systèmes. Cette version de l'outil prend en compte la modélisation à l'aide des automates temporisés stochastiques. Pour vérifier les modèles ATS, UppAal SMC intègre le Model-Checking statistique pour permettre l'évaluation de propriétés exprimées dans la logique $PCTL$.

L'interface graphique de l'outil UppAal SMC permet d'organiser le projet comme suit (Figure B.1). Le projet UppAal créé se présente sous forme de plusieurs fenêtres. Chaque fenêtre permet de représenter un élément du projet. Pour l'instanciation des modèles ATS, l'onglet *Déclaration* est nécessaire. Il permet de déclarer les variables, les horloges et les événements synchronisants utilisés dans les ATS. Dans cette partie, l'instanciation des patrons est également détaillée, permettant ainsi de déclarer toutes les valeurs que les paramètres du patron peuvent prendre et déterminer le nombre d'instanciation effectué. Toutes les déclarations effectuées dans cette fenêtre sont globales et donc concernent tous les patrons qui vont être modélisés dans ce projet UppAal SMC. Par la suite, les patrons sont ensuite modélisés dans les fenêtres suivantes, chaque patron est représenté par un modèle et une fenêtre de déclaration locale. En effet, l'interface graphique d'UppAal SMC permet de déclarer les variables et paramètres localement pour un modèle donné. La dernière fenêtre du projet est *Composition du système*, elle permet de faire appel aux patrons que l'utilisateur veut exécuter, pour chaque patron, un nombre d'instanciation est défini permettant ainsi la flexibilité dans le nombre de modèles souhaité.
Pour vérifier les propriétés souhaitées, les propriétés sont déclarées dans la fenêtre *Vérifieur*.

Pour vérifier une propriété à l'aide du Model-Checker UppAal SMC, la propriété PCTL est insérée dans le "Vérifieur". Les paramètres de vérification sont alors spécifiées.

A l'issue du réglage des paramètres statistiques, UppAal permet de lancer la vérification et à la fin de cette dernière, une fenêtre de résultats s'ouvre (Figure B.3) permettant d'avoir le nombre de simulations, le calcul de probabilité et intervalle de confiance.

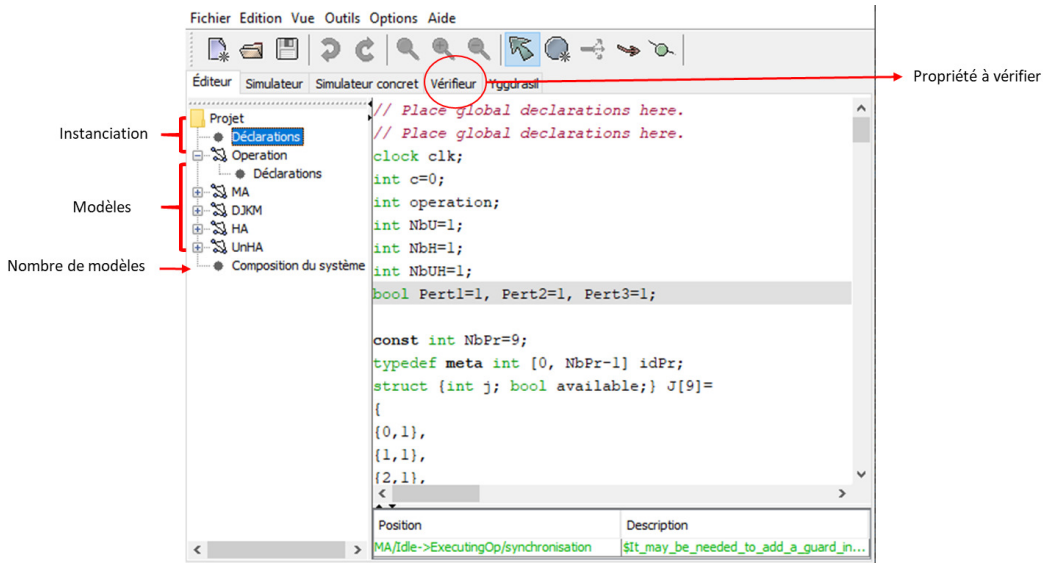


FIGURE B.1 – Outil UppAal SMC : présentation

```
const int NbPr=9;
typedef meta int [0, NbPr-1] idPr;
struct {int j; bool available;} J[9]=
{
{0,1},
{1,1},
{2,1},
{3,1},
{4,1},
{5,1},
{6,1},
{7,1},
{8,1}

};
```

FIGURE B.2 – Exemple de déclaration dans UppAal SMC

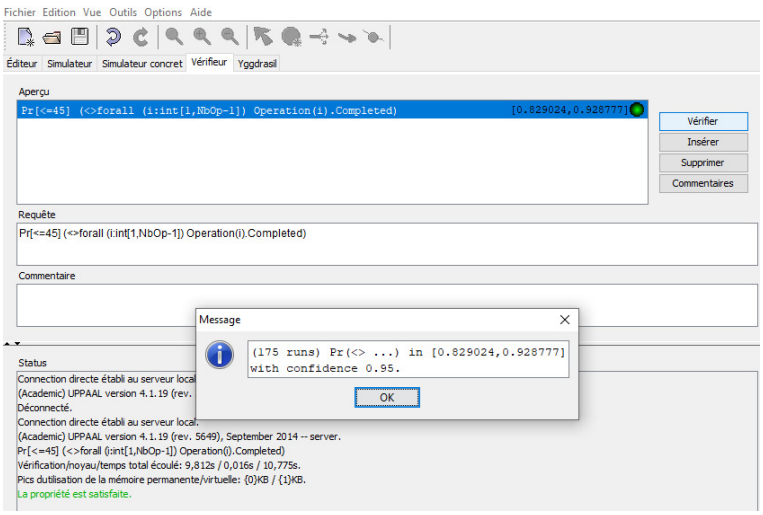


FIGURE B.3 – Exemple de vérification d'une propriété sur UppAal

Annexe C

Niveau de service VS L_λ-robustesse

L'ordonnancement de production est un problème qui surgit très souvent dans la littérature scientifique. Le sujet que nous traitons est une mutation de ce dernier à un problème plus réaliste qui prend en compte les perturbations auxquelles un atelier de production peut faire face. La formulation du problème dépend du type d'évaluation qu'on veut faire. La robustesse est une notion qui caractérise l'évaluation de l'ordonnancement de production sous perturbations. L'objectif ici est de positionner notre cadre d'étude de la robustesse (chapitre 2) avec celui proposé dans [Aubry, 2018]. Ce dernier repose sur la L_λ-robustesse.

C.1 La L_λ-robustesse

Pour une description par scénario des perturbations, [Aubry, 2018] définit qu'une solution d'ordonnancement s est L_λ-robuste sur un ensemble d'instances P (représentant l'espace des perturbations) relativement au critère de robustesse λ si elle vérifie la condition : $\lambda(s, z, P) \leq L_\lambda$. Une solution est alors dite robuste si elle permet de garantir un niveau de performance L_λ selon un critère de robustesse λ quelle que soit l'instance de P.

En pratique $\lambda(s, z, P)$ mesure la performance de s relativement au critère z sur l'ensemble P. Dans [Kouvelis and Yu, 1997], plusieurs mesures $\lambda(s, z, P)$ sont proposées. L'une d'elle consiste à fixer $\lambda(s, z, P) = \max_{I \in P} (z(s, I))$ avec $z(s, I)$ qui mesure le critère z pour s appliquée au scénario I de P. Les approches par scénario considère souvent que chaque scénario est équiprobable (loi uniforme sur P).

C.2 Comparaison avec le niveau de service

Si $z = C_{max}$ et $\tilde{d} = L_\lambda$, la L_λ-robustesse revient donc à fixer $RL_{lim} = 100\%$ et à considérer une loi uniforme pour chaque incertitude. Avec ces paramètres particuliers, une solution robuste dans notre cadre est également L_λ-robuste. On peut également noté que nous avons considéré dans le chapitre 2 uniquement le cas du C_{max} comme critère de performance pour l'ordonnancement. Nous aurions pu généraliser notre cadre d'étude en considérant n'importe quel z comme fonction objectif et généraliser $\tilde{d}$ par un z_{limite}. Ainsi la L_λ-robustesse peut être vue comme un cas particulier du niveau de service. Ceci explique que le cadre d'étude de la robustesse proposé dans le chapitre 2 de cette thèse est plus riche que celui proposé dans [Aubry, 2018] où cinq problèmes de robustesse avaient pu être identifiés.

Annexe D

Front de Pareto

D.1 Définition d'un front de Pareto

Le front de Pareto est l'ensemble des solutions de compromis. Sur la figure D.1, les points A et B sont deux points du front de Pareto : A ne domine pas B, B ne domine pas A, mais tous les deux dominent le point C. Le but de l'optimisation multi-objectif est de déterminer le front de Pareto pour un problème donné et ainsi traiter le compromis entre plusieurs objectifs. Une première approche naïve habituelle consiste à traiter successivement les objectifs. Le résultat favorise alors des solution extrêmes. Une approche un peu plus évoluée consiste à pondérer les différents objectifs. Cependant cette approche ne laisse pas le choix au décideur. De plus certaines solutions non-dominées ne peuvent pas être obtenues quels que soient les coefficients (lorsque le front de Pareto n'est pas convexe). Enfin, il est généralement difficile de faire la somme pondérée de certaines grandeurs. Il existe un certain nombre de méthodes de résolution spécifiques à un problème permettant de déterminer le front de Pareto. Bien que performantes et permettant parfois d'obtenir l'ensemble du front, ces approches présentent bien évidemment le défaut de ne pas pouvoir être employées pour un problème quelconque.

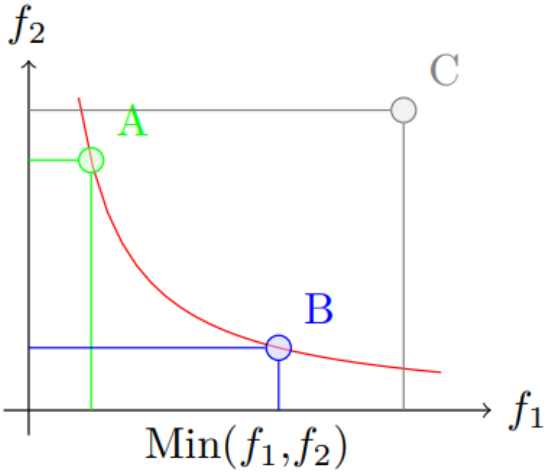


FIGURE D.1 – Exemple d'un front de Pareto

Au delà de deux objectifs, l'algorithme adopté permet de construire une surface de Pareto

(Figure D.2) se basant sur chaque objectif. Un algorithme simple et exhaustif est alors d'exécuter successivement les objectifs les uns après les autres. Même si ce type d'algorithme risque d'être vite limité en termes de taille de problème traité, il reste néanmoins efficace dans la résolution de problèmes muti-objectif (plus de 2). Pour simplifier l'exécution de ce genre d'algorithmes des solveurs existent(par exemple Choco solver).

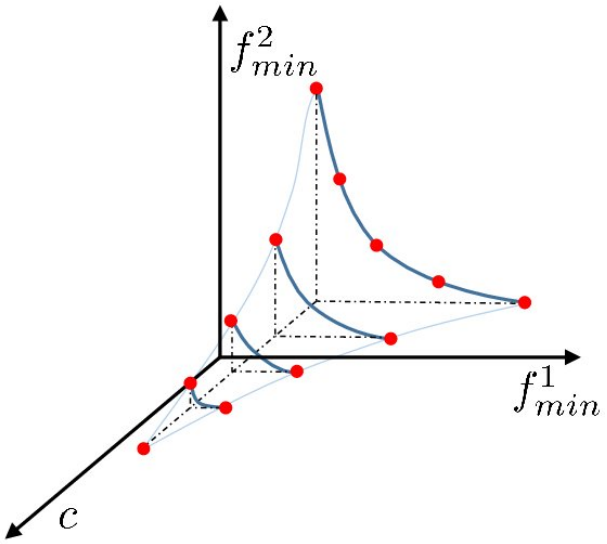


FIGURE D.2 – Exemple d'une surface de Pareto

Résumé

L'objectif principal de cette thèse est de proposer une approche permettant de considérer les perturbations dans le cadre d'un problème d'ordonnancement de la production. L'approche proposée permet de mettre l'accent sur l'importance de l'évaluation de la performance d'un ordonnancement dans un cadre perturbé.

L'ordonnancement de la production est un problème important et commun au monde de l'industrie et de la recherche scientifique. Dans le but de proposer des solutions d'ordonnancement, les méthodes proposées proposent des ordonnancements optimaux permettant de minimiser ou maximiser un critère défini. Cet objectif est certes adopté depuis plusieurs années mais ne répond plus tout à fait à la dynamique des ateliers de production. La prise en compte des perturbation est donc un enjeu majeur dans la problématique de l'ordonnancement.

Pour répondre à cet enjeu, nous proposons, de définir la performance et l'approche à adopter pour traiter l'impact des perturbations sur l'ordonnancement. La robustesse est alors identifiée comme étant une performance adaptée pour évaluer cet impact. Plus encore, spécifier la robustesse permet d'identifier un cadre formel pour spécifier les différents problèmes de robustesse auxquels l'ordonnancement doit faire face. Pour traiter le problème de la robustesse d'un ordonnancement, nous proposons un processus d'évaluation basé sur les modèles de systèmes à événements discrets. Plus précisément, les automates temporisés stochastiques et le model-checking statistique sont utilisés afin de mesurer la robustesse d'un ordonnancement perturbé. L'évolution des systèmes de production et leurs contraintes de flexibilité et d'agilité exige que l'approche d'évaluation soit également flexible et adaptable. Le processus proposé dans le cadre de cette thèse permet non seulement de s'adapter à plusieurs problèmes d'atelier de production mais également à prendre en compte des situations de décisions réelles dans un contexte industriel.

Mots-clés: Ordonnancement de la production, Perturbations, Robustesse, Systèmes à événements discrets, Model Checking statistique

Abstract

The main objective of this thesis is to propose an approach to consider perturbations in the context of a production scheduling problem. The proposed approach emphasizes the importance of evaluating the performance of a scheduling in a disturbed environment.

Production scheduling is an important and common problem in the world of industry and scientific research. In order to propose scheduling solutions, the suggested methods propose optimal scheduling to minimize or maximize a defined criterion. This objective has certainly been adopted for several years but no longer fully meets the dynamics of production workshops. Taking perturbations into account is therefore a major issue in the scheduling problem.

To address this issue, after an introduction and state of the art, we propose, in the main body of the manuscript, the definition of the performance and appropriate approach to deal with the impact of perturbations in scheduling. Robustness is then identified as a suitable performance to

assess this impact. Furthermore, specifying robustness allows to identify a formal framework to specify the different robustness problems that scheduling must face. To address the problem of scheduling robustness, we propose an evaluation process based on discrete-event system models. More precisely, stochastic timed automata and statistical model-checking are used to measure the robustness of a perturbed schedule. The evolution of production systems and their flexibility and agility constraints requires that the evaluation approach must also be flexible and adaptable. The process proposed in this thesis allows not only its adaptation to several production workshop problems but also to take into account real decision situations in an industrial context. To illustrate the feasibility and applicability of this process, the UppAal SMC tool is used.

Keywords: Scheduling Production, Perturbations, Robustness, Discrete Event Systems, Statistical Model-Checking