Conception d'un solveur matériel spécifique pour la résolution rapide du problème SAT appliqué à l'évaluation du risque en génie industriel

Khadija Bousmar

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Conception d'un solveur matériel spécifique pour la résolution rapide du problème SAT appliqué à l'évaluation du risque en génie industriel

THÈSE

présentée et soutenue publiquement le 6 juillet 2018

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention Automatique, Traitement du signal et des images, Génie informatique)

par

Khadija BOUSMAR

Composition du jury

Rapporteurs :	Emmanuel SIMEU	MdC-HDR, TIMA, Université Grenoble Alpes
	Alice YALAOUI	MdC-HDR, Université de Technologie de Troyes
Examinateurs :	Hamid ALLAOUI	Professeur, Université d'Artois
	Anis CHELBI	Professeur, ENSIT Tunis
	Zineb HABBAS	Professeur, Université de Lorraine
	Mohamed TABAA	MdC, LPRI, EMSI Casablanca
Invités :	Sid-Ali ADDOUCHE	MdC, IUT de Montreuil
	Karim ALAMI	Professeur, LPRI, EMSI Casablanca
	Abbas DANDACHE	Professeur, LGIPM, Université de Lorraine
Directeur de thèse :	Sofiene DELLAGI	MdC-HDR, LGIPM, Université de Lorraine
Co-directeur de thèse :	Fabrice MONTEIRO	Professeur, LGIPM, Université de Lorraine

Laboratoire de Génie Informatique, de Production et de Maintenance (EA 3096)

Mis en page avec la classe thesul.

Je dédie cette thèse

A mes parents et ma grand-mère :

Qui m'ont accompagné par leurs prières, et leurs douceurs, puisse Dieu leur prêter longue vie, santé et bonheur et faire en sorte que jamais je ne vous déçoive.

A mes sœurs et à mon frère :

Ma moitié, sans vous je ne saurai quoi faire, je vous souhaite une vie pleine de bonheur et de succès et que Dieu, le tout puissant, vous protège et vous garde.

A mon mari :

Une spéciale dédicace à toi qui compte énormément pour moi, et à qui je porte beaucoup de tendresse, d'amour et de respect, puisse Dieu te protège et te garde.

Remerciements

Les travaux de recherche présentés dans ce manuscrit ont été réalisés au sein du Laboratoire Pluridisciplinaire de Recherche et Innovation (LPRI) de l'EMSI Casablanca (Ecole Marocaine des Sciences de l'Ingénieur) en collaboration avec le laboratoire LGIPM (Laboratoire de Génie Industriel en Production et Maintenance) de Metz dont je remercie tous les membres.

Je tiens tout d'abord à exprimer ma gratitude à mon directeur et co-directeur de thèse, messieurs Sofien DELLAGI et Fabrice Monteiro, quoi que j'écrive, je ne pourrai pas exprimer ma joie et ma reconnaissance envers vous de m'avoir offert cette opportunité de faire équipe avec vous et de profiter de vos recommandations et conseils tout au long de ce parcours, sans oublier bien sûr votre côté humain qui m'a tant aidé et poussé à croire en moi et en ce que je fais et de ce que je veux être.

Mes remerciements s'enchainent avec Mme Zineb HABBAS, une femme forte, professionnelle, toujours présente pour me guider vers le bon chemin à emprunter.

Je tiens à exprimer ma reconnaissance également à trois personnes en or, serviables et motivantes, Messieurs Abbas DANDACHE, Philippe JEAN et Mohamed TABAA.

J'exprime ma gratitude aux membres du jury pour l'honneur qu'ils me font en évaluant mes travaux : Merci à Mme Yalice ALAOUI MdC-HDR à l'Université de Technologie de Troyes et Mr. Emmanuel SIMEU MdC-HDR, TIMA à l'Université Grenoble Alpes, d'avoir accepté d'être les rapporteurs de ce travail de recherche.

Merci à Mr.Hamid ALLAOUI Professeur à l'Université d'Artois, à Mr.Anis CHELBI Professeur de l'ENSIT de Tunis, à Mohamed TABAA MdC, à LPRI de l'EMSI Casablanca d'avoir accepté d'examiner ce travail de recherche.

Merci à Sid-Ali ADDOUCHE MdC de l'IUT de Montreuil, à Karim ALAMI Professeur et directeur pédagogique de l'Ecole Marocaine des Sciences de l'Ingénieur (EMSI).

Je tiens à remercier le président directeur général Kamal DAISSAOUI, le directeur pédagogique Karim ALAMI ainsi que le directeur du laboratoire LPRI Mohamed TABAA, tous de l'EMSI de Casablanca, pour leur disponibilité, la confiance qu'ils m'ont donné, et leur contribution au développement de la recherche au Maroc via des collaborations d'enseignement et de recherche afin de s'ouvrir sur le monde de la recherche et innovation et de pouvoir créer les métiers d'avenir.

J'adresse mes remerciements à tous mes collègues du Laboratoire Pluridisciplinaire de Recherche et Innovation (LPRI) et mes collègues de l'EMSI Casablanca d'être là pour m'encourager et croire en moi, ainsi qu'à ceux du laboratoire (LGIPM) de Metz de l'université de Lorraine avec qui j'ai pu partager beaucoup de moments dans un cadre agréable. Je remercie tous mes amis et spécialement : Laila (Voldemort), Yahya, Sanaa, Jawhar, Fatima ezzahra, Simo, Oumaima, Safa, Mouna, Mohamed, Redouane, Abderrahmane, Zakaria, ...

Mon immense amour et grand Merci va spécialement vers ma famille, en premier ordre mes chers aimables, uniques parents au monde avec un papa (Moulay chrif Said) protecteur, joyeux et généreux, une maman (Lalla Malika) tendre, douce et attentionnée ainsi que ma grand-mère, un grand merci à ces trois personnes à qui je dois tout dans la vie et qui ont fait de moi ce que je suis actuellement grâce à l'aide de dieu. Que ce modeste travail soit l'exaucement de vos vœux tant formulés, le fruit de vos innombrables sacrifices, bien que je ne vous en acquitterai jamais assez. Mes sœurs et mon frère, mes amours et les mains qui m'ont toujours protégée et qui ont toujours su me guider vers la bonne voie, mes beaux-frères et ma belle-sœur pour leurs conseils, mes neveux et nièces qui ont su me combler d'amour et de joie durant toute cette période. Je remercie également mon cher et doux époux qui m'a toujours encouragé à aller vers l'avant en croyant en moi et en mes compétences. Mes tantes, oncles et cousins veuillez trouver ici les expressions de mes sincères reconnaissances pour vos amabilités, vos générosités, et vos aides précieuses.

Avant d'achever je n'oublierai pas de remercier mes amis du club d'Aikido : Dojo EMSI pour leurs encouragements et compliments.

Je finie par exprimer ma gratitude à toutes personnes que j'ai omis de citer avec qui j'ai eu un échange durant ces trois années de thèse qui ont contribué à enrichir mon expérience personnelle et professionnelle.

Avant-propos

Cette thèse, préparée sous la direction du Dr. Sofiene DELLAGI et la co-direction du Pr. Fabrice MONTEIRO, s'inscrit dans le cadre d'une collaboration d'enseignement et de recherche entre l'Université de Lorraine et l'École Marocaine des Sciences de l'Ingénieur (EMSI) de Casablanca.

La thèse est financée principalement par l'EMSI dans le cadre de cette collaboration ayant notamment comme objectif l'accompagnement de l'EMSI pour le développement de l'activité de recherche en conception des systèmes électroniques ainsi que la création du Laboratoire Pluridisciplinaire en Recherche et Innovation (LPRI), dirigé par le Dr. Mohamed TABAA.

Cette collaboration a permis de développer un savoir-faire commun entre le Laboratoire de Génie Informatique de Production et Maintenance (LGIPM) de Metz et le Laboratoire Pluridisciplinaire en Recherche et Innovation (LPRI) de Casablanca, notamment dans le domaine de la conception architecturale de solutions rapides pour le développement d'outils d'aide à la gestion du risque industriel.

Dans le cadre de cette collaboration, plusieurs évènements nationaux et internationaux sur la recherche ont été organisés, tels que :

— les éditions successives de "Journée Doctorale des Sciences de l'Ingénieur" (JDSI'14, JDSI'15, JDSI'16 et JDSI'17), avec une forte participation de l'Université de Lorraine ;

— la conférence IEEE ICM 2015 (organisé du 20 au 23 décembre 2015 à Casablanca), sous la présidence du Dr. Kamal DAISSAOUI (Président de l'EMSI) et du Pr.Mohamad SAWAN (Directeur du Laboratoire PolySTIM, Montréal), ainsi que l'implication du Dr. Mohamed TABAA, du Pr. Abbas DANDACHE, du Pr. Fabrice MONTEIRO et du Pr. Fouad EL HAJ HASSAN (Université Libanaise).

Cette collaboration a également permis de développer le lien et la synergie entre les acteurs concernés par le programme d'enseignement de l'EMSI en les impliquant dans le programme d'enseignement et recherche, notamment via les formations, la mobilité des enseignants/chercheurs entre Casablanca et Metz, les stages de recherche proposés aux étudiants inscrits à l'EMSI dans les Masters délocalisés de l'Université de Lorraine, afin de les préparer à intégrer le monde de la recherche.

Publications

Khadija BOUSMAR, Fabrice MONTEIRO, Zineb HABBAS, Sofiene DELLAGI, Abbas DANDACHE, « Modelling industrial manufacturing problem using ILP solver : case of production analysis», 29ème conférence internationale IEEE ICM, 10-13 Décembre 2017, Beirut, Liban.

Khadija BOUSMAR, Fabrice MONTEIRO, Zineb HABBAS, Sofiene DELLAGI, Abbas DANDACHE, « A pure hardware k-SAT solver architecture for FPGA based on generic tree-search », 29ème conférence internationale IEEE ICM, 10-13 Décembre 2017, Beirut, Liban.

Khadija BOUSMAR, Fabrice MONTEIRO, Zineb HABBAS, Sofiene DELLAGI, Abbas DANDACHE, « Modelling industrial preventive maintenance using ILP solver », 6ème édition du workshop international sur l'Innovation et Nouvelles Tendances dans les Systèmes d'Information (INTIS), 24-25 Novembre 2017, Casablanca, Maroc.

Khadija BOUSMAR, Fabrice MONTEIRO, Zineb HABBAS, Sofiene DELLAGI, Abbas DANDACHE, « Modelling industrial problems using ILP solver : case of manufacturing and preventive maintenance », Journée Doctorale des Sciences de l'Ingénieur (JDSI), 04 Novembre 2017, Marrakech, Maroc.

Khadija BOUSMAR, Fabrice MONTEIRO, Zineb HABBAS, Sofiene DELLAGI, Abbas DANDACHE, « A pure generic hardware k-SAT solver architecture for FPGA », Journée Doctorale des Sciences de l'Ingénieur (JDSI), 10 Décembre 2016, Casablanca, Maroc.

Mohamed TABAA, Karim ALAMI, Abbas DANDACHE, Z. BENABBOU, Brahim CHOURI, Khadija BOUSMAR, « Initiation à la conception et réalisation d'un réseau de capteurs intelligents : Maison Intelligente>, 14ème journée pédagogique de la Coordination Nationale pour la Formation en Microélectronique et en nanotechnologies (CNFM), 29 novembre au 1 décembre 2016, Montpellier, France.

Khadija BOUSMAR, Fabrice MONTEIRO, Zineb HABBAS, Sofiene DELLAGI, Abbas DANDACHE, Mohamed TABAA, « The SATisfiability boolean using miniSAT solver for NP-complete problems », 4th day on information technologies and modeling TIM16, 2 juin 2016, Casablanca, Maroc.

Khadija BOUSMAR, Fabrice MONTEIRO, Zineb HABBAS, Sofiene DELLAGI, Abbas DANDACHE, « A new FPGA-based DPLL algorithm to improve SAT solvers », 27ème conférence internationale IEEE ICM, 20-23 Décembre 2015, Casablanca, Maroc.

Khadija BOUSMAR, Fabrice MONTEIRO, Zineb HABBAS, Sofiene DELLAGI, Abbas DANDACHE, « Développement d'outils d'aide à la décision dans la gestion du risque industriel », Journée Doctorale des Sciences de l'Ingénieur (JDSI), 18 Avril 2015, Casablanca, Maroc.

Résumé

Dans cette thèse, nous abordons un sujet dans le domaine du génie industriel se rapportant à la résolution d'un problème de décision, fondamental dans la théorie de la complexité et de la satisfiabilité propositionnelle, nommé SAT. Ce dernier est présenté généralement sous un formalisme mathématique, permettant de modéliser des problèmes complexes, tant académiques qu'issus du monde réel. Ces problèmes sont présentés sous une forme booléenne dans le but de tester leur faisabilité. Ils sont relatifs à plusieurs domaines applicatifs, tels que la vérification de matériels et logiciels, les télécommunications, la médecine, et notamment la planification. L'évolution et les progrès observés ces dernières années dans le domaine de la résolution des problèmes utilisant SAT, a permis de renforcer la conviction que ce domaine peut être encore plus prometteur dans la résolution des problèmes difficiles (complexes ou NP complexes) et qu'il faut lui accorder plus d'intérêt.

C'est dans cette optique, que nous nous sommes intéressés à l'appliquer sur des problèmes purement industriels afin de proposer des contributions dans un nouveau domaine d'application. L'objectif de cette thèse est de développer des outils d'aide à la décision pouvant être employés dans le domaine de la gestion du risque industriel. Bien que le formalisme SAT soit très puissant, dans la pratique, quand les problèmes ciblés sont de taille importante, les outils de résolution s'avèrent moins performants. Par conséquent le but de ces travaux de recherche est de développer une architecture matérielle rapide (avec une implémentation ciblée sur FPGA) permettant une accélération massive de la résolution grâce au niveau élevé de traitement parallèle de l'approche matérielle.

Dans cette thèse, deux aspects principaux sont étudiés et développés pour résoudre un problème de gestion des ressources de production industriel. Ces aspects sont d'une part, les principes de base de fonctionnement et de résolution d'un solveur générique paramétrable SAT, d'autre part, des méthodes adaptées au principe de fonctionnement retenu pour le solveur matériel. En effet, bien que ciblant des buts comparables à ceux de l'approche logicielle (optimisation du parcours de l'espace de recherche), l'approche matérielle requiert le développement de méthodes de résolutions spécifiques. Ces dernières ont été spécifiquement optimisées pour le domaine applicatif cible qui est celui de l'industrie. L'efficacité de l'approche matérielle développée a montré des résultats satisfaisant, point de vu du nombre de variables utilisés et temps de résolution sur les problèmes testés.

Mots-clés : *SAT, solveur SAT, gestion du risque industriel, problème combinatoire, architecture matérielle de calcul.*

Abstract

In this thesis, we address a topic in the field of industrial engineering related to solving a fundamental decision problem in the theory of complexity and propositional satisfiability called SAT. The latter is usually presented in a mathematical formalism, allowing the modelling of complex problems, both academic and from real world. These problems are presented in boolean form in order to check their feasibility. They relate to several application areas, such as hardware and software verification, telecommunications, medicine, and planning. The evolution and progress observed in recent years in the field of problem-solving using SAT has made it possible to reinforce the conviction that this field can be even more promising in solving difficult (complex or complex NP) problems and that more attention needs to be dedicated to it.

It is with this in mind that we have taken an interest in applying it to purely industrial problems in order to propose contributions in a new field of application. The objective of this thesis is to develop decision-support tools that can be used in the field of industrial risk management. Although the SAT formalism is very powerful, in practice, when the targeted problems are large, the resolution tools prove to be less effective. Therefore, the aim of this research is to develop a rapid hardware architecture (with FPGA-targeted implementation) that allows massive acceleration of resolution due to the high level of parallel processing of the hardware approach.

In this thesis, two main aspects are studied and developed to solve a problem of management of industrial production resources. These aspects are, on the one hand, the basic principles of operation and resolution of a generic SAT configurable solver and, on the other hand, methods adapted to the operating principle adopted for the hardware solver. Indeed, although targeting goals comparable to those of the software approach (optimization of the search space path), the material approach requires the development of specific resolution methods. These have been specifically optimised for the target application area of industry. The effectiveness of the material approach developed showed satisfactory results, point of view of the number of variables used and resolution time on the problems tested.

Keywords : *SAT, SAT Solver, industrial risk management, combinatorial problem, hardware computing architecture.*

Sommaire

Introduction générale

Problématique

Dans un contexte industriel, on peut définir le risque industriel comme la possibilité qu'un évènement accidentel se produise dans un site industriel et puisse (selon sa gravité et sa probabilité) mettre en jeu des procédés et/ou produits dangereux, entrainant dans la suite des conséquences graves sur le personnel, les biens, l'environnement et les riverains. Dès lors, pour limiter et éviter ces risques dangereux, des règlementations spécifiques ont été imposées en industrie, introduisant des contrôles réguliers afin de garantir la sécurité en éliminant au maximum les risques résiduels ainsi que ceux dépassant le seuil d'acceptabilité, normalement précisé dans les référentiels de sécurité. Une notion dite de « vision zéro » ou « risque zéro » a été proposée en Suède et en Suisse, puis dans d'autres pays, visant à supprimer tout apparition de ces événements, mais s'est malheureusement avérée utopique, vu leurs probabilités aléatoires et leurs gravités.

Un autre problème se situe dans les mauvaises interprétations des termes employés de risques, sécurité. La terminologie des risques est riche, mais les multiples nuances génèrent un blocage en matière de savoir-faire et de partage des connaissances. Prenons l'exemple d'un industriel voulant faire l'analyse d'un système pour identifier les risques globaux pouvant l'affecter : il se retrouve parfois bloqué, confronté aux différentes analyses faites par les sous-traitants des divers sous-systèmes, chacun employant un langage, une méthodologie et une terminologie qui lui sont propres.

De multiples méthodes peuvent être employés pour faire l'analyse du risque, mais la disparité entre les unes et les autres nuit généralement à la continuité du développement, du management et de la gestion des risques. Parmi celles-ci, on peut citer : APR, AMDEC, Arbre d'événements, Arbre de causes...

D'autres alternatives s'offrent aux industriels, basées sur des méthodes d'évaluation subjective des risques et utilisant des graphes de risques ainsi que des matrices de criticité.

Pour une aide à la décision des risques efficace, plusieurs facteurs doivent être réunis. On doit malheureusement constater le manque d'outils fiables et d'une terminologie unifiée pour l'ensemble des acteurs industriels. Les méthodes d'analyse des risques citées précédemment (APR, AMDEC, etc) se présentent plutôt comme des démarches interface de saisie d'analyses de risque que comme de véritables « Systèmes Interactifs d'Aide à la Décision » (SIAD), dont

l'industrie pourrait tirer grand bénéfice si l'existence et l'usage se développait.

De ce constat est issu notre intérêt pour le développement cet axe SIAD, le but étant à terme de pouvoir assurer dynamiquement la gestion du risque en fonction de l'évolution en temps réel des paramètres qui impactent l'estimation du risque. L'approche que nous proposons, combine les domaines des mathématiques et de l'informatique, pour modéliser le problème de risque industriel, et la conception d'architectures électroniques pour l'implantation sur technologie FPGA de calculateurs dédiées rapides (solveurs matériel). L'accélération matérielle offerte par un solveur matériel dédiée ouvre la porte à des vitesses de résolution bien plus élevées que celles traditionnellement autorisées par l'approche purement logicielle. Au cœur de notre approche, le choix s'est porté sur la conception d'un solveur dédié à la résolution de problème SAT (SATifiabilité), le problème de risque à résoudre devant être préalablement modélisé sous forme SAT avant d'être résolu sur le solveur.

Formellement, SAT est un problème fondamental de décision, très utilisé en théorie de la complexité et de la satisfiabilité propositionnelle. Il est adopté dans des disciplines diverses telles que la recherche opérationnelle ou l'intelligence artificielle. SAT recouvre un formalisme simple mais très expressif, qui permet de représenter un grand nombre de problèmes de décision difficiles, tant académiques qu'issus du monde réel.

Un problème SAT se présente comme une formule propositionnelle booléenne, sous forme de conjonction de disjonctions selon la terminologie mathématique, ou « produit de sommes » selon le vocabulaire des électroniciens. La résolution de SAT consiste à déterminer si le problème admet ou non une solution, c'est-à-dire, donnant la valeur booléenne *vrai* à la formule. L'importance du problème SAT a émergé lors de la démonstration faite par Stephen Cook en 1971 de son appartenance à la classe des problèmes NP (Non déterministe Polynomial) complets, le mettant ainsi dans la catégorie des problèmes majeurs en théorie de la complexité. Malgré la logique propositionnelle simple employée, sa complexité et sa résolution théorique restent particulièrement difficiles. L'intégration de SAT à la classe des problèmes NP-complets n'a rendu son étude théorique et pratique que plus importante encore.

Le formalisme SAT a connu un fort succès et est devenu une pratique courante dans la modélisation et la résolution de nombreux problèmes académique et issus du monde réel dans des domaines applicatifs très divers, tels que : vérification de matériels et logiciels, télécommunications, planification, déduction, automatique, bio-informatique, etc. Ceci est largement dû à la simplicité et l'élégance du formalisme SAT, et le fait que, comme beaucoup de travaux l'ont montré, il offre un compromis optimal entre efficacité algorithmique, méthode de résolution, puissance et facilité de représentation des problèmes NP-complets.

Organisation du mémoire

La suite de ce mémoire est composé de trois chapitre et d'une conclusion générale.

Le premier chapitre présente le domaine d'application industriel. Ce chapitre pose le contexte industriel et les terminologies employées, en levant les potentielles ambiguïtés relatives aux problèmes industriels. Il présente un état de l'art sur le management des risques industriels et la gestion des ressources de production. Le chapitre se termine par la présentation de quelques problèmes industriels types tels que la production et la gestion de la maintenance, qui seront modélisés vers la fin du chapitre à l'aide d'un formalisme mathématique linéaire, nommé : (ILP), la programmation linéaire en nombre entier qui sera aussi expliquée, détaillée et argumenté pour son utilisation dans ce premier chapitre.

Le deuxième chapitre situe le problème de « satisfiabilité », désigné le plus souvent par l'expression « problème SAT » , par la présentation des préliminaires étant la base de la logique propositionnelle, la notion des classes d'instances et la définition formelle de SAT afin de mieux se familiariser avec ce format mathématique. Ce chapitre se suit par la présentation et la formalisation des problèmes de types NP-Complets dont SAT fait partie, ainsi que l'illustration de quelques exemples de problèmes réels SAT. On continue avec les différentes méthodes de recherche de solutions utilisées telles que la recherche arborescente qui se suivra par les algorithmes de résolution utilisant cette méthode et étant définie en ordre chronologique. Puis on termine ce second chapitre par des heuristiques logicielles qui viennent compléter ces algorithmes afin d'accélérer et d'optimiser le temps de résolution.

Le troisième et dernier chapitre introduit les architectures matérielles et hybrides massivement parallèles dédiées à la résolution de SAT. Le développement réalisé dans les deux premiers chapitres nous a permis de mettre tout en œuvre pour ainsi déployer notre architecture et présenter le type de recherche arborescente utilisée. Cette architecture proposée nous a permis de montrer que la réduction des temps de résolution matérielle pour un problème industriel suivant le formalisme SAT est possible, tenant en compte le nombre de clauses et instances à tester par la favorisation de l'analyse parallèle arborescente qui est un bon moyen de parvenir à nos fins. Ces résultats obtenus sur FPGA, ont été présentés et discutés après l'implémentation de notre solveur proposé avant de conclure le chapitre.

Ce mémoire de thèse s'achève par une conclusion générale dans laquelle nous discutons l'ensemble de nos propositions et travaux au regard des objectifs initiaux de l'étude, puis quelques perspectives à court et moyen sur la poursuite de ces travaux.

Chapitre 1

Problématique industrielle : état de l'art, modélisation et contribution au problème de production

Sommaire

1.1 Introduction

Le monde industriel devient de plus en plus complexe, vaste et incertain, conduisant à un accroissement des besoins de supervision, de étude et d'analyse simultanée de tous les facteurs de performance. Jour après jour, de nouvelles activités, règlementations et normes apparaissent, dictées notamment par de nouveaux besoins des industriels et pour faire face au monde imprévisible dans lequel on vit, de toujours plus agressif et concurrentiel. Le besoin est croissant de développer et employer des méthodes et dispositifs permettant de représenter, interpréter, évaluer, maîtriser et gérer, à un coût minimal, les risques dans leur diversité (nature et gravité).

Le but de ce chapitre est d'investiguer la notion de risque qui émerge du domaine industriel auquel nous nous intéresssons en particulier, et d'analyser des problèmes industriels réels en fonction de la notion de risque. Une modélisation de ces problèmes sera présentée, en faisant ressortir sa complexité et la difficulté d'en proposer un résolution fiable et rapide. En effet, les problèmes que l'on cherche à résoudre aujourd'hui, qu'ils proviennent du domaine industriel (ou de tout autre domaine en général), sont toujours plus complexes, de par leur taille croissante et la diversité de facteurs à prendre en compte. L'approche habituelle est alors de chercher à simplifier le problème en le décomposant en sous-problèmes que l'on va chercher à traiter de manière indépendante. Si cette approche permet bien en général de réduire la complexité et d'accélérer la résolution, elle interdit l'accès à une solution globale optimale en ne permettant par la prise en compte simultanée de l'ensemble des contraintes du problème. C'est ce constat qui nous incite à proposer de nouvelles méthodes de résolution rapides.

Ce premier chapitre est divisé en deux parties.

La première commence avec la présentation des différentes notions associées à la sécurité. Sont tout d'abord introduits les 5 concepts de base de la sécurité : sûreté de fonctionnement, fiabilité, disponibilité, maintenabilité et sûreté. Sont ensuite exposées les notions de danger et de phénomène dangereux, complétées par les notions de gravité (insignifiant, marginal, critique et catastrophique) et de fréquence d'occurrence (invraisemblable, improbable, rare, occasionnel, probable et fréquent). Cette première partie se poursuit avec la définition du terme risque ainsi que sa classification et types, pour se terminer par une brève classification des méthodes d'analyse (déterministes, probabilistes, qualitatives et quantitatives) et la présentation de certaines d'entre elles : Analyse Préliminaire de Risque (APR), l'Analyse des Modes de Défaillances de leur Effets (AMDE), arbre de défaillance, etc.

La deuxième partie de ce chapitre aborde la modélisation des problèmes industriels liée à l'évaluation du risque. L'approche proposée pour la modélisation repose sur un formalisme appelé programmation linéaire en nombre entiers ou ILP (pour *Integer Linear Programming*, qui permet de manière assez simple et directe la transcription formelle des problèmes abordés. Ce formalisme est utilisé comme passerelle vers le formalisme SAT qui est présenté ultérieurement (de même que la technique de passage de du modèle ILP au modèle SAT).

Le chapitre se termine sur un exemple de modélisation ILP d'un problème de production et maintenance prédictive, où sont considérés les contraintes de gestion de ressources et les risques de survenue d'événements pouvant provoquer des dommages dangereux. Une analyse des limites de cette modélisation ILP complète l'exemple.

1.2 Sécurité

Un absence de risque inacceptable signifie que le système est bien sécurisé. Cette définition est appuyée par la norme [ISO/CEI Guide 2, 1986], la gravité et la sécurité y étant associées : « la sécurité est l'absence de risque de dommage inacceptable » [ISO/CEI Guide 2, 1986]. Cette relation entre le risque et gravité est à nouveau réaffirmée dans [ISO/CEI Guide 51, 1999] : « la sécurité est l'absence de risque inacceptable », pour ajouter la probabilité d'occurrence et la quantifier.

Pour résumer, la sécurité représente l'absence de risques inacceptables et de dangers susceptibles de créer des dommages majeurs. Elle joue un rôle très important dans l'acceptabilité d'un risque par la mesure de niveau de confiance. Après ces définitions générales de sécurité, nous commençons dans les sous-sections suivantes par présenter les 5 concepts de base (sûreté de fonctionnement, fiabilité, disponibilité, maintenabilité, sûreté) liés à la sécurité avant de passer aux concepts de risques et dangers.

1.2.1 Sûreté de fonctionnement

On peut définir la sûreté de fonctionnement comme l'assurance de la réussite d'une mission d'une entité en garantissant son bon fonctionnement. C'est pour cela que l'on utilera constamment l'expression « fonction requise » qui affirme d'après la norme [CEI 50(191), 1990] que l'accomplissement d'une ou plusieurs fonctions est obligatoire afin de délivrer un service donné.

On considère la sûreté de fonctionnement comme un moyen d'atteindre la maîtrise des risques en suivant des démarches, respectant des méthodes et utilisant des outils pour y arriver. Selon Mortureux [1], « c'est un atout majeur du concept de sûreté de fonctionnement de réunir des approches motivées par la fiabilité, la disponibilité, la maintenabilité et la sécurité, mais c'est un piège de vouloir réduire à une valeur le résultat de ces démarches ». À ces définitions, s'ajoute celle de Heurtel [2], qui définit la sûreté de fonctionnement comme « utilisant un ensemble de concepts et de méthodes garantissant un bon fonctionnement d'une entité et en étant une activité d'ingénierie qualitative et quantitative ».

La sûreté de fonctionnement est exprimée par un ensemble de paramètres mesurables utilisant les espérances mathématiques et des approches stochastiques. Ci-dessous, sont énumérées les plus utilisés dans le domaine de la gestion des risques.

- **MTTF (Mean Time to Failure) :** espérance mathématique de la durée de bon fonctionnement avant l'apparition de la première défaillance. Il est a noter que, lorsque le taux de défaillance λ est constant, alors MTTF $= 1/\lambda$.
- **MTTR (Mean Time To Repair) :** espérance mathématique de la durée de réparation.
- **MUT (Mean Up Time) :** espérance mathématique de la durée de fonctionnement.
- **MDT (Mean Down Time) :** espérance mathématique de la durée d'indisponibilité.
- **MTBF (Mean Time Between Failures) :** espérance mathématique de la durée entre deux défaillances consécutives.

1.2.2 Fiabilité

Selon Sallak, Simon et Aubry [3], l'évaluation des différentes architectures d'un système, en mettant en parallèle les performances requises par le biais de données statistiques, est le but des études prévisionnelles de fiabilité *reliability*) d'un système donnée.

D'après la norme [CEI 50(191), 1990], la fiabilité est « l'aptitude d'une entité à accomplir une fonction requise, dans des conditions données, pendant un intervalle de temps donné ». Ceci est vrai si l'entité accomplit la fonction requise à l'instant 0 d'un intervalle donnée. Par contre, une défaillance surgit par l'arrêt de cette fonction ou aptitude.

La fonction de fiabilité est exprimée analytiquement par

$$\mathrm{R}(t) = \text{Prob}(E \text{ non défaillante sur } [0, t]) \tag{1.1}$$

où E et $[0, t]$ sont respectivement l'entité et l'intervalle de temps considérés.

Le taux de défaillance est défini par λ, la probabilité de défaillance entre t et $t + dt$ étant donnée par $\lambda \cdot dt$, avec aucune défaillance apparue à l'instant t. On notera que le taux de défaillance λ a souvent avoir une forme de type « baignoire » comme représenté dans la figure 1.1 et que ce taux peut changer fortement en fonction des composants électroniques. Généralement, en contexte industriel, le taux de défaillance est introduit sous forme de vitesse d'occurrence des pannes.

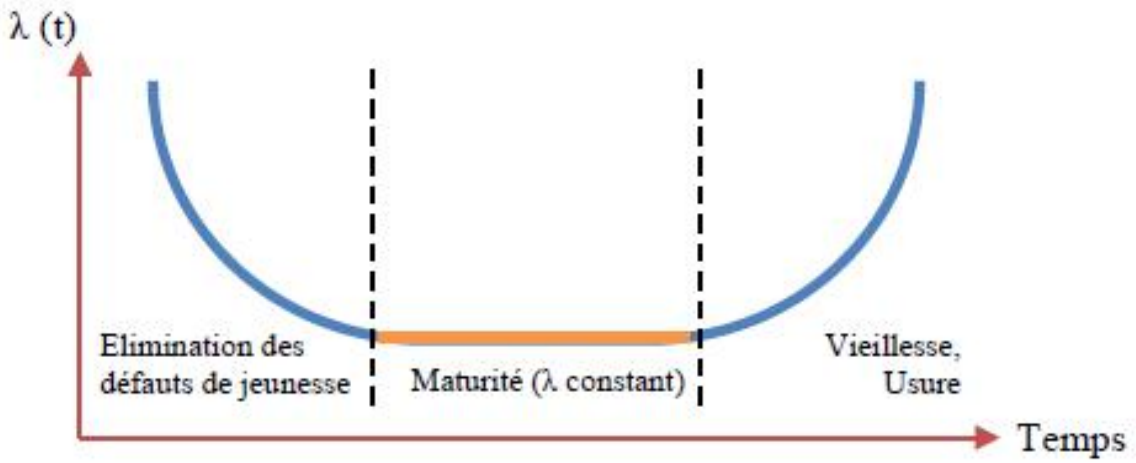

FIGURE 1.1 – Évolution du taux de défaillance d'un composant.

1.2.3 Disponibilité

D'après la norme [CEI 50(191), 1990], la disponibilité (*availability*) dépend de la fiabilité, de la maintenabilité et de la logistique de maintenance. On peut également admettre, selon la norme [CEI 61069, 1996] la définition suivante : « aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données, à un instant donné ou pendant un intervalle de temps donné, en supposant que la fourniture des moyens nécessaires est assurée ». Cette définition est conforme à ce qui avait été défini dans la norme précédente.

On ajoute à cette définition ce que Mortureux [1] dit à ce propos : « la disponibilité est une synthèse de la fiabilité et la maintenabilité ; c'est la proportion de temps passé en état de remplir les fonctions requises dans des conditions données ».

Pour calculer la disponibilité, on se base sur la probabilité $\mathrm{A}(t)$ d'être en état à un instant t donné, c'est-à-dire,

$$\mathrm{A}(t) = \mathrm{Prob}(E \text{ non défaillante à l'instant } t) \tag{1.2}$$

La disponibilité étant mesurée par une probabilité, on ne peut jamais garantir pour les défaillances dangereuses une probabilité nulle. C'est la raison pour laquelle l'on doit impéra-

tivement les détecter au plus vite afin d'interrompre leur propagation. Sécurité et disponibilité représentent deux concepts généralement difficiles à concilier.

1.2.4 Maintenabilité

D'après la norme [CEI 50(191), 1990] : « dans des conditions données d'utilisation, aptitude d'une entité à être maintenue ou rétablie dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est accomplie dans des conditions données, avec des procédures et des moyens prescrits ». La maintenabilité mesure donc la capacité à assurer des fonctions requises sous des conditions proposées, par une maintenance donnée.

La maintenabilité est la probabilité $\mathrm{M}(t)$ d'être en état de fonctionnement à l'instant t, sachant que l'équipement était en panne à la date $t = 0$. Formellement, la maintenabilité peut être définie de manière analytique comme suit :

$$\mathrm{M}(t) = \mathrm{Prob}(E \text{ est réparée sur } [0,t] \ / \ E \text{ est en panne à la date } t = 0) \tag{1.3}$$

où E et $[0,t]$ sont respectivement l'entité et l'intervalle de temps considérés. Si le taux de réparation μ est constant, alors $M(t) = e^{-\mu t}$.

Dans la norme [CEI 50(191), 1990], la logistique de maintenance (*Maintenance support performance*) est définie comme suit : « aptitude d'une organisation de maintenance à fournir sur demande, dans des conditions données, les moyens nécessaires à la maintenance d'une entité conformément à une politique de maintenance donnée ».

1.2.5 Sûreté

D'après la norme [CEI 61069, 1996] dédiée aux processus industriels, la sûreté est « l'assurance fournie par le système de sa capacité à refuser toute entrée incorrecte ou tout accès non autorisé et à pouvoir éventuellement en informer ». Il est important de noter que « sûreté de fonctionnement » et « sûreté » recouvrent deux définitions aux significations différentes, qu'il ne faut pas confondre, bien que s'inscrivant dans le sens large du terme « sûreté ».

Les nuances des termes « sécurité » et « sûreté » peuvent ici encore créer des malentendus. Quand on parle d'accidents, l'expression généralement employée est « sécurité des installations », visant à d'empêcher l'apparition de dommages. Les attaques externes volontaires, mais pas uniquement, sont quant à elles normalement liées à la « sûreté ». Les intrusions malveillantes et les malveillances internes en font partie également partie.

De même, deux types de sécurité sont distingués dans les systèmes informatiques, comme les indentifie J.-L. Laprie [4] : une sécurité innocuité assimilable à la définition ci-dessus du terme sécurité (*safety* en anglais) et une sécurité confidentialité, proche de la définition de sûreté donnée par la norme [CEI 61069, 1996].

1.3 Notions de danger et de phénomène dangereux

1.3.1 Danger

La notion de danger peut se définir de plusieurs manières, mais en général, elle reste une propriété intrinsèque inhérente à un type d'évènement ou entité qui peut entraîner des conséquences sérieuses et des dommages (cf. .tableau 1.1).

TABLE 1.1 – Définitions de la notion de danger.

Source	Définition
[GT Méthodologie, 2003]	La notion de danger définit une propriété intrinsèque d'une substance (ex. : butane, chlore), d'un système technique (ex. : mise sous pression d'un gaz), d'une disposition (ex. : élévation d'une charge), d'un organisme (ex. : microbes), etc., de nature à entraîner un dommage sur un « élément vulnérable ». Sont ainsi rattachées à la notion de « danger », les notions d'inflammabilité/d'explosivité, de toxicité, de caractère infectieux, etc., inhérent à un produit et celle d'énergie disponible (pneumatique ou potentielle) qui caractérisent le danger.
[BSI OHSAS 18001, 2005]	Situation, condition ou pratique qui comporte en elle-même un potentiel à causer des dommages aux personnes, aux biens ou à l'environnement. Une source ou une situation pouvant nuire à un individu par blessure ou atteinte à la santé, dommage à la propriété et à l'environnement du lieu de travail ou une combinaison de ces éléments.
[Directive 96/82/EC (SEVESO II), 9 décembre 1996]	Propriété intrinsèque d'une substance dangereuse ou d'une situation physique de pouvoir provoquer des dommages pour la santé humaine et/ou l'environnement.
[NF EN 61508, décembre 1998]	Le danger désigne une nuisance potentielle pouvant porter atteinte aux biens (détérioration ou destruction), à l'environnement, ou aux personnes.
[AQS-GT OORS, mars 1996]	État ou situation comportant une potentialité de dommages.
[CEI 300-3-9, 1995]	Source potentielle de dommage.

1.3.2 Phénomène dangereux

Un phénomène dangereux peut être défini comme un processus de matérialisation du danger et qui peut produire des accidents. Des définitions issues des normalisation sont présentées dans le tableau 1.2.

TABLE 1.2 – Définitions de la notion de phénomène dangereux.

Source	Définition
[GT Méthodologie, 2003]	Libération d'énergie ou de substances produisant des effets susceptibles d'infliger un dommage à des cibles (ou éléments vulnérables) vivantes ou matérielles, sans préjuger de l'existence de ces dernières.
[ISO 14971, 2000], [CEI 300-3-9, 1995], [EN 292/ISO 12100, 1995]	Source potentielle de dommage.

1.4 Notions de gravité, fréquence d'occurence et exposition

1.4.1 Gravité

La gravité se définit comme l'importance des choses qui ne peuvent se considérer avec légèreté, et qui par la suite peuvent présenter des conséquences fâcheuses. Ces dernières sont classifiées en fonction de leurs importances et caractérisées par leur gravité. Ce travail est généralement fait par des experts du domaine.

La définition des classes de gravité est faite, par les experts, en nombre pair afin d'éviter la position médiane des classifications impaires faites en général. De plus, l'utilisation d'un lexique et d'une terminologie communs doivent être établis dans le but de bien communiquer et faire passer les informations, afin d'éviter tout malentendu en cas d'audit ou demande d'avis d'expert. Par ailleurs, pour éviter l'apparition de ces conflits et mauvaises interprétations, les industriels adoptent tout naturellement une dénomination des classes de gravité en termes de niveaux (0, 1 , 2, ...).

On ne peut pas parler de gravité dans le domaine du risque professionnel sans évoquer les dommages causés à l'Homme. Les différents échelons de gravité peuvent être présentés comme suit par le biais de la norme [ISO 14971, 2000] :

— **négligeable :** incident qu'il ne vaut pas la peine de prendre en compte et qui n'exige aucun acte médical ;

— **minime :** incident de peu d'importance, tels que par exemple, de légères blessures qui ne nécessitent pas un traitement médical mais relevant des premiers soins ;

— **mineure :** blessures ou maladies mineures de moindre importante mais qui nécessitent un traitement médical ;

— **majeure :** blessures ou maladies graves, avec une infirmité permanente ;

— **catastrophique :** décès d'une ou plusieurs personnes.

Cette notion de gravité ne couvre pas que les préjudices portés à l'homme et au système pour le domaine industriel, mais l'environnement en fait également partie. Le tableau 1.3 présente les quatre éhellons de gravité de la norme ferroviaire [NF EN 50126, janvier 2000].

TABLE 1.3 – Échellons de gravité d'après la norme [NF EN 50126].

Gravité	**Conséquences pour les personnes ou l'environnement**	**Conséquences pour le service**
Insignifiant	Blessures légères d'une personne	
Marginal	Blessures légères et/ou menace grave pour l'environnement	Dommages mineurs pour un système
Critique	Un mort et/ou une personne grièvement blessée et/ou des dommages graves pour l'environnement	Perte d'un système important
Catastrophique	Des morts et/ou plusieurs personnes gravement blessées et/ou des dommages majeurs pour l'environnement	Dommages graves pour un ou plusieurs systèmes

1.4.2 Fréquence d'occurence

Pour calculer une fréquence d'occurrence d'un événement, une estimation est faite du nombre moyen d'occurrences sur une période de temps donnée (année, jour, mois, etc.) à partir de situations connues.

Le tableau 1.4 présente les classes et niveaux de fréquences proposées par la norme [NF EN 50126, janvier 2000].

Le tableau 1.5 présente une simplification de la métrique de fréquences d'occurrence adoptée dans le cadre de l'APR du système de contrôle/commande ERTMS [GTR 55, 2000], présenté avant simplification dans le tableau 1.4 de la métrique proposée dans la norme [NF EN 50126].

Il faut cependant mentionner que les termes utilisés dans ces deux tableaux admettent une marge de latitude dans leur interprétation. Par exemple, « plusieurs fois », « plusieurs reprises » ou « fréquemment » peuvent tout aussi bien renvoyer à 5 comme à 2000 occurrences. D'où une absence d'attribution de probabilités à ces niveaux.

TABLE 1.4 – Échelle de fréquence d'occurences d'après la norme [NF EN 50126].

Niveau	Description
Invraisemblable	On peut admettre que la situation dangereuse ne se produira pas, voire est très improbable.
Improbable	On peut admettre que la situation dangereuse est peu susceptible d'avoir lieu mais qu'elle le peut exceptionnellement.
Rare	À un moment donné du cycle de vie du système, la situation dangereuse est raisonnablement susceptible de se produire.
Occasionnel	On peut s'attendre à ce que la situation dangereuse se produise plusieurs fois.
Probable	On peut s'attendre à ce que la situation dangereuse se produise souvent, et à plusieurs reprises.
Fréquent	La situation dangereuse est fréquente, donc continuellement présente.

TABLE 1.5 – Adaptation de l'échelle de fréquence d'occurences de la norme [NF EN 50126].

Niveau	Description
Invraisemblable	Occurence extrêmement invraisemblable durant la vie d'un système. $\Rightarrow \quad \leq 10^{-9}/h$
Rare	Occurence invraisemblable mais possible durant la vie d'un système. $\Rightarrow \quad > 10^{-9}/h$
Occasionnel	Occurence vraisemblable à plusieurs reprises durant la vie d'un système.
Fréquent	Occurence vraisemblablement fréquente durant la vie d'un système.

1.5 Notions de risque

1.5.1 Risque

En 1993, R. Flanagan et G. Norman [5] citent le mot « risque » dans leur livre « Risk Management and Construction » comme étant moderne, où il est compris, évalué, représenté, géré, perçu de manière différente. L'utilisation du mot « Risk » a été faite par les Anglos-Saxons vers le 17ème siècle pour désigner des situations dangereuses où un événement grave peut se produire.

Ce risque est généralement défini par plusieurs facteurs qui jouent un rôle très important pour déterminer l'existence d'un événement grave ou pas. Parmi ces paramètres, on peut citer la gravité et l'occurrence, qui seront expliquées dans les sous-sections suivantes.

1.5.2 Classification du risque

Pour classer les risques, une matrice de criticité Gravité/Occurrence est utilisée. Celle-ci croise les niveaux de gravité et de probabilité d'occurrence pour positionner les zones de

risques.

TABLE 1.6 – Matrice de criticité (G/O) de la norme [NF EN 50126].

	Insignifiant	Marginal	Critique	Catastrophique
Invraisemblable	négligeable	négligeable	négligeable	négligeable
Improbable	négligeable	négligeable	acceptable	acceptable
Rare	négligeable	acceptable	indésirable	indésirable
Occasionel	acceptable	indésirable	indésirable	indésirable
Probable	acceptable	indésirable	inacceptable	inacceptable
Fréquent	indésirable	inacceptable	inacceptable	inacceptable

Le tableau 1.6, proposé par la norme [NF EN 50126, janvier 2000], présente un tel tableau. À partir de cette répartition, les risques sont regroupés en différentes catégories : risques maîtrisés, risques maîtrisables et risques non maîtrisables.

Les risques maîtrisés sont ceux qui regroupent les risques négligeables et acceptables, tandis que les risques tolérables et indésirables font partie des risques maîtrisables. Les risques non maîtrisables sont les risques inacceptables et résiduels. Ceci est résumé dans la figure 1.2.

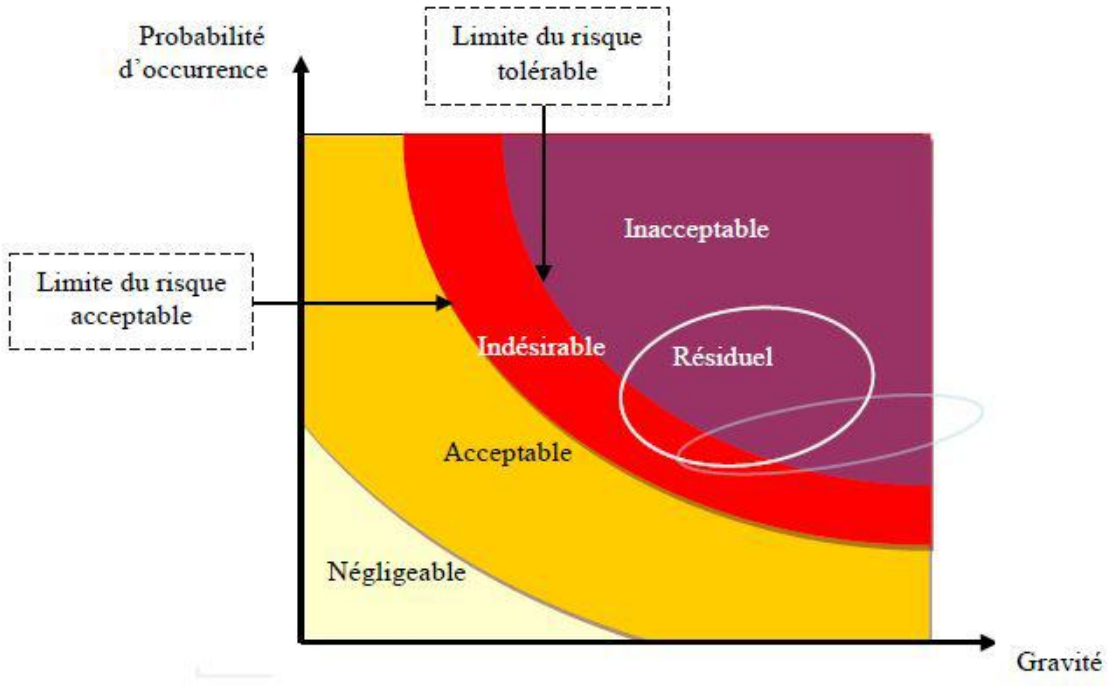

FIGURE 1.2 – Classification des risques.

La probabilité d'occurrence évaluée au quotidien joue un rôle très important dans la définition des risques, quels que soient leurs gravités.

1.5.2.1 Risques maîtrisés

Risque négligeable

Dans l'évaluation d'un système, pour mettre en évidence les risques présents, les risques négligeables ne sont généralement pas pris en compte. Ceux-ci peuvent être définis comme suit :

— **[GT Aspects sémantiques du risque, 1997] :** un risque est négligeable s'il est inférieur à un certain seuil, par exemple 10^{-9} par an ; il s'agit d'un risque dont on ne se soucie pas de l'occurrence au quotidien ;

— **[HSE, 1992] :** le risque négligeable fait référence à un niveau de risque dont l'occurrence est de l'ordre de 1 par million et par année et au-dessous, et dont la possibilité de réalisation n'affecte pas la vie courante.

Risque acceptable

Un risque est acceptable s'il peut être considéré comme insignifiant, même avec une probabilité d'occurrence faible. Considérons l'exemple des trains : on les prend même si on sait qu'il existe une probabilité non nulle de déraillement ou autre.

Dans la littérature, on peut classer les définitions de risque acceptable en deux catégories : la première est celle où les valeurs de la société jouent sur l'acceptabilité du risque, alors que la deuxième est plutôt basée sur la prise de décision (voir [ISO/CEI Guide 73] et [GT 7 – CEI]).

— **[AQS-GT OORS, mars 1996] :**
  — risque acceptable pour les personnes : un risque n'est accepté que s'il y a contrepartie et/ou s'il est inférieur au risque déjà encouru ;
  — risque acceptable pour les entreprises : un risque acceptable ne se conçoit que dans le cadre résiduel, après la mise en œuvre des mesures de prévention et de protection notamment celles imposées par la législation et la réglementation en vigueur ;

— **[GT Aspects sémantiques du risque, 1997] :**
  un risque est acceptable en référence à un objectif de sécurité donné ; il n'est acceptable que s'il est inférieur à un seuil donné (par exemple, 10^{-5} par an) ; c'est un risque avec lequel on consent à vivre en contrepartie d'un bénéfice et dans la mesure où il est contrôlé ;

— **[GT 7 – CEI] :**
  risque acceptable : valeur d'un risque résultant d'une décision explicite établie de façon objective ; il est parfois préférable, pour certaines branches d'activité, de parler de risque admissible ou de risque limite ;

— **[ISO/CEI Guide 73, 2002] :**
l'acceptation du risque dépend des critères de risques retenus par la personne qui prend la décision ;

— **[ISO/CEI Guide 51, 1999] :**
un risque accepté dans un contexte donné basé sur des valeurs courantes de notre société ;

— **[Laurant, 2003] :**
traduit la notion relative à un risque intégré tel que dans le contexte de la vie courante.

1.5.2.2 Risques maîtrisables

Risque tolérable

Un risque tolérable se définit comme étant un risque pouvant être pris en considération avec vigilance.

D'après le [GT Méthodologie], « la tolérance du risque résulte d'une mise en balance des avantages et des inconvénients (dont les risques) liés à une situation, situation qui sera soumise à révision régulière afin d'identifier, au fil du temps et chaque fois que cela sera possible, les moyens permettant d'aboutir à une réduction du risque ». Plusieurs définitions ont été données dans la littérature pour définir les risques tolérables :

— **[HSE, 1992] :**
risque que l'on ne puisse pas considérer comme négligeable ou comme quelque chose que l'on peut ignorer mais que l'on doit garder présent à l'esprit et pour lequel on doit engager des mesures de réduction aussitôt qu'il est possible ;

— **[Laurant, 2003] :**
le risque toléré traduit, à condition d'en retirer certains bienfaits, la volonté de vivre avec les risques que l'on ne saurait ni ignorer, ni considérer comme négligeables, mais avec la confiance qu'ils sont correctement maîtrisés ;

— **[GT Méthodologie, 2003] :**
le risque tolérable est le résultat de la recherche d'un équilibre optimal entre une sécurité absolue idéale et les exigences technico-économiques ;

— **[NF EN 61508, décembre 1998], [EN 292/ISO 12100, 1995] :**
risque accepté dans un certain contexte et fondé sur les valeurs actuelles de la société ;

— **[ISO/CEI Guide 51, 1999] :**
risque accepté dans un certain contexte et fondé sur des valeurs admises par la société.

Risque indésirable

Un risque indésirable est défini comme étant un risque tolérable moyennant un contrôle de suivi et des mesures appropriées.

— **[NF EN 61508, décembre 1998] :**
risque indésirable, tolérable uniquement s'il est impossible de réduire le risque ou si le coût de la réduction est disproportionné par rapport à l'amélioration possible.

Le risque indésirable est souvent lié au risque tolérable, vis-à-vis de la définition donnée par la norme [NF EN 61508].

1.5.2.3 Risques non maîtrisables

Risque résiduel

Un risque résiduel est un risque qui demeure malgré les différentes mesures possibles prises. Le risque résiduel est défini de deux façons par le [GT OORS] de l'AQS : les définitions données ci-dessous sont semblables hormis la toute première qui ne prend pas en compte toutes les mesures possibles prises avant la qualification du risque résiduel.

— **[GT Aspects sémantiques du risque, 1997] :**
  — risque qui subsiste après avoir appliqué des mesures de réduction ;
  — risque qui subsiste après avoir appliqué toutes les mesures de réduction disponibles ;

— **[AQS-GT OORS, mars 19967] :**
risque qui subsiste quand on a fait « de son mieux » en fonction du « possible actuel » ;

— **[NF EN 61508, décembre 1998], [CEI 1050, février 1991] :** risque restant après que toutes les mesures de prévention aient été prises ;

— **[ISO/CEI Guide 51, 1999] :**
risque subsistant après que des mesures de prévention aient été prises ;

— **[ISO/CEI Guide 73, 2002] :**
risque subsistant après le traitement du risque.

Risque inacceptable

Un risque inacceptable est un risque résiduel non tolérable.

1.6 Management des risques

Des processus intégrant plusieurs activités essentielles pour la sécurité sont généralement respectés pour le management et la gestion des risques. Une fois encore, comme pour les autres définitions, ces termes sont nuancés, et donnent l'impression qu'ils désignent la même étude. Ci-dessous, quelques définitions issues de la littérature.

— **[ISO/CEI Guide 73, 2002] :**
activités coordonnées visant à diriger et piloter un organisme vis-à-vis du risque ; le management du risque inclut typiquement l'appréciation du risque, le traitement du risque, l'acceptation du risque et la communication relative au risque ;

— **[Barbet, mars 1996] :**
la démarche de gestion des risques consiste à planifier, acquérir les informations, modéliser l'exposition du système aux risques et enfin conduire le système ;

— **[CEI 300-3-9, 1995] :**
application systématique des politiques de gestion, des procédures et des usages aux tâches d'analyse, d'évaluation et de maîtrise du risque.

D'après le guide [ISO/CEI Guide 51, 1999], l'appréciation des risques est l'ensemble du processus d'analyse et d'évaluation des risques, alors que le guide [ISO/CEI Guide 73, 2002] présente plutôt l'évaluation de l'acceptabilité des risques et non pas des risques eux-mêmes.

1.7 Analyse de risque

L'analyse de risque est faite par la collecte d'informations afin d'identifier les entités sources et cibles de danger pour estimer le risque.

— **[Larousse, 2005] :**
analyse : étude minutieuse, précise faite pour dégager les éléments qui constituent un ensemble, pour l'expliquer, l'éclairer (ex : faire l'analyse de la situation) ;

— **[ISO/CEI Guide 73, 2002] :**
utilisation systématique d'informations pour identifier les facteurs de risque et pour estimer le risque ;

— **[ISO/CEI Guide 51, 1999] :**
utilisation des informations disponibles pour identifier les phénomènes dangereux et estimer les risques.

1.8 Maîtrise des risques

La maîtrise des risques (*risk control*) est un processus consistant à évaluer la gravité et la fréquence des risques. D'une manière générale, la maîtrise des risques vise à les ramener sous un seuil d'acceptabilité par les différentes démarches de la gestion et décision.

— **[ISO/CEI Guide 73, 2002] :**
maîtrise : possibilité d'agir sur quelque chose ; fait de le dominer techniquement, intellectuellement, scientifiquement ; sûreté de l'exécution dans un domaine technique ou artistique ;

— **[ISO/CEI Guide 73, 2002] :**
actions de mise en oeuvre des décisions de management du risque ; la maîtrise du risque peut impliquer la surveillance, la réévaluation et la mise en conformité avec les décisions ;

— **[GT Aspects sémantiques du risque, 1997] :**
ensemble des disciplines concourant à la réduction et au contrôle du risque, incluant l'évaluation et la gestion du risque.

1.9 Classification des méthodes d'analyse de risques

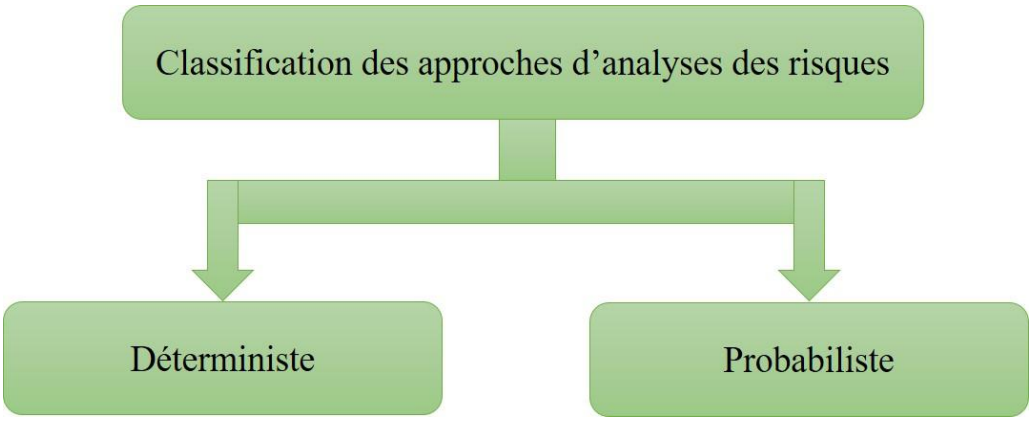

FIGURE 1.3 – Approches d'analyse des risques.

1.9.1 Approche déterministe

L'approche déterministe est utilisée dans des domaines à « moindre risque significatif » tels que les domaines militaire, nucléaire, transports guidés et autres... Cette approche consiste à déduire les actions pouvant créer des conséquences et accidents en déterminant le pire cas

possible (*worst case*) en allouant une gravité supérieure aux conséquences éventuelles.

1.9.2 Approche probabiliste

L'approche probabiliste, comme son nom l'indique, est basée sur le calcul de probabilités relatives à l'occurrence d'événements du processus matérialisant un scénario d'accident donné. L'approche déterministe décide du dispositif de défense en profondeur. L'approche probabiliste intervient par la suite, pour l'analyser. On constate alors qu'elle fait partie des approches complémentaires, afin d'appuyer l'approche déterministe.

1.9.3 Méthodes quantitatives

Les méthodes quantitatives sont des méthodes de recherche utilisant des outils d'analyse mathématique et statistique, dans le but de décrire, d'expliquer et prédire des phénomènes sous forme de variables mesurables. Elles sont généralement utilisées pour évaluer la sûreté de fonctionnement et entre autres la sécurité. Ces méthodes reposent, comme approche probabiliste, sur le calcul de probabilités telles que l'estimation quantitative de la probabilité d'occurrence d'un événement possible, ou sur des modèles différentiels tels que les automates à états finis, les chaines de Markov, les réseaux de Pétri, etc.

Les méthodes quantitatives présentent de nombreux avantages, car elles garantissent :

— l'évaluation de la probabilité des composantes de la sûreté de fonctionnement ;

— l'apport d'une aide pour mieux juger du besoin d'une amélioration de la sécurité ;

— une meilleure concertation et entente entre différents opérateurs en matière de sécurité (sous-systèmes travaillant ensemble) ou services et équipes (maintenance, etc.) ;

— l'analyse des risques ;

— la fixation des données de sécurité ;

— l'examen de l'acceptabilité des risques en prenant en compte la durée des situations dangereuses, la répétition des contrôles, etc. ;
— la comparaison et l'ordonnancement des étapes à respecter, en mettant en tête celles qui réduisent les risques.

1.9.4 Méthodes qualitatives

Les méthodes qualitatives regroupent un ensemble de méthodes de recherche utilisées dans les études qualitatives. Parmi les méthodes utilisées dans le domaine industriel, l'APR,

l'AMDEC, l'Arbre de Défaillances et l'Arbre d'Evénements représentent des méthodes qualitatives qui font de la classification des risques ou parfois des estimations de fréquences d'occurrence.

1.10 Méthodes d'analyse de risques

Dans cette section, une brève présentation des méthodes d'analyse de risque sera faite.

1.10.1 Analyse Préliminaire de Risque

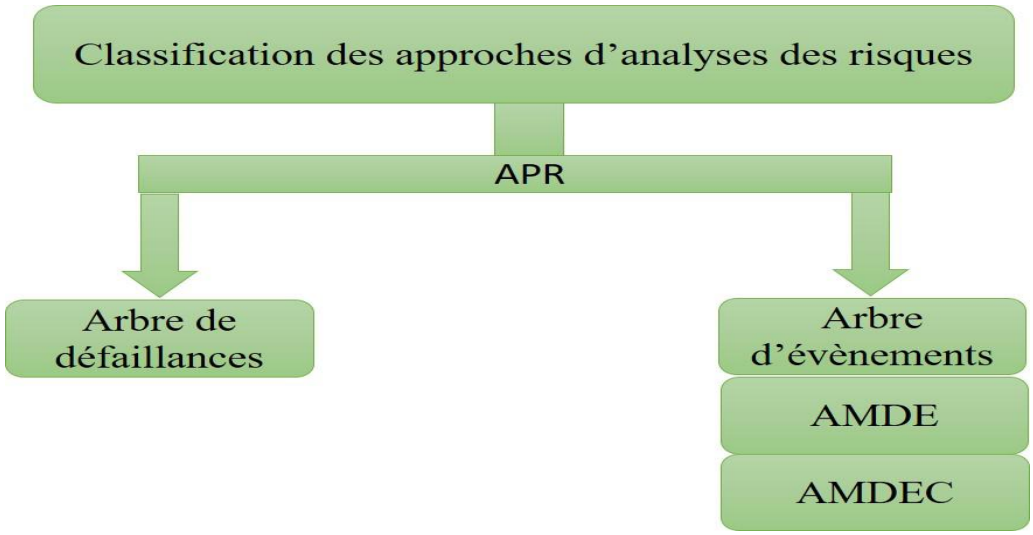

FIGURE 1.4 – Classification des approches de risque.

Au début des années 1960, l'Analyse Préliminaire de Risque (Danger) a été développée généralement pour les domaines aéronautiques et militaires.

D'après la norme [CEI 300-3-9, 1995], « l'APR est une technique d'identification et d'analyse de la fréquence du danger qui peut être utilisée lors des phases amont de la conception pour identifier les dangers et évaluer leur criticité ».

Cette méthode d'analyse utilise des listes (*check-lists*) de contrôles d'entités et évènements dangereux spécifiques aux différents domaines examinés afin de faire ressortir tous les incidents dangereux susceptibles de se produire, causant un dommage ou une situation dangereuse.

L'APR est une méthode préliminaire, qui a besoin d'être complétée par d'autres méthodes d'analyse, telles que l'arbre de défaillance, l'AMDE ou l'AMDEC, qui seront décrites dans les sous-sections suivantes.

L'APR ne met en avant généralement que les situations dangereuses qui peuvent se produire dans un système. Les autres méthodes étant employées par la suite afin d'examiner en détail la raison du déclenchement des accidents.

1.10.2 Analyse des Modes de Défaillance, de leurs Effets (AMDE) et de leur Criticité (AMDEC)

L'AMDE a été utilisé pour la première fois en 1960 dans l'industrie aéronautique. Depuis, elle s'est répendue dans différents domaines, pour évoluer ensuite en AMDEC, en relation avec la criticité des défaillances.

Pour mieux définir cette méthode d'analyse, on utilise la norme [CEI 300-3-9, 1995], pour laquelle l'AMDE est une technique fondamentale d'identification et d'analyse de la fréquence des dangers, qui vérifie tous les modes de défaillances d'un équipement donné et leurs effets, tant sur les autres composants que sur le système lui-même.

Pour résumer les étapes de traitement et d'analyse requises dans l'AMDE ou l'AMDEC pour évaluer la sécurité et la fiabilité d'un système, on note la nécessité de :

— déterminer les modes de défaillance ;

— déterminer les déclencheurs éventuels de chaque mode ;

— évaluer les conséquences élaborées ;

— évaluer, pour l'AMDEC, la criticité des conséquences.

1.10.3 Analyse par Arbre de Défaillances (FTA – *Fault Tree Analysis*)

Comme pour les méthodes d'analyse précédentes, l'année 1960 fut l'année d'élaboration des méthodes d'analyse des situations dangereuses. L'analyse d'arbre de défaillance s'y ajoute également, du fait de la compagnie américaine « Bell Phone » où elle a été utilisée pour la première fois dans l'étude de la sécurité des systèmes militaires, en évaluant la sécurité des tirs de missiles.

Comme son nom l'indique, c'est une méthode qui se base sur une représentation graphique arborescente des différents détails d'un événement suspect. Elle est généralement utilisée pour identifier les raisons qui ont présidé à l'apparition de cet événement.

L'arbre de défaillance se présente comme une méthode complète vu qu'elle dégage tous les détails lors de l'apparition d'un évènement dangereux, en établissant un lien entre les différents facteurs pouvant causer la situation, tels que les erreurs humaines, les facteurs

environnementaux, les défaillances d'équipement et les facteurs organisationnels.

Comme pour l'AMDE et l'AMDEC, l'arbre de défaillance suit des étapes pour dégager les risques et analyser un système :

— détermination du système et de ses frontières ;

— détermination des événements identifiés, par des méthodes telle que l'APR ;

— élaboration des arbres de défaillances en posant une par une les différentes actions causant des dommages afin de déterminer les causes pouvant les déclencher.

La construction de l'arbre de défaillance se fait par le calcul des probabilités des événements, celle de l'évènement au sommet étant calculée par la propagation des probabilités de répétition des évènements de base. Le calcul des plus petites combinaisons d'évènements, appelées coupes minimales (chemin critique), se fait de la même manière que le calcul de la probabilité de l'évènement au sommet, mais en n'utilisant que les plus petits ensembles d'évènements de base risquant de conduire à une situation indésirable et redoutable. Ce processus est poursuivi jusqu'à l'enchainement de toutes les combinaisons des évènements de base conduisant à un accident.

Il est important de préciser que ce sont des jugements d'experts, des retours d'expériences (REX), des audits ainsi que des tests qui fournissent les calculs des probabilités des événements de base, et qui vont par même aider à améliorer la sécurité et la fiabilité d'un système en constituant un arbre de défaillance bien détaillé.

1.11 Gestion des ressources de production

La littérature est riche de travaux portant sur la gestion de ressources de production. On y trouve des travaux liés à la gestion des ressources, l'optimisation des affectations des opérateurs ainsi que la gestion optimale des activités de la maintenance et/ou de production. L'objectif de ce dernier axe est le développement de nouvelles politiques optimales de maintenance et/ou de production sous une contrainte technico-économique, en tenant compte de diverses contraintes opérationnelles liées à ces deux enjeux. Parmi tous ces aspects, on a sélectionné quelques axes qui sont présentés ci-dessous à travers une étude bibliographique.

Cadence de production

On dénombre plusieurs travaux de recherche liés à l'optimisation des cadences de production selon la demande à satisfaire en tenant compte de la dégradation du système de production, ainsi qu'un ajustement au réglage des cadences de production selon des contraintes technico-économiques. Parmi ces travaux, on citera les travaux de Ayed [6] (2011) et Dellagi [7] (2017).

Les travaux de Ayed et al [6] tournent autour de la maintenance intégrée à la production avec la contrainte de sous-traitance. Les auteurs proposent un plan optimal de maintenance et production qui garantit une minimisation simultanée des coûts de production, de maintenance de l'inventaire et des demandes perdues. Le problème proposé est celui d'une machine qui fait appel à un sous-traitant pour compléter le reste de la demande sous une contrainte de satisfactions de client. Un plan de maintenance préventive est mis en place pour la machine principale tenant compte de la dégradation cette dernière qui évolue selon le temps et l'usage.

Par conséquent, la sous-traitance joue simultanément deux rôles consistant à compléter le manque de demande afin de réduire les demandes perdues ainsi que les coûts que la réduction de la cadence de production de machine principale qui entraine une réduction de sa dégradation et par conséquent une réduction de ses coûts de maintenance. Par le biais d'une modélisation analytique, les auteurs proposent une fonction objectif intégrant les coûts de maintenance pour la machine principale et les coûts de production, d'inventaire et de demandes perdues par rapport aux systèmes de production composé des deux machines. L'optimisation du modèle analytique développé a donné lieu simultanément à un plan de maintenance préventive pour la machine principale et surtout un plan de production pour chacune des machines. On notera que la décision économique de la cadence de production de la machine sous-traitante permet de réduire d'une part les coûts de demandes perdues et d'autre part les coûts de maintenance de la machine principale puisqu'on réduit sa cadence de production et par conséquent sa dégradation. Les auteurs présentent dans cet article un exemple numérique et une étude de sensibilité pour valider le modèle analytique proposé.

Toujours dans le cadre des stratégies de maintenance intégrée à la production Dellagi et al [7] ont récemment proposé dans leur article, un plan optimal de maintenance intégrée à la production, en tenant compte de l'impact de la fluctuation des cadences de production entre deux périodes successives, sur un horizon de temps fini. En effet, la pénalisation de l'écart de cadence production entre deux périodes successives est prise en considération dans la fonction objectif qui intègre à la base les coûts liées à la production, la maintenance, le stockage et les demandes perdues. L'optimisation de la fonction objectif leur permet d'obtenir simultanément un plan de maintenance préventive et un plan de production minimisant la fonction objectif, tout en réduisant au maximum l'écart de cadence de production entre deux sous-périodes successives. Ceci constitue l'originalité de cette étude. Le plan optimal de maintenance intégrée à la production correspond formellement aux cadences de production pour chaque sous-période afin répondre à une demande aléatoire prédéfinie, et au nombre d'actions de MP à réaliser sur un horizon de temps fini.

Il est clair que la gestion des cadences de production joue un primordial sur l'aspect économique de la gestion des ressources de production.

Multi-produits

Les études de gestion de cadence de production n'ont pas été traitées uniquement dans

le cas des systèmes de production mono-produits. On assiste en effet à l'émergence d'études traitant du cas de multi-produits. On peut citer les travaux de L. Mifdal, Z. Hajej, S. Dellagi [8] (2015).

Dans cet article, les auteurs présentent une extension des travaux effectués dans le cadre de la maintenance intégrée à la production d'un produit unique. Ils traitent en effet ici du développement de nouvelles politiques de maintenance intégrée à la production dans le cas d'un système de production muti-produits. Une planification optimale des tâches de maintenances et de réglage de cadence de production est établie pour chaque produit tout en tenant compte de diverses contraintes opérationnelles. On notera qu'un modèle analytique est proposé dans cette étude, afin de développer une fonction objectif coût intégrant la maintenance, la production, l'inventaire et les demandes perdues pour plusieurs produits, tout en tenant compte d'une contrainte de satisfaction de client exigée pour chaque produit. Cette fonction est optimisée, grâce à une simulation numérique, afin d'établir un plan de maintenance et une cadence de production optimale pour chaque sous-produit.

Gestion de stocks

Dans un même objectif de gestion de stock, des travaux récents de Samir Haddad [9] (2012) sur les lignes de production, s'intéressent à un ligne de M machines et de $M-1$ stocks tampons, afin de traiter une séquence de N produits distincts, par lots de tailles connues. Pour ce faire, il définit un temps de mise en route pour chaque produit, ainsi qu'un temps de traitement pour chaque machine.

Étant admis dans son mémoire, qu'ils ont une taille connue, les $M-1$ stocks tampons sont utilisés dans le traitement pour isoler les machines, présentes dans la ligne de production, ayant une cadence de production qui change d'un produit à l'autre. Ce traitement est fait après utilisation d'un modèle de programmation pour définir la durée de traitement de chacune des séquence des N produits. Il présente un programme permettant également de renvoyer les détails des produits de chaque machine se trouvant en blocage, la maintenance des machines non fiables et la famine.

1.12 Positionnement de la problèmatique industrielle de la thèse : prise en compte simultanée des contraintes de sécurité, cadence de production et de gestion des ressources

De l'étude bibliographique, on peut noter que la majorité des chercheurs s'intéresse à la gestion simultanée de divers aspects industriels. Beaucoup de travaux s'intéressent notamment à la maintenance intégrée. Si ces travaux arrivent généralement à poser de façon analytique le

problème, la résolution s'en avère généralement compliquée.

On peut le plus généralement trouver des résolutions numériques pour les problèmes à faible niveau de complexité et des méthodes heuristiques pour les problèmes plus complexes. Mais, dans les travaux analysés, rares sont ceux qui évoquent précisément les méthodes de résolution mis en œuvre et/ou qui s'intéressent aux limites de leur méthodes de résolutions en cas d'accroissement de la taille et complexité des modèles qu'ils proposent. On notera que les problèmes abordés sont traités par des méthodes « optimisées > au sens méthodologique et non au sens résolution. La méthode de résolution peu être optimisée envers l'un des aspects, mais pas forcément pour un traitement conjoint des différents contraintes du problème, en type et nombre.

Une des originalités des travaux présentés dans cette thèse est de mettre en avant le traitement simultané des contraintes de sécurité, de cadence de production et de gestion des ressources d'un problème industriel, en recherchant l'obtention rapide d'une solution, malgré la complexité du problème.

Dans notre étude, on s'intéresse aux contraintes simultanées de gestion de ressources de production, de respect des cadences et de respect de la sécurité. On notera que la contrainte de sécurité est rarement prise en compte au même niveau que la gestion des ressources de production. C'est dans cette optique que l'on a cherché à formaliser le problème de production avec les différentes contraintes de natures différentes, en vue de les traiter simultanément, et de pouvoir tester ainsi également l'existence (faisabilité) d'une solution.

1.13 Modélisation par ILP (*Integer Linear Programming*)

Dans cette section, nous présentons une méthode de modélisation dite de programmation linéaire en nombre entiers, qui sera utilisée dans la suite du chapitre pour modéliser des problèmes industriels réels.

1.13.1 Principes et formulation

La programmation linéaire en nombres entiers [10, 11], ILP en abrégé, est le nom donné à un formalisme permettant de représenter des problèmes d'optimisation linéaire discrets, et d'algorithmes de résolution correspondant. Il s'agit par définition d'un problème linéaire, dit « programme linéaire » (LP – *Linear Programming*), dont les variables $X = (x_1, \cdots, X_n)$ ne peuvent prendre que des valeurs entières [12], c'est-à-dire, $x_i \in \mathbb{Z}$ au lieu de $x_i \in \mathbb{R}$. Il s'agit d'un problème NP-difficile (*NP-hard*). En effet, autant la résolution d'un problème LP est simple et rapide, grâce à des algorithmes tels que le SIMPLEX [], il en est tout autrement du problème ILP du fait de la contrainte d'intégralité concernant les variables x_i. Réciproquement, tout problème NP-complet peut en principe être formulé comme un ILP.

Sous forme canonique, un problème de programmation linéaire en nombre entier peut être exprimée de la manière suivante :

$$\max({}^{T}C \cdot X) \Rightarrow A \cdot X \leq B, X \in \mathbb{Z}^n \tag{1.4}$$

où A est une matrice, B et C des vecteurs et X le vecteur de n variables entières. Il est à noter qu'une inégalité peut toujours être réécrite sous forme d'inégalité d'infériorité en multipliant par (-1) chacun des côtés de l'inégalité. Une autre formulation existe (équation (1.5)), appelée forme standard [12], qui élimine les inégalités en ajoutant des variables d'écart « s » auxiliaires :

$$\max({}^{T}C \cdot X) \Rightarrow A \cdot X + S = B, X \in \mathbb{Z}^n \tag{1.5}$$

Il existe un cas particulier notable de ILP, qui est l'un des 21 problèmes complets de Karp [13], où les inconnues sont binaires, c'est-à-dire, $x_i \in \{0, 1\}$. La programmation linéaire entière, et en particulier celle en $\{0, 1\}$, est largement utilisée pour formuler de nombreux problèmes d'optimisation combinatoire et de recherche opérationnelle. La formulation arithmétique de ILP rend souvent la tâche intellectuellement plus simple que la formulation booléenne de SAT (que l'on verra dans les chapitres suivants).

1.13.2 Exemple de problème formulé en ILP $\{0, 1\}$

Cette sous-section présente un exemple de formulation ILP en $\{0, 1\}$ appliqué au problème de coloration d'un mapmonde. Il s'agit de déterminer le nombre minimal de couleur différentes qu'il faut utiliser pour colorer les pays sur un mapmonde sans jamais attribuer la même couleur à deux pays adjacents. Considérons l'existence de n pays. Une solution triviale mais non optimale au problème de coloriage du mapmonde est d'attribuer une couleur différente à chaque pays. En effet, n couleur différentes permettent de satisfaire la constrainte d'adjacence.

En utilisant l'ensemble de n^2 variables $P = \{p_{i,j} \ / \ (i, j) \in \{1, \cdots, n\}^2\}$, où la variable $p_(i, j)$ vaut 1 quand la couleur j est attribuée au pays i, ainsi que l'ensemble de n variables $C = \{C_j \ / \ j \in \{1, \cdots, n\}\}$, où c_j à 1 indique que la couleur j est utilisée par au moins un pays, il est possible de formuler le problème simplement de la manière suivante :

$$\min(\sum_i c_i) \quad \rightarrow \quad \text{minimisation du nombre de couleurs utilisées} \tag{1.6}$$

$$\forall i, \ c_j \geq p_{i,j} \quad \rightarrow \quad \text{la couleur } j \text{ est utilisée} \tag{1.7}$$

$$\forall i, \ \sum_j p_{i,j} = 1 \quad \rightarrow \quad \text{une et une seule couleur attribuée par pays} \tag{1.8}$$

$$\forall j, \ p_{i,j} + p_{k,j} \leq 1 \quad \rightarrow \quad \text{si les pays } i \text{ et } j \text{ sont adjacents} \tag{1.9}$$

Si k couleurs, avec $1 < k < n$, sont nécessaires au minimum pour satisfaire le problème, alors la solution n'est pas unique : il existe en fait $C_k^n = n!/((n-k)! \cdot k!)$ solutions différentes à k couleurs.

1.14 Modélisation de problèmes industriels

Cette section s'intéresse aux problèmes industriels de production et de maintenance préventive où une étude est faite pour extraire les contraintes et risques pouvant se produire dans une usine et provoquer des dommages dangereux. Ces contraintes sont ensuite modélisées en utilisant une formulation ILP.

1.14.1 Production

Le problème de production est un problème typique d'optimisation combinatoire [14]. Il fait partie du processus de la chaîne d'approvisionnement. En production, plusieurs étapes doivent être respectées, créant un processus spécifique pour convertir les matières premières, les pièces ou les composants en produits finis, afin de répondre aux attentes et exigences d'un client. En outre, des facteurs techniques doivent également être pris en compte pour modéliser le problème, comme le montre la figure 1.5. Ils se matérialisent notamment dans le nombre de machines utilisées ou la manipulation des matières premières. S'y ajoute le facteur humain, la fabrication employant habituellement une relation Homme-Machine avec, dans une production à grande échelle, une répartition des tâches en fonction du personnel (opérateurs, ingénieurs, personnel technique).

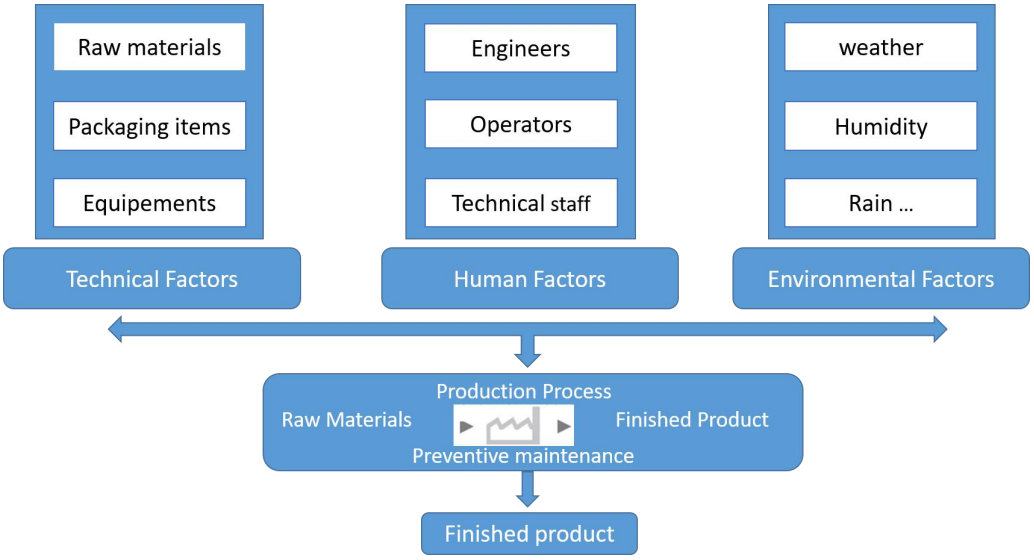

FIGURE 1.5 – Facteurs impliqués dans le processus de fabrication et maintenance préventive.

1.14.2 Formulation ILP du problème de production sous contrainte de sécurité

Cette sous-section présente une approche de formulation ILP pour le problème de production. Les contraintes qui peuvent considérées dans une telle formulation concernent le nombre

et type de machines et/ou de robots ainsi que de personnels (employés, opérateurs) employés pour la production, le nombre d'heures de travail cumulées/consécutives pour chaque personnel/machine, le respect des étapes de sécurité, la gestion des conflits d'utilisation/activation de machines et/ou personnels, les disponibilités/indisponibilités de matière première, etc.

L'exemple de formulation suivant montre, à échelle réduite, la méthodologie que nous employons dans notre approche pour formaliser un problème de production [15, 16].

Considérons l'intervalle de temps ΔT sur lequel la modélisation porte. Il est constitué d'un l'ensemble (ordonné) de sous-intervalles ΔT_i avec $i \in \{1, \cdots, \tau\}$. Si tous les intervalles sont de durée égale (discrétisation par échantillonnage, par exemple), nous pouvons simplifier la notation en ne conservant que le numéro d'indice de l'intervalle. Par abus de notation et terminologie, nous noterons l'indice t et l'intervalle de temps ΔT_t, instant t.

Considérons un ensemble de μ de machines utilisées pendant la production. Pour modéliser le fonctionnement des machines, nous utilisons l'ensemble de variable suivant :

$$\{m_{i,t} \ / \ (i,t) \in \{1, \cdots, \mu\} \times t \in \{1, \cdots, \tau\}\}$$

où $m_{i,j}$ prenant ses valeurs dans $\{0, 1\}$ indique l'activité ou non à l'instant t de la machine i.

Supposons qu'un contrainte limite l'utilisation de la machine i plus d'une certaine durée $D_max(i)$. Selon la même approche que pour les intervalles de temps, cette durée sera représentée comme un nombre d'intervalle D_i. Cette contrainte peut alors se traduire de la manière suivante :

$$\sum_{t=t_1}^{t_2} m_{i,t} \leq D_i, \ \forall (t_1, t_2) \in \{1, \cdots, \tau\}^2 \text{ avec } t_2 - t_1 = D_i + 1 \tag{1.10}$$

De la même manière, considérons π personnels, dont l'activité à l'instant t sera matérialisée par la variable $p_{i,t}$. Supposons que l'on veuille indiquer qu'une machine ne peut fonctionner sans qu'un personnel parmi i, j et k ne soit présent (actifs). Cette contrainte peut s'écrire de la manière suivante :

$$m_{i,t} \leq p_{i,t} + p_{j,t} + p_{k,t}, \forall t \in \{1, \cdots, \tau\} \tag{1.11}$$

La présence simultanée obligatoire de ces mêmes personnels se traduira par :

$$m_{i,t} \leq p_{i,t}, m_{i,t} \leq p_{j,t}, m_{i,t} \leq p_{k,t}, \forall t \in \{1, \cdots, \tau\} \tag{1.12}$$

Si l'on veut ajouter que le nombre de personnels présents (sans différencier le type de personnel) doit être au moins égal au nombre de machines actives, cela s'écrira :

$$\sum_i m_{i,t} \leq \sum_j p_{i,t}, \ \forall t \in \{1, \cdots, \tau\} \tag{1.13}$$

Pour différencier des catégories de personnels, on peut créer des séries de variables différentes $p_{i,t}^{'}, \ p_{i,t}^{"}, \cdots$ pour chacune des catégories plutôt que de identifier « manuellement » les variables $p_{i,t}$.

Une contrainte d'exclusion de présence simultanée de deux personnels identifiées i et j s'écrira :

$$p_{i,t} + p_{j,t} \leq 1,\ \forall t \in \{1, \cdots, \tau\} \tag{1.14}$$

Une contrainte limitant à N la présence simultanée de personnels (indifférenciés) s'écrira :

$$\sum_i p_{i,t} \leq N,\ \forall t \in \{1, \cdots, \tau\} \tag{1.15}$$

Il est facile d'extrapoler ces types de contraintes à l'activité de machines. Il est de même assez facile d'étendre le problème en ajoutant d'autres types « d'entités ». Il est possible par exemple de modéliser l'utilisation d'outils avec des contraintes imposant pour un activité donnée, qu'un personnel soit requis pour le bon fonctionnement d'une machine tout en imposant que ce personnel dispose d'un outil donné.

Modéliser des activités de maintenance est possible selon la même stratégie. Des types de personnels peuvent être dédiés à des types de maintenance. Une activité « unitaire » de maintenance étant modélisé par un pool de variables. Alors, des contraintes telles que « machine inactive pendant maintenance », « nombre de maintenances simultanées limité » sont généralisables selon l'approche décrite.

1.14.3 Limites

L'approche décrite ci-dessus est une approche ILP en $\{0, 1\}$. D'autres formulations ILP, avec d'autres jeux de variables, non restreints à ces deux seules valeurs, sont également envisageables, présentant chacune leurs propres avantages et inconvénients.

Comme cela a été indiqué, ILP est un problème NP-complet dont la résolution s'avère (comme pour les autres problèmes NP-complet) complexe et rapidement prohibitive du point temps de résolution avec la croissance de la taille du problème.

La stratégie que nous nous sommes fixés pour résoudre de manière efficace ces problèmes est de développer une architecture matérielle dédiée de solveur en vue de bénéficier d'un important potentiel d'accélération. De sa forme arithmétique, ILP est à ce jour presque exclusivement traité par des approches logicielles. C'est la raison pour laquelle, nous nous sommes intéressés à un autre formalisme, le formalisme SAT, de nature booléenne, plus adapté à l'objectif de développement d'un solveur matériel dédié. C'est cet objectif qui sous-tend notre choix de modélisation ILP en $\{0, 1\}$ tels que nous l'avons décrit ci-dessus. En effet, il est relativement aisé de transcoder un problème de sa formulation ILP en $\{0, 1\}$ vers une formulation SAT. Nous décrirons au chapitre 2 les règles que nous appliquons pour effectuer cette transformation.

L'utilisation de l'ILP comme étape intermédiaire avant un passage en SAT a été préféré au un codage direct en SAT à cause de la plus grande difficulté intellectuelle à raisonner sur le problème de production directement dans le formalisme SAT.

1.15 Conclusion

Dans ce chapitre, nous avons commencé par présenter les bases de la sécurité, ainsi que les différentes notions du danger et du risque, en s'aidant des différentes normalisations existantes. Nous avons également présenté quelques travaux bibliographiques sur la gestion de ressources de production, de gestion de cadence ainsi que le cas des multi-produits. Nous avons ensuite détaillé différentes méthodes complémentaires d'analyse du risque (APR, AMDE, AMDEC et arbre de défaillance), l'APR permettant d'identifier rapidement les gros problèmes potentiels avant de confier le traitement aux autres méthodes d'analyse. Celles-ci suivent un nombre d'étapes pour définir l'impact des modes de défaillances et de les hiérarchiser.

Après cet état de l'art, nous avons abordé la modélisation de problèmes industriels. Une formulation de type ILP (programmation linéaire en nombres entiers) restreinte aux valeurs $\{0, 1\}$ a été proposée avec une description de la méthodologie employée pour encoder un problème pratique. Ce formalisme a été choisi comme étape intermédiaire vers un formalisme SAT. Ce dernier, décrit dans le chapitre suivant, est un formalisme puissant mais intellectuellement plus difficile utilisé pour encoder directement les problèmes industriels qui nous intéressent.

L'idée principale de ces travaux de recherche touchent donc généralement deux parties : la première concerne la formalisation des problèmes NP complet venant du domaine industriel qui sont présenté au premier chapitre, et la deuxième, la plus importante qui est l'étape majeur et la grande partie de ces travaux : est celle de la résolution de ces problèmes NP complets à travers un solveur SAT puissant et purement parallèle.

L'objectif des chapitres suivants est de proposer une nouvelle architecture matérielle de résolution dédiée, ou solveur, reposant sur le formalisme SAT. Le chapitre 2 présente un état de l'art sur le formalisme SAT et les méthodes et heuristiques de résolution, logicielles pour l'essentiel. Dans le chapitre 2, nous présentons les travaux relatifs au développement de notre solveur SAT dédié.

Chapitre 2

État de l'art SAT et de méthodes de résolution correspondantes

Sommaire

2.1 Introduction

Comme nous l'avons vu dans le chapitre précédent, le monde industriel a vu se développer de nouvelles règlementations, normes et activités, qui l'ont rendu plus vaste encore et incertain. Au chapitre précédent, nous avons introduit une méthode de formulation du problèmes de production de type ILP a solutions en $\{0, 1\}$. Cette modélisation n'est qu'une étape intermédiaire vers la modélisation SAT qui constitue notre véritable cible. En effet, notre objectif est de pouvoir accélérer la résolution de problèmes de production complexes à l'aide de solveurs matériels dédiés que nous nous proposons de développer. À cette fin, nous avons fait le choix

du formalisme SAT que nous présentons dans ce chapitre.

Le problème de satisfiabilité appelé communément problème SAT (pour SATisfiability) est un problème de décision d'une grande importance dans la théorie de la complexité. Le problème SAT englobe un formalisme simple mais très expressif, permettant de représenter un très grand nombre de problèmes de décision difficiles tant académiques que réels. La cryptographie, la recherche opérationnelle, l'intelligence artificielle, l'industrie, la vérification de matériels et de logiciels, sont autant de disciplines connues pour la complexité de leurs études qui, à un moment où un autre, se sont appuyés sur SAT pour leur analyse et traitement. L'optimisation d'un emploi du temps, la coloration de graphe, le voyageur de commerce, l'affectation/répartition de fréquences constituent des exemples classiques de problèmes modélisés et traités par SAT. Le formalisme SAT a également connu un fort succès dans la modélisation et la résolution de nombreux problèmes industriels.

La première partie de ce chapitre a pour objectif de présenter de manière détaillée le problème SAT ainsi que quelques unes de ses variantes. Nous commençons par introduire les notions de base de la logique booléenne dont il est issu. Nous poursuivons par le rappel de résultats théoriques de complexité, bien connus dans la littérature. Cette première partie se termine par la présentation de quelques extensions de SAT telles que les problèmes k-SAT et Max-SAT ainsi que de quelques problèmes académiques classiques.

La deuxième partie du chapitre s'intéresse à la résolution des problèmes de type SAT, et aux diverses travaux qui y ont été consacrés. À l'origine, il s'agissait de trouver une méthodologie permettant de traiter ce type de problème. Dans la mesure où un problème SAT est constitué de variables booléennes ne pouvant prendre deux valeurs différentes, les premiers chercheurs ont rapidement opté pour méthodes de recherche arborescente, qui se répartissent en deux catégories principales, selon que la recherche est effectuée en profondeur d'abord (DFS, *Depth First Search*) ou en largeur d'abord (BFS, *Breadth Firts Search*). Différents algorithmes ont été progressivement développés, cherchant à a trouver une ou toutes les solutions, selon des stratégies complètes ou imcomplètes qui garantissent respectivement de trouver ou non une solution quand elle existe, les stratégies complètes étant cependant malheureusement beaucoup plus coûteuses en termes de temps de résolution.

Nous décrirons l'algorithme DPLL, inventé en 1962 par Davis, Putnam, Logeman et Loveland. un des tous premiers algorithmes efficaces pour résoudre SAT, employant la recherche en profondeur et basé sur une stratégie complète. Nous présenterons également QUINE et CDCL, autre type d'algorithmes, employant des méthodes de recherche incomplétes.

Nous terminerons ce chapitre par la description d'heuristiques employées dans les solveurs logiciels pour augmenter leur rapidité par la réduiction de l'espace de recherche.

2.2 Notions préliminaires

2.2.1 Définitions de base

Definition 1 *Une proposition est une affirmation qui peut être vrai ou fausse.*

Exemple : l'affirmation « la marée est haute » peut-être vrai ou fausse en fonction de l'endroit et l'instant où elle est énoncée.

Definition 2 *Une variable booléenne ou variable propositionnelle peut prendre la valeur 1 (vraie) ou 0 (fausse).*

Definition 3 *L'affectation est un opérateur qui affecte à une variable booléenne une valeur de l'ensemble {1, 0} ou, respectivement, de l'ensemble {vrai, faux}. Cet opérateur est également appelé assignement.*

Definition 4 *Un littéral est l'expression d'une variable sous sa forme positive ou négative. À toute variable v, correspondent deux littéraux, v et $\neg v$. Si v vaut 0, alors $\neg v$ vaut 1 et inversement.*

2.2.2 Définitions de base

L'algèbre de Boole est dû au célèbre mathématicien George Boole. Elle est définie sur trois opérateurs majeurs, à savoir : la négation, le ET logique et le OU logique, dont les notations formelles sont respectivement $\neg$, $\wedge$ et $\vee$. L'opérateur $\neg$ est un opérateur unaire qui inverse la valeur de vérité associée à une variable binaire. Les opérateurs $\wedge$ et $\vee$ sont des opérateurs binaires : $a \wedge b$ n'est vrai et $a \vee b$ n'est faux que quand les deux variables propositionnelles a et b sont respectivement vraies et fausses. Les tables de vérité associée à ces trois opérateurs sont regroupées dans le tableau suivant :

TABLE 2.1 – Tables de vérités booléenne pour $\neg$, $\wedge$ et $\vee$.

a	b	$\neg a$	$a \wedge b$	$a \vee b$
vrai	vrai	faux	vrai	vrai
vrai	faux	faux	faux	vrai
faux	vrai	vrai	faux	vrai
faux	faux	vrai	faux	faux

Definition 5 *Une clause est une disjonction de littéraux. Une clause est satisfaite si et seulement si au moins un des littéraux est vrai.*

Exemple : la clause $a \vee b$ est vraie si l'une ou l'autres des variables a et b est vraie, fausse sinon.

Definition 6 *Une formule sous forme normale conjonctive ou CNF (*Conjonctive Normal Form*) est une conjonction de clauses. Une telle formule est vraie si et seulement si toutes les clauses sont satisfaites.*

Exemple : la formule propositionnelle suivante est sous forme normale conjonctive

$$F = (a \vee b) \wedge (a \vee \neg c) \wedge (c \vee d)$$

Definition 7 *Soit F une forme normale conjonctive définie sur l'ensemble des variables V. Une interprétation est une affectation totale de toutes les variables de V. Une affection d'un sous-ensemble seul de V est une interprétation partielle. Si le cardinal de V est n, alors le nombre d'intermprétations de F est égal à 2^n.*

Definition 8 *Soit F une formule en forme normale conjonctive définie sur l'ensemble des variables V. Une solution de F est une interprétation complète de V qui satisfait toutes les clauses de F.*

Definition 9 *Soit F une formule en forme normale conjonctive définie sur l'ensemble des variables V. F est satisfiable si et seulement si elle admet une ou plusieurs solutions. Elle est insatisfiable si elle n'admet aucune solution.*

Exemple : la formule $F = (\neg a \vee b) \wedge (c \vee \neg d)$ est satisfiable car elle admet au moins la solution $a \leftarrow 0$, $c \leftarrow 1$ quelles que soient les valeurs de b et d.

Exemple : la formule $F = (\neg a \vee b) \wedge (c \vee \neg d) \wedge a \wedge \neg a$ est trivialement insatisfiable car elle contient deux clauses unitaires contradictoires

Definition 10 *Soit F une CNF définie sur l'ensemble des variables V et qui composée de l'ensemble des clauses C. F peut être simplifiée si une variable v de V est assignée à 1 (respectivement 0). Dans ce cas les clauses de C peuvent être séparées selon l'interprétation du littéral de la manière suivante :*

- *C_{v+}, les clauses de C contenant le littéral v sous forme positive ;*
- *C_{v-}, les clauses de C contenant le littéral v sous forme négative ;*
- *$C_{v\notin}$, les clauses de C ne contenant pasle littéral v.*

Exemple : soit F la CNF définie sur les clauses suivantes : $c_1 = a \vee c$ et $c_2 = \neg a \vee d$. Si on affecte à a la valeur 1 ; alors l'ensemble des clauses se résume à $c_2 = d$.

Definition 11 *Une clause est une tautologie si et seulement si elle est vraie quelles que soient les valeurs de vérités assignées à ses variables. Par exemple, $a \wedge \neg a$ est une tautologie.*

Definition 12 *Une clause est unitaire si et seulement si elle ne contient qu'un seul littéral. On peut simplifier une formule F en commençant par satisfaire les clauses unitaires.*

2.2.3 Problème SAT

Definition 13 *Étant donnée une formule F en forme normale conjonctive sur V et l'ensemble des clauses C, le problème SAT est un problème de décision qui consiste à trouver une interprétation complète des variables de V qui satisfasse l'ensemble des clauses C.*

2.3 Notions de base sur la théorie de la complexité

Definition 14 *Classe P [17, 18] – un problème est dans la classe P (*Polynomial time*) si et seulement s'il peut être résolu par un algorithme déterministe en un temps polynômial.*

Definition 15 *Classe NP [17, 18] – un problème est dans la classe NP (*Non Deterministic Polynomial time*) si et seulement si n'importe quelle solution du problème peut être vérifiée en un temps polynômial.*

Remarque : tout problème pouvant être résolu par un algorithme polynômial peut aussi être résolu par un algorithme polynômial non déterministe. D'où la conjecture $P \subseteq NP$. La classe NP est composée de deux parties, à savoir, les problèmes NP-complets et les problèmes NP-difficiles, comme cela est montré dans la figure 2.1.

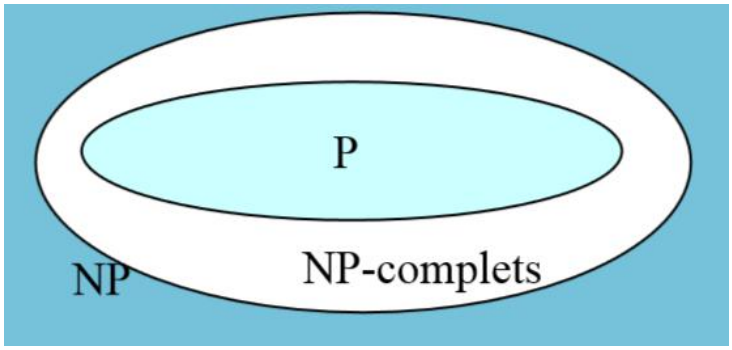

FIGURE 2.1 – Hiérarchie des complexités.

Definition 16 *Un problème de décision Π est NP-complet s'il appartient à la classe NP et s'il admet une réduction polynômiale vers un autre problème NP-complet.*

Definition 17 *Un problème d'optimisation Π est NP-difficile s'il appartient à la classe NP et s'il admet une réduction polynômiale vers un autre problème NP-complet.*

SAT est le premier problème de décision pour lequel il a été prouvé, par Cook en 1970 [17], qu'il était NP-complet. La majorité des problèmes d'optimisation sont NP-difficiles.

2.4 Exemples de problèmes SAT

2.4.1 Problème k-SAT

Definition 18 *Un problème k-SAT, est un problème SAT où l'arité des clauses est égale à k. L'arité des clauses est le nombre maximum de littéraux contenus dans une clause. Le problème 3-SAT est un problèmes SAT où toutes les clauses sont d'arité 3.*

Le problème k-SAT a été prouvé NP-comple pour $k \geq 3$. Le problème 2-SAT est polynômial.

2.4.2 Problèmes max-SAT et max-weighted-SAT

Definition 19 *Un problème max-SAT [19] est un problème d'optimisation consistant à trouver une instanciation de toutes les variables qui maximise l'ensemble de clauses satisfaites.*

Il existe une version max-weighted-SAT pondérée de max-SAT où toutes les clauses sont pondérées par des poids. Le problème consiste alors à trouver une solution qui maximise le poids total des clauses satisfaites.

Les problèmes max-SAT et Max-Weighted-SAT sont NP-difficiles.

2.4.3 Problèmes des pigeons et autres

Le problème des pigeons [20] (voir figure 2.2 est l'un des problèmes NP-complets les plus connus. Étant donnés n pigeons et $n-1$ pigeonniers, le problème consiste à répartir les pigeons dans les pigeonniers avec au plus un pigeon par pigeonnier.

FIGURE 2.2 – Problème des pigeons.

Le problème de coloriage de graphe [21] consiste à colorier les nœuds d'un graphe, chacun ne pouvant se voir attribuer qu'une seule couleur, et une même couleur ne pouvant être attribuée à deux nœuds adjacents, c'est-à-dire, reliés par un arc. Le problème de coloriage de mapmonde (tel que présenté dans le chapitre précédent) peux être assimilé au problème de coloriage de graphe. Il s'agit de colorier les pays (resp. régions) d'une carte, en n'attribuant qu'une seule

couleur par pays (resp. région) et en ne coloriant deux pays (resp. régions) adjacents qu'avec des couleurs différentes.

FIGURE 2.3 – Exemple de problème de coloration de graphe.

2.5 Résolution de SAT

De nombreux procédés ont été proposés pour résoudre le problème SAT, nommés algorithmes. Comme pour tout problème d'optimisation, il existe un grand nombre de méthodes et algorithmes pour résoudre SAT [22, 23], pouvant être classés en deux catégories [23] : les algorithmes de recherche complets ou algorithmes énumératifs (méthodes exactes ou complètes) et les algorithmes de recherche incomplets (méthodes incomplètes) basés sur le principe de « générer puis tester » ou sur la recherche locale.

Les méthodes exactes garantissent l'obtention d'une solution, si elle existe. Ce sont des techniques de recherche énumératives qui explorent un espace complet de solutions. Elles garantissent l'optimalité d'un problème d'optimisation et l'obtention de la solution d'un problème de décision. Elles sont par contre très coûteuses en temps. Plusieurs algorithmes exacts peuvent être trouvés dans la littérature, le plus répandu étant DPLL [24].

2.5.1 Résolution arborescente binaire

Cette section traite des méthodes de recherche arborescentes dans le but de résoudre des problèmes combinatoires de décision, d'ordonnancement ou d'optimisation. Ces problèmes sont généralement NP-difficiles [25] ce qui a poussé les chercheurs à développer de nouvelles méthodes de simplification du parcours de recherche [26]. La plupart sont basées sur le concept de graphe orienté et de parcours de graphe.

Un arbre binaire est constitué de nœuds, de chaque nœud partant un arc « gauche » et un arc « droite ». Chaque nœud n'est la destination que d'un seul arc, à l'exception du nœud racine de l'arbre qui n'est la destination d'aucun arc. Dans une représentation visuelle telle

que celle de la figure 2.4, le nœud racine figure de manière conventionnelle au sommet de l'arbre (représentation paradoxalement inversée d'un arbre). Chaque nœud est au sommet d'un sous-arbre, les arcs gauche et droit sortant de ce nœud pointant respectivement vers des sous-sous-arbres gauche et droit plus petits. Chacun de ces derniers peut être vide, auquel cas il ne contient aucun nœud. Dans le cas contraire, les arcs pointent sur les sommets respectifs de ces sous-arbres. Un nœud terminal ou feuille est un nœud dont les deux sous-arbres gauche et droit sont vides. Par souci de simplification, il est courant d'omettre les arcs pointant vers des sous-arbres vides.

Une définition récursive formelle d'un arbre binaire est la suivante : un arbre binaire est soit vide, soit constitué d'un nœud d'où partent des arcs gauche et droite pointant chacun vers un arbre binaire.

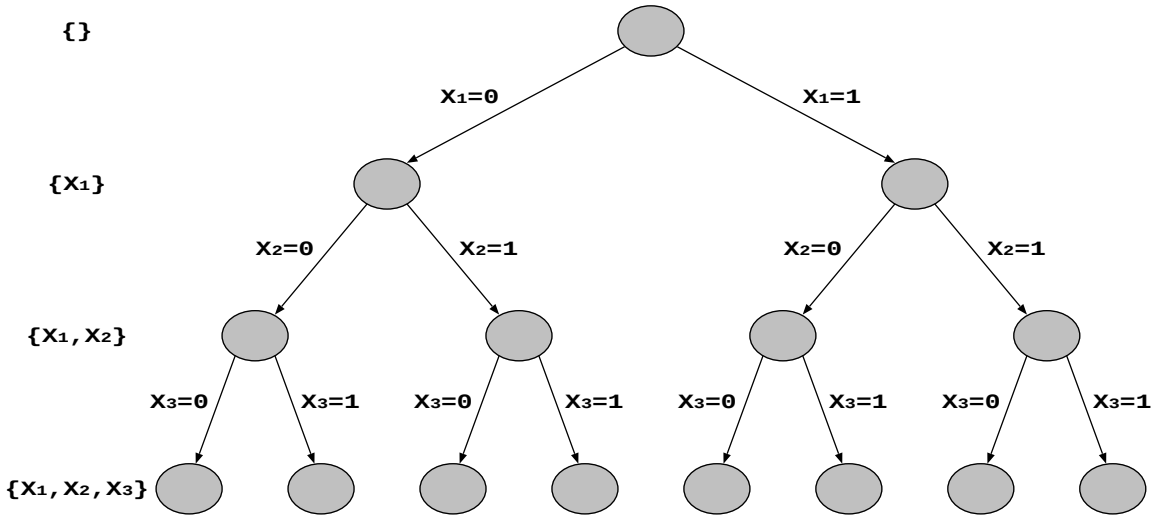

FIGURE 2.4 – Espace de recherche de problème SAT représenté sous forme d'arbre binaire.

L'espace de recherche d'un problème de SAT peut être structuré sous forme d'arbre binaire, ouvrant ainsi la voie à des stratégies, largement utilisées dans les problèmes NP-complets, de recherche par parcours d'arbre. La recherche se termine dès lors qu'une solution est trouvée (recherche de faisabilité) ou bien, lorsque plus aucune solution ne peut être trouvée lors d'une recherche exhaustive de solutions ou bien en l'absence de solutions.

Un problème SAT donné peut être représenté par une variété de structures arborescentes, la représentation n'étant pas unique. La figure 2.4 illustre une telle possibilité, où un nœud représente une instanciation plus ou moins complète des variables, le nœud racine (profondeur 0 de l'arborescence) correspondant à la situation où aucune variable n'est affectée. Un arc liant un nœud de la profondeur p et à un de la profondeur $p + 1$ représente une affectation de la variable X_{p+1}. Par conséquent, un nœud à la profondeur p a toutes les variables de $\{x_1, x_2, \cdots, X_p\}$ assignées et celles de $\{x_{p+1}, \cdots, X_n\}$ non assignées. Compte tenu de l'assignation des variables sur un nœud, la valeur correspondante prise par la formule booléenne peut être évaluée. La valeur peut être SAT si toutes les clauses sont vraies, UNSAT si au moins

une clause est fausse et indécidable dans le cas contraire. Il est important de noter que pour cet arbre, tous les nœuds d'un sous-arbre partagent la même valeur que sa racine lorsque celle ci vaut SAT ou UNSAT. Cette propriété est généralement exploitée par les algorithmes de recherche pour réduire l'espace de recherche.

2.5.1.1 Parcours en profondeur d'abord (DFS) et largeur d'abord (BFS)

Les stratégies de recherche par parcours d'arbre peuvent être classées en deux types principaux, comme indiqué dans la figure 2.5, à savoir la recherche en profondeur d'abord [30,44] (DFS – *Depth First Search*) et la recherche en largeur d'abord [31] (BFS – *Breadth First Search*).

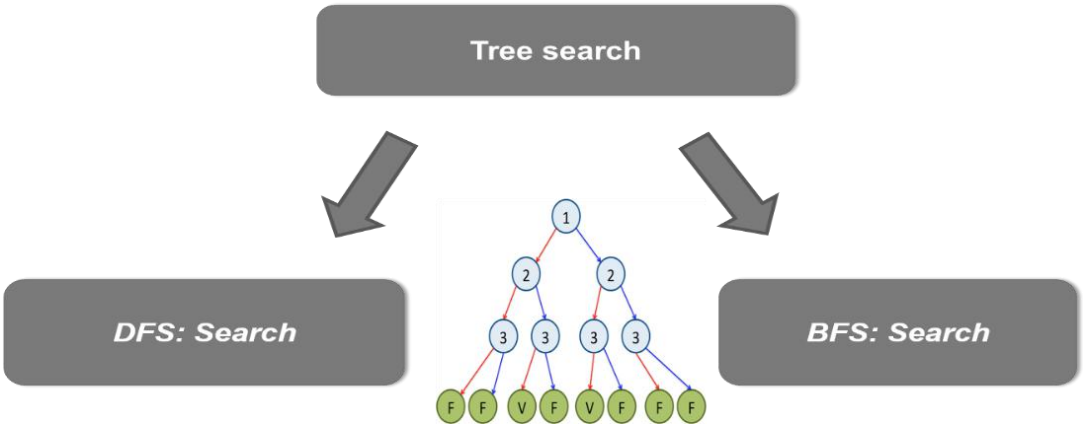

FIGURE 2.5 – Types de recherche arborescente.

DFS est une stratégie de recherche d'une solution parcourant l'arbre nœud par nœud, de la racine vers les feuilles autant que possible. Appliqué à l'arbre de la figure 2.4, chaque fois qu'un nœud est évalué à UNSAT la recherche retourne vers le nœud parent (*backtrack*) afin d'explorer d'autres branches. Si le nœud est évalué à SAT, le parcours peut être stoppé quand seule la faisabilité est recherchée. À l'opposé, en cas de recherche exhaustive des solutions, le parcours se poursuit avec un backtrack similaire au cas UNSAT. Le parcours du sous-arbres dont le nœud courant est la racine est inutile puisque l'on sait qu'il ne contient que des solutions). Lorsque la recherche retourne à la racine sans qu'autre branche ne subsiste à explorer, la recherche se termine. Si aucun nœud SAT n'a été trouvé, le problème n'a pas de solutions.

BFS est une stratégie qui cherche à examiner les nœuds d'un graphe en privilégiant le parcours des nœud d'un même niveau, ou les branches collatérales. En commence par le nœud racine, on cherchera à explorer en premier les tous les nœuds du niveau suivant avant d'approfondir plus en avant l'exploration de l'arbre. Cette stratégie ne s'applique pas bien à l'arbre de la figure 2.4, mais s'avérer bien adaptée pour d'autres types d'arbres de représentation de SAT.

L'approche DFS est bien adaptée à un traitement séquentiel, et donc très adéquate pour une implémentation logicielle. Le traitement selon une approche BFS [27] tend à être très

gourmand en ressources mémoire. Lorsque la largeur est explorée dans son intégralité, on remarquera que le backtrack n'a pas de raison d'être puisque tous les nœuds en amont du nœud en cours d'évaluation ont déjà été évalués. BFS tend plus naturellement que DFS vers la possibilité d'un traitement parallèle du parcours de recherche, moyennant une complexité accrue.

Les deux approches ont en commun de ne jamais requérir plus d'une évaluation par un nœud.

2.5.1.2 Algorithmes de recherche complets et incomplets.

Les algorithmes de recherche complet garantissent la découverte d'une solution s'il en existe [13] (ou de toutes les solutions en cas de recherche exhaustive). Pour un problème SAT à n variables, un arbre tel que celui de la figure 2.4 contient $1 + 2 + \cdots + 2^n$, soit $2^{n+1} - 1$ nœuds. Bien que, compte tenu de propriétés telles que celles décrites pour l'arbre de la figure 2.4 (valeurs des nœuds d'un sous-arbre dont la racine s'évalue à SAT ou UNSAT), tous les nœuds d'un arbre n'aient pas besoin d'être traversés, le nombre de nœuds le devant croît très rapidement avec la taille des problèmes [28]. ce qui fait exploser les durées de résolution. Pour contrecarrer cette tendance, diverses techniques vont être mises en œuvre pour essayer de simplifier le parcours en transformant/simplifiant dynamiquement l'arbre au fur et à mesure du parcours. Bien sûr, seules sont admissibles les transformations ne modifiant pas l'espace des solutions.

Les algorithmes de recherche cherchent à réduire autant que possible les temps de recherche mais en courant le risque de rater de solutions. Contrairement aux algorithmes basés sur une recherche « exhaustive » avec backtrack, les méthodes incomplètes sont généralement basées sur techniques de recherche locale stochastique, en privilégiant des parties de l'arbre où l'on considère plus fortes les chances de trouver une solution au problème SAT. Cependant, les heuristiques employés ne permettent pas de garantir que les solutions, quand il en existe, ou tout du moins une solution, se trouve dans les parties explorées. Les stratégies mises en œuvre dans l'élaboration des heuristiques visent à réduire autant que possible le risque de ne trouver aucune solution alors qu'il en existe. Les algorithmes de recherche incomplets sont généralement employés pour la résolution de problèmes de minimisation, où les approximations induites par les heuristiques permettent de trouver des solutions qui sans être optimales s'en rapprochent (solutions sub-optimales).

2.5.2 Algorithmes de résolution DP/PDLL

Davis et Putnam ont proposé, en 1960, la première méthode exacte pour résoudre les problèmes SAT présentés sous forme CNF. Mais pour rendre cet algorithme utilisable sur la plupart des solveurs SAT complets, des modifications ont été introduites en 1962 par Davis, Putnam, Logemann et Loveland [29]. Les sections suivantes expliquent en détail le bases et principes de ces algorithmes.

2.5.2.1 Algorithme Davis-Putnam

La première méthode exacte pour résoudre les problèmes SAT a donc été proposée par Davis et Putnam en 1960 [29]. Elle est basée sur la simplification de la formule F donnée sous forme CNF [**?**], par la suppression ou l'ajout de clause, selon les besoin. La suppression se fait par la vérification de l'existence de clauses tautologiques là où l'intersection de deux ensembles de clauses n'est pas vide. L'ajout de résolvantes se fait par contre uniquement si les clauses ne présentent pas des tautologies comme le montre l'exemple suivant : soit C_1 et C_2, deux clauses d'une formule F CNF ; une résolvante $(C_1 - \{l\}) \bigcup (C_2 - \{\bar{l}\})$ sera ajoutée contenant les littéraux $l \in C_1$ et $\bar{l} \in C_2$.

Une fois toutes les résolvantes de tous les littéraux l ajoutées, on remarquera la suppression de toutes les clauses contenant les littéraux l, ce qui simplifie l'évaluation de la formule F et facilite la prise de décision quant à sa satisfiabilité de F. D'après Davis et Putnam, le résultat de la formule est donné après toutes les résolutions où elle peut être satisfiable si toute les clauses sont vraies, dans le cas contraire une seule condition la rend insatisfiable qui est l'apparition d'une clause vide ou fausse. Ceci est décrit dans l'algorithme 1 suivant [29].

Algorithme 1 Algorithme DP (Davis-Putnam)

Entrée: Une formule Φ en CNF
Sortie: Satisfiabilité de Φ

```
pour chaque variable l apparaissant dans Φ faire
    Résolvantes ← ∅ ;
    pour chaque couple de clauses (C1, C2) de CΦ faire
        si l ∈ C1 et l̄ ∈ C2 alors
            C ← (C1 − {l}) ⋃ (C2 − {l̄}) ; {création d'un nouvelle clause}
            si C n'est pas une tautologie alors
                Résolvantes ← Résolvantes ⋃ C ; {ajouter à C l'ensemble des Résolvantes en l}
            fin si
        fin si
    fin pour
    Clausesl ← {γ ∈ CΦ | l ∈ γ ou l̄ ∈ γ} ; {ensemble de clauses contenant la variable l}
    Φ ← Φ− Clausesl ;
    Φ ← Φ ⋃ Résolvantes ;
fin pour
si Φ contient une clause vide alors
    retourner « Insatisfaisable » ;
sinon
    retourner « Satisfaisable » ;
fin si
```

Parmi les inconvénients de l'algorithme DP, on trouve le grand nombre de résolvantes ajoutées lors des simplifications, ce qui présente un énorme problème en taille, et par conséquent, en temps de résolution. Il faut noter aussi que la procédure DP, si elle répond à la demande de

déterminer l'existence de solutions pour une formule F donnée, elle n'est malheureusement pas capable d'en identifier explicitement.

Pour éliminer ces problèmes, deux ans plus tard, des modifications ont été apportées par Davis, Putnam, Logemann et Loveland à l'algorithme DP en divisant le problème en deux, par la création d'un arbre binaire [30] et l'application récursive de l'algorithme à chaque sommet. C'est l'algorithme DPLL.

2.5.2.2 Algorithme Davis-Putnam-Logemann-Loveland

L'algorithme DLL (Davis, Putnam, Logemann, Loveland) [24], inventé en 1962, constitue une révolution pour la résolution des problèmes NP-complets. Elle a été utilisée sur la plupart des solveurs SAT complets. C'est un algorithme qui examine toutes les affectations possibles au problème SAT, à savoir 2^n possibilités où n est le nombre de variables booléennes du problème SAT considéré. Comme pour l'algorithme DP, une simplification de la formule F s'impose par les littéraux monotones ou unitaires. Elle se fait également par l'assignation des littéraux $l \in L$ (en général variable $v \in V$) par des valeurs de vérités qui permettant de supprimer les redondants dans les clauses (la simplification de la formule F est faite par le littéral v et notée $F\backslash v$ ou par $\overline{v}$ et notée $F\backslash\overline{v}$). Cette première modification de l'algorithme DP est détaillée dans les algorithmes 2, 3 et 4. L'algorithme 2 constitue l'algorithme DPLL proprement dit. La propagation unitaire, décrite dans l'algorithme 3, simplifie une formule dans laquelle on trouve des clauses contenant un seul littéral qui, devant forcément être évaluée à vrai, imposent la valeur de leurs littéraux et donc variables. Cette simplification par la suppression de toutes les occurrences des littéraux l des clauses unitaires s'effectue jusqu'à épuisement des simplifications possible ce qui rend la résolution des formules plus facile [31]. Cette simplification est détaillée dans l'algorithme 4.

La méthode mise en œuvre dans DPLL est basée sur les algorithmes de recherche complets, ce qui permet de parcourir l'arbre de recherche intégralement, par l'assignation des variables une par une. On précise également que cet algorithme utilise l'approche DFS (recherche en profondeur d'abord) [32], en fixant une variable et continuant à descendre dans l'arbre pour la vérification de la satisfiabilité ou non satisfiabilité. Ce processus s'exécute tout en simplifiant les clauses et, comme il a été expliqué, si au moment de la recherche de la solution, avec les affectations aux variables données, une clause vide (dite fausse également) est détectée, l'algorithme fait un backtrack, c'est-à-dire, un retour en arrière d'un niveau k de l'arbre de recherche vers la variable présente au niveau $k-1$ qui n'a pas encore été assignée, afin de tester toutes les autres branches restantes de l'arbre. Ceci se répète pour le reste des variables, jusqu'à satisfaction de la formule F, en affichant la solution, ou par insatisfaction après avoir vérifié toutes les possibilités implicites de la formule F donnée.

Algorithme 2 Algorithme DPLL (Davis-Putnam-Logemann-Loveland)

Entrée: Une formule Φ en CNF
Sortie: Satisfiabilité de Φ et une solution satisfaisable
si $\Phi = \varnothing$ **alors**
retourner « satisfaisable » ;
fin si
$\Phi \leftarrow$ PropagationUnitaire(Φ) ;
si Φ contient une clause vide **alors**
retourner « Insatisfaisable » ;
fin si
Sélection d'un variable l dans C_Φ grâce à l'heuristique utilisée ;
si DPLL($\Phi \bigcup l$) retourne satifaisable **alors**
retourner « Satisfaisable » ;
fin si

Algorithme 3 Algorithme « PropagationUnitaire »

Entrée: Une formule Φ en CNF
Sortie: Φ simplifiée
tant que il n'y a pas de clause vide et qu'une clause unitaire $C \in \Phi$ existe **faire**
Considérer C une clause unitaire et l son littéral non affecté ;
$\Phi \leftarrow$ Simplifier(Φ,l) ;
fin tant que
retourner Φ ;

Algorithme 4 Algorithme « Simplifier »

Entrée: Une formule Φ en CNF, un litteral l
Sortie: Φ simplifiée
tant que clause $C \in \Phi$ **faire**
si $l \in C$ **alors**
$\Phi \leftarrow \Phi - \{C\}$;
sinon si $C(\bar{l} \in C)$ **alors**
$C \leftarrow C - l$;
fin si
fin tant que

2.5.2.3 Algorithme de résolution CDCL

Une nouvelle extension de DPLL, appelée CDCL, a été inventée par Marques-Silva et Sakallah en 1996 [33] [37], et été amélioré en 2001 par Moskewicz et al. Cette extension permet de faire un retour arrière non chronologique dans le but d'apprendre des nouvelles clauses et d'analyser les conflits. La grande différence entre les algorithmes DPLL et CDCL, et qui constitue l'avantages majeur de ce dernier se situe dans le saut d'une partie de l'arbre, appelé un back jumping, et qui permet d'ignorer une partie de l'arbre de recherche ne possédant pas de solutions.

La logique de l'algorithme CDCL repose sur l'élimination des sous arbres de recherche présentant une inconsistance et un conflit pendant les analyses précédentes des clauses. Ce processus est appelé l'apprentissage.

L'algorithme CDCL utilise généralement un graphe d'implication permettant de mettre en mémoire les différents parcours qui ont été suivis lors de l'interprétation partielle des clauses, ainsi que l'analyse des conflits rencontrés pendant la recherche de solutions. Son intelligence se présente au niveau du backtracking, tout comme pour le DPLL, sauf que les solveurs utilisant CDCL, mettent à jour les poids des variables figurant dans les clauses analysées, comme le fait l'heuristique dynamique VSIDS [34], présentée dans la section suivantes, à côté d'autres heuristiques. Contrairement à DPLL qui utilise les algorithmes complets avec des parcours de recherche en profondeur, CDCL quant à lui, se base sur les algorithmes et méthodes de recherche incomplets, en ne donnant qu'une idée partielle des implications entre les littéraux. L'algorithme 5 détaille son fonctionnement.

Les solveurs CDCL sont utilisés pour résoudre des problèmes pratiques provenant de plusieurs disciplines comme la cryptographie, l'industrie, l'intelligence artificielle, et l'ordonnancement. En effet, les solveurs utilisant l'algorithme CDCL sont capables de gérer des problèmes de plusieurs centaines de milliers de variables. La première étape est consituée par la modélisation de la formule en CNF. Ensuite, vient l'étape d'analyse où est présente l'intelligence de l'algorithme, coupant les parties de l'arbre de recherche générant des conflits, permettant d'accélérer le parcours de l'arbre. Ceci est fait par les explorations successives en utilisant l'apprentissage, l'élément clé du CDCL, car tous les conflits déjà rencontrés pendant les précédents parcours ont été éliminés, où d'autres ont été mis en mémoire comme informations utiles, comme l'ajout des clauses dans la base de clauses après chaque conflit rencontré, mais qui présentera toujours le même problème avec quelques modifications qui le rendront facile à analyser. Ces modifications sont faites par l'ajout des négations des littéraux qui ont mené des clauses à un conflit En effet, il faudra créer des disjonctions de conjonctions des clauses présentant des conflits, afin d'arrêter rapidement la perte de temps à analyser des parties inutiles de l'arbre de recherche. Dans un tel cas, le conflit n'est pas représenté lors d'un nouvelle analyse du problème avec CDCL, tout simplement car si la propagation unitaire a été utilisée, un littéral sera satisfait si toutes les autres des différentes clauses ont été contredits. Ci-dessous, quelques définitions utilisées dans le graphe d'implication.

Definition 20 *Qand deux littéraux de la même variable ne peuvent satisfaire des clauses pré-*

sentées d'un problème donné, un conflit apparaît dans le graphe d'implication.

Definition 21 *L'ensemble qui contient les nœuds du problème menant à un conflit est appelé : « côté conflit du graphe ».*

Definition 22 *L'ensemble qui contient les nœuds du problème aidant à une décision en éliminant les conflits est appelé « côté raison du graphe ».*

La coupure dans l'arbre de recherche, présente aussi dans le graphe d'implication, est faite au niveau des deux côtés : conflit et raison.

Algorithme 5 Algorithme CDCL

Entrée: Une formule Φ en CNF
Sortie: CDCL(F)

$V_{aff} \leftarrow \varnothing$, $profondeur \leftarrow 0$;
tant que Vrai **faire**
  pour chaque $l \in V_{aff}$ **faire**
    $F \leftarrow F \backslash l$;
  fin pour
  si $\exists$ une clause vide **alors**
    si profondeur = 0 **alors**
      retourner Faux ;
    fin si
    $C \leftarrow$ la clause « conflit » déduite ;
    $l \leftarrow$ littéral $\in C$ affecté à la profondeur du confilt ;
    profondeur $\leftarrow \max\{$ profondeur(v) / $\forall v \in C \backslash l\}$;
    $V_{aff} \leftarrow V_{aff} \backslash \{\forall v$ / profondeur(v) > profondeur ;
    $V_{aff} \leftarrow V_{aff} \cdot l$;
    $F \leftarrow F \bigcup C$;
  sinon
    si $V_{aff} = V$ **alors**
      $\{V$ est l'ensemble des variables de $F\}$
      retourner Vrai ;
    fin si
    $l \leftarrow$ choisir un littéral d'une variable non affectée ;
    $V_{aff} \leftarrow V_{aff} \cdot l$;
    profondeur $\leftarrow$ profondeur $+1$
  fin si
fin tant que

2.5.3 Heuristiques de choix de variables

L'algorithme DPLL a connu diverses variantes, tout d'abord avec des différences au niveau du choix des variables de décision à examiner (qui peut être à priori quelconque à chaque

étape et niveau de nœud de l'arbre) ou heuristiques de sélection [35], et deuxièmement, et deuxièmement dans le backtrack effectué pour les formes des données adoptées. Le choix de variables est primordial pour la simple raison que, s'il est bien effectué, il peut simplifier le parcours de l'arbre, par l'élimination de branches non importantes, tôt durant la recherche de solution. Comme présentée dans la section précédente, la propagation unitaire joue également un rôle primordial pour couper les branches de l'arbre et ainsi simplifier la résolution de la formule F.

Une heuristique est une méthode ayant pour object de simplifier la résolution des problèmes, en favorisant le retour de bonnes solutions, mais qui sans la moindre garantie d'optimalité. On fait souvent appel à des heuristiques [36] [20], lorsque l'on est confronté à la difficulté de résoudre en un temps de recherche réduit, ou tout du moins acceptable, un problème de grande taille, c'est-à-dire, à l'espace de recherche extrêmement vaste. Une bonne heuristique, est une méthode qui rend une solution proche de l'optimum. Elle est également caractérisée par sa complexité acceptable, de l'ordre d'une complexité polynomiale, et par sa capacité à « éviter les mauvaises solutions ».

Une heuristique efficace a pour objectif principal de réduire la temps et la taille de l'espace de recherche des solution, tout en minimisant l'explosion combinatoire. Ce qui ne se fera qu'en diminuant la profondeur de l'arbre de recherche, par la minimisation du nombre de choix de variables de décision à faire pour chaque nœud et niveau [37].

Le choix du temps consacré au choix de chaque variable au regard du nombre total de nœuds dans l'arbre, détermine généralement la complexité de la création d'une heuristique. Par exemple, une heuristique parcourant tous les nœuds en diminuant la taille de l'arbre et avec un temps de parcours lent pourra s'avérer une heuristique moins performante qu'une autre capable de traiter rapidement plus de nœuds.

En recherche arborescente, deux stratégies se présentent : la recherche statique et la recherche dynamique. La recherche statique se caractérise par la définition d'un ordre statique pour le choix des valeurs et variables. La recherche dynamique fait au contraire le choix des valeurs ou des variables de manière reconfigurable avec pour objectif de trouver une solution rapidement. Ces deux approches peuvent déterminer le niveau d'efficacité des algorithmes DPLL et CDCL créés à l'aide des heuristiques correspondantes employées pour le choix des prochaines variables à être assignées. L'ordre de sélection des variables peut tout aussi bien rendre le temps de résolution exponentiel, comme il peut le réduire en minimisant le nombre des nœuds à traverser. Néanmoins, choisir une variable optimale à tester, à n'importe quelle étape de la recherche arborescente est en soit-même un problème NP-complet [38, 39].

De nombreuses heuristiques [40] ont fait preuve de leurs efficacités à rendre les algorithmes DPLL et CDCL plus performants. Nous en exposons quelques unes parmi les plus utilisées dans la résolution des problèmes SAT.

2.5.3.1 MOMS

MOMS (*Maximum Occurrences in clauses of Minimum Size*) [36, 41] a pour objectif de favoriser les petites clauses en choisissant la variable qui a le plus grand nombre d'occurrences dans les plus petites clauses de la formule. En effet, le choix de cette variable va réduire la taille les clauses en déterminant le bon assignement, et donc simplifier le reste des clauses à l'aide de l'emploi de la propagation unitaire décrite précédemment.

Pourtant, cette heuristique n'est pas optimale car elle s'appuie toujours sur la propagation unitaire. Elle a de plus un inconvénient majeur qui est celui du traitement des clauses unitaires créées par une suite de propagations des variables qui ne sont prises en compte, contrairement à celles produites immédiatement lors des affectations **[JW90, DABC96, DB96]**. Le passage par la propagation unitaire était toujours obligatoire, on en déduit alors que pour éviter cet inconvénient, une utilisation immédiate de la propagation unitaire s'impose comme unité de mesure.

2.5.3.2 UP

UP (*Unit Propagation*) ou propagation unitaire a pour objectif de simplifier l'arbre de recherche en calculant le nombre de répétition de chaque littéral des variables, que l'on appelle score, et qui doit être minimisé. C'est l'heuristique la plus utilisée dans les évolutions de l'algorithme DPLL. Cette heuristique est généralement utilisée par les solveurs de la famille SATZ [30].

2.5.3.3 VSIDS

VSIDS (*Variable State Independet Decaying Sum*) [34], ou somme de décomposition indépendante de l'état variable, repose sur un principê de traitement différent de MOMS et UP. Cette heuristique est basée sur des compteurs liés au variables, nommés activités des variables. Dès qu'une activité est grande, c'est-à-dire, que le compteur d'une variable présente une grande valeur, la variable associé à ce compteur est choisie pour traitement. Cependant si un conflit apparaît, une analyse est faite pour déterminer les facteurs qui l'ont dévoilé, et trivialement, également les variables qui l'ont causé et comme résultat involontaire, l'activité de ces variables augmente. L'avantage de cette heuristique par rapport à MOMS, c'est qu'elle est souvent associée à des structures de données nommées paresseuses, auxquelles MOMS est inapplicable. Malgré cet avantage, elle présente également un inconvénient au niveau du choix des variables à vérifier au sommet de l'arbre, qui est généralement aléatoire. Hors, comme il a été dit précédemment, une heuristique efficace se caractérise par le choix des variables qui agissent fortement sur l'espace de recherche de l'arbre.

2.5.3.4 BOHM

L'heuristique BÔHM [42], du nom de son inventeur, est basée sur un algorithme de recherche en backtrack, qui a prouvé il y a longtemps son efficacité pour la résolution des ins-

tances aléatoires de SAT. Le traitement de cette heuristique se fait par la sélection, à chaque étape de l'algorithme, d'une variable ayant un vecteur maximal $(H_1(x), H_2(x), \cdots, H_n(x))$ tel que :

$$H_i(x) = \alpha \max(h_i(x), h_i(\neg x)) + \beta \min(h_i(x), h_i(\neg x)) \tag{2.1}$$

Après calcul, $H_i(x)$ représente le nombre de clauses qui n'ont pas été résolues, pour les i littéraux contenant un littéral l. On remarque ainsi que cette heuristique, en favorisant les petites clauses, à des points en commun avec MOMS, en sélectionnant un littéral qui les rendent satisfaisables, quand il est assigné à vrai, ou bien en minimisant la taille de ces dernière quand il est assigné à faux.

2.5.3.5 DLCS

DLCS (*Dynamic Largest Combined Sum*), ou plus grande somme combinée dynamique, est une heuristique qui permet de compter le nombre de clauses contenant un littéral l d'une variable v : si le littéral l apparait plus souvent que le littéral $\bar{l}$ dans une formule, alors on affecte l à vrai, l'inverse sinon. DCLS est une heuristique simple,un peu plus rapide que Jeroslow-Wang unilatérale [42].

2.5.3.6 DLIS

DLIS (*Dynamic Largest Individual Sum*), ou plus grande somme individuelle dynamique, est une heuristique qui ne diffère pas beaucoup de DLCS. Cette heuristique traite les occurrences d'un littéral l, en ignorant sa négation. Elle est plus rapide pour trouver des résultats que DLCS et avec de meilleurs choix : on choisit aléatoirement un littéral qu'on juge être le plus efficace et qui rendra le plus de clauses satisfaites. Variante : choisir α qui maximise score$(\alpha) = \sum_{\alpha \in C, C \in \Phi} 2^{-|C|}$. Sur certains benchmarks, elle est deux fois plus rapide que Jeroslow-Wang unilatérale [42].

2.5.3.7 RAND

RAND, ou random, ce qui veut aléatoire, est une heuristique qui permet de choisir au hasard la prochaine variable à affecter et qui permettra de satisfaire le plus de clause. Elle a en commun avec les heuristiques DLCS et DLIS, le comptage du nombre d'apparition des littéraux l et $\bar{l}$ dans des clauses vivantes, pour choisir le plus efficace.

2.6 Transcodage ILP vers SAT

Dans le chapitre 1, nous avons introduit le formalisme ILP en $\{0, 1\}$ pour formaliser des problèmes de production. Cependant, c'est avec le formalisme SAT que nous voulons résoudre ces problèmes. Ils nous est donc nécessaire de disposer d'une méthodologie pour transcoder le

problème depuis ILP en $\{0,1\}$ vers SAT.

Il est possible, en appliquant les quelques règles énoncées ci-dessous, de transcoder des problèmes ILP simples, non bornés aux solutions en $\{0,1\}$.

Considérons deux variables $X, Y \in \{0,1,\cdots,N\}$ d'un problème ILP. À ces deux variables, nous faisons correspondre deux ensembles de variables booléennes de substitution $\rightarrow X_0^*, X_1^*, \cdots, X_N^*$ et $Y_0^*, Y_1^*, \cdots, Y_N^*$ pour l'encodage en SAT (par souci de clarté, nous marquons d'un astérisque les variables du problème SAT pour les distinguer de celles du problèmes ILP. Tout d'abord, X et Y doivent prendre des valeurs. Ceci doit être matérialisé par le faut qu'une et une seule variable de chaque ensemble prend la valeur $Vrai$, soit $X_i^* = Vrai$ et $Y_j^* = Vrai$. Toutes les autres sont assignées à $Faux$. Cela se formalise de la manière suivante :

— $\bigvee_{k=0}^{k=N} X_k^* = Vrai$

— $\bigvee_{k=0}^{k=N} Y_k^* = Vrai$

— $\forall i,j \in \{0,1,\cdots,N\},\ (\neg X_i^* \vee \neg X_j^*) = Vrai,\ (\neg Y_i^* \vee \neg Y_j^*) = Vrai$

Alors, on peut utiliser les règles suivantes pour transformer des inégalités du problème ILP en clauses du problème SAT équivalent.

— $X < K \rightarrow \forall i \in \{K, K+1, \cdots, N\},\ (\neg X_i^*) = Vrai$

— $X < Y \rightarrow \forall i <= j \in \{0,1,\cdots,N\},\ (Y_j^* \Rightarrow \neg X_i^*) = (\neg Y_j^* \vee \neg X_i^*) = Vrai$

— $X \neq Y \rightarrow \forall i,j \in \{0,1,\cdots,N\},\ (\neg X_i^* \vee \neg Y_j^*) = Vrai$

Par extension, la méthode peut être extrapolée au cas ILP à solutions en $\{0,1\}$

À chaque variable du problème ILP à solutions en $\{0,1\}$ on fait correspondre une variable équivalente SAT, $X \rightarrow X^*$. Typiquement, on trouve dans le problème d'origine, des égalités et des inégalités arithmétiques qu'il faut pouvoir transcoder. On peut se baser sur les quelques règles suivantes, simples à extrapoler aux problèmes de grande taille, pour effectuer ce transcodage. Pour renforcer le fait que toute clause C au sein d'une formule F CNF doit être évaluée à $Vrai$, nous le traduirons explicitement ci-dessous dans l'écriture $C = Vrai$.

— pour une égalité,
$X_i = Y_j \rightarrow X_i^* = Y_i^*$

— une inégalité du type de celles de la section 1.13.2, où D est une constante positive,
$\sum_{k=i}^{k=j} X_k^* < D$ avec $j - k = D + 1 \rightarrow \exists k \in \{i,\cdots,j\},\ \neg X_k^* = True$

— un événement singulier matérialisé par un ensemble de variables X_k^* sera transcrit

$\forall k \in \{1, \cdots, N\},\ (\neg X_k^*) = Vrai$

— lorsque l'indice représente un intervalle temporel, un événement temporel singulier ne pouvant avoir lieu après (resp. avant) une date T donnée se transcrira
$\forall k \in \{0, \cdots, T-1\},\ \neg X_k^* = True$ (resp. $\forall k \in \{T+1, \cdots, N\},\ \neg X_k^* = True$)

— l'occurrence obligatoire d'un événement singulier avant un autre, matérialisés respectivement par les ensemble de variables X_i^* et Y_j^*, se transcrira
$\forall k \in \{0, \cdots, T-1\},\ \neg X_k^* = True$ (resp. $\forall kin\{T+1, N\},\ \neg X_k^* = True$)

2.7 Conclusion

Dans ce chapitre, nous sommes passés du formalisme ILP en $\{0, 1\}$ introduit au chapitre 1 pour représenter des problème de production industrielle, au formalisme SAT de logique booléenne propositionnelle. Le formalisme ILP en $\{0, 1\}$ permettant une transcription directe bien plus aisée du problème de production que ne le permet le formalisme SAT que nous voulons utiliser pour la résolution. C'est à ce titre que nous l'avons utilisé comme étape intermédiaire, et proposé des règles de transcodage de ILP en $\{0, 1\}$ vers SAT.

Après avoir présenté la problématique et le formalisme SAT, et introduit les notion de base de complexité et de problèmes NP-complets, nous avons présenté les paradigmes à la base de l'exploration arborescente de l'espace de recherche des problèmes SAT. Les algorithmes fondamentaux DPLL et CDCL sur lesquels se basent la plupart des algorithmes des recherches ainsi que les principales heuristiques employées pour réduire/accélérer la recherche dans ont présentés.

Le chapitre 3 s'ouvre vers les approches de résolution basées sur l'accélération matérielle employant des technologies électroniques reconfigurables de type FPGA. Après un état de l'art sur ces approches, nous y proposons notre architecture matérielle de solveur dédiée à la résolution de problèmes k-SAT.

Chapitre 3

Solveur matériel dédié à la résolution de k-SAT : contribution et résultats expérimentaux

Sommaire

3.1 Introduction

Le chapitre 2 nous a permis d'introduire le formalisme SAT et d'aborder les méthodes ainsi que les algorithmes et heuristiques mis en œuvre pour la résolution du problème SAT, dans les solveurs dédiés, essentiellement logiciels. Il s'agit du formalisme par l'intermédiaire duquel nous souhaitons résoudre le problème de production présenté dans la section 1.14.1

de chapitre 1. Hors, un problème réel dans ce domaine peut facilement arborer une taille conséquente, entrainant des temps de résolution exponentiels. C'est l'une des principales motivations actuelles pour le développement de travaux ayant pour objectif l'accélération de la résolution du problème SAT par l'utilisation d'accélérateur matériels. Notre démarche s'inscrit dans cette orientation, notre objectif étant de proposer une nouvelle architecture matérielle de solveur, spécifiquement dédiée à la résolution de SAT.

Dans le chapitre 2, nous avons également décrit la méthodologie que nous déployons pour convertir en formalisme SAT un problème de type production préalablement modélisé selon le formalisme ILP en $\{0, 1\}$ introduit en section 1.14.2.

Ce nouveau chapitre se concentre donc sur la conception d'un solveur k-SAT purement matérielle, et indépendant de l'instance du problème traité. Il est organisé en deux parties principales.

Afin de bien cerner le sujet, la première partie s'intéresse aux architectures hybrides logicielles/matérielles et matérielles seules proposées dans la littérature pour les solveurs SAT, ainsi qu'à leurs performances et limites. Cette partie s'achève sur un bilan comparatif.

La deuxième partie est dédiée à la présentation l'architecture du solveur que nous proposons ainsi que l'analyse des caractéristiques et fonctionnement des trois blocs qui le composent, à savoir, SAT_CACHE, SAT_CORE et SAT_JOURNAL. La stratégie de parcours de l'arbre de recherche implémentée par le solveur est décrite, ainsi que les choix qui y ont contribué : parallélisme massif de traitement et limitation des risques d'inactivité potentiel liés à des accès aux données lents. Cette architecture est générique au sens où sont paramétrables, lors de la synthèse, des caractéristiques du solveur qui déterminent l'arité k, le nombre max de variable des problèmes k-SAT que le solveur peut traiter, où le degré de parallélisme de l'architecture.

Le chapitre termine par la présentation de résultats de simulations et les résultats expérimentaux d'implantation sur FPGA. Afin de valider l'approche proposée. Ces résultats ont été réalisés sur la carte FPGA NexysVideo de chez Digilent (basée sur un FPFA Xilinx de la famille Axtix-7), en générant plusieurs solveurs différent entre eux par le choix des paramètres de synthèse..

3.2 État de l'art des architectures hybrides ou purement matérielles

3.2.1 Suyama et al.

L'architecture proposée par Suyama et al. [43, 44, 45, 46, 47] est basée sur un solveur SAT spécifique à l'instance qui permet, en fonction de l'instance donnée du problème, de retourner toutes les solutions ou un nombre d'entre elles. Cet algorithme est caractérisé, à chaque moment du traitement, par l'affectation d'une variable qui est évaluée par la suite. Des techniques ont été suggérées par Suyama et al. afin de choisir la variable de décision suivante.

Ces techniques sont et statiques [43], et dynamiques [44, 45, 46, 47]]. Ce qui caractérise la première technique dynamique est la propagation unitaire expérimentale. C'est l'obligation d'affecter expérimentalement les deux valeurs possibles à chaque variable libre pour en déterminer plus tard avec un nombre maximum d'implications à générer par les variables et valeurs sélectionnées. La deuxième technique est basée sur l'heuristique MOMS (Maximum Occurrence in Clauses of Minimum Size) qui permet de choisir et ainsi vérifier la formule donnée tout en choisissant la variable qui apparaît le plus souvent dans les clauses binaires de petites tailles [44]. L'architecture est composée de plusieurs blocs : contrôle, évaluation, backtrack, branch et résultat, comme le montrent les figures 3.1 et 3.2.

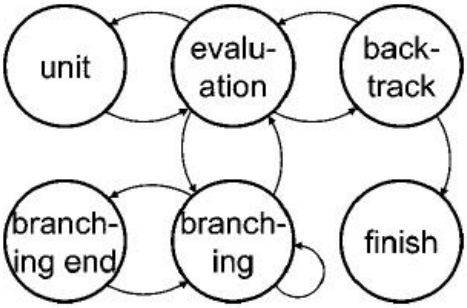

FIGURE 3.1 – Machine d'état de Suyama et al.

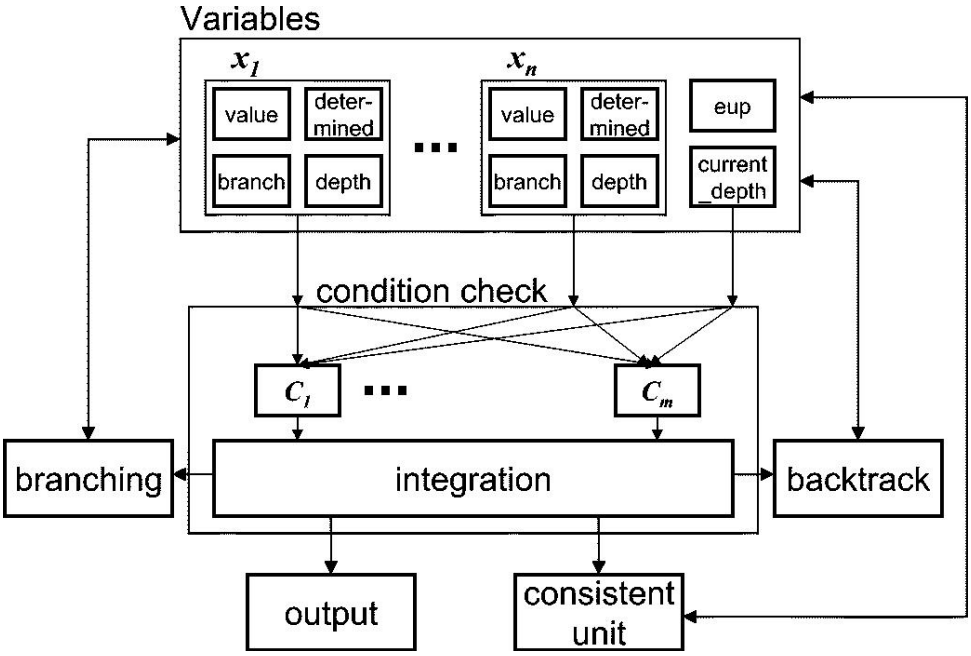

FIGURE 3.2 – Architecture de Suyama et al.

Le bloc de contrôle enregistre les valeurs affectées à chaque variable pour chaque niveau de l'arbre. Ces affectations sont faites, soit aléatoirement soit par décision. Ensuite, les valeurs de toutes les variables sont injectées dans les clauses respectives, qui sont à leur tour évaluées

simultanément. Puis, les résultats de l'évaluation de chaque clause sont combinés et, selon le résultat final, on procède soit à la ramification, soit au backtrack.

Chaque formule CNF doit être d'abord convertie au format 3-SAT (dans lequel le nombre de littéraux par clause est égal à 3) en introduisant des variables auxiliaires. Ensuite, un programme en C spécialement développé, analyse la formule résultante et génère la description HDL comportementale correspondante. Sur la base de ce code HDL, un circuit spécifique à l'instance est synthétisé et mis en œuvre à l'aide d'outils disponibles dans le commerce. En utilisant cette stratégie, un certain nombre de circuits ont été mis en œuvre sur un FPGA Altera FLEX10K250 cadencé à 10 MHz. Suyama et al. ont été capables de réaliser des accélérations de 1 à 10 fois par rapport à l'algorithme POSIT (PrOpositional SatIsfiability Testbed) [48] exécuté sur un UltraSPARC-II / 296 MHz sur certains exemples de la DIMACS (Center for Discrete Mathématiques & Informatique Théorique) [27]. Cependant, le temps consacré à la compilation et à la configuration du matériel (environ une heure) n'a pas été pris en compte. Lorsqu'une formule CNF ne peut loger dans le FPGA, il a été proposé de diviser le circuit sur plusieurs FPGA. Toutefois, en raison des fortes en matière de communications FPGA, l'utilisation résultante de la logique était très faible (environ 13%) [44].

3.2.2 Zhong et al.

Zhong et al. ont implémenté un solveur SAT, spécifique à l'instance, basé sur l'algorithme DP [29]. Dans leur travail initial [49], ils ont construit un circuit d'implication et une machine à états finis (FSM) pour chaque variable de la formule, toutes les machines d'état étant interconnectées dans une chaîne série, comme cela est montré dans la figure 3.3(a).

À chaque période de temps, une seule machine d'état finie (FSM) est active. Dès que cette machine d'état termine son traitement, elle transfère le contrôle, soit à la prochaine (progression recherche), soit à la précédente (backtracking). Chaque machine d'état connaît la valeur actuelle de sa variable (qui peut être « 0 », « 1 » ou libre) et sait si cette valeur a été attribuée ou implicite. Une solution est trouvée si la dernière machine d'état (droite) essaie d'activer la machine d'état suivante. Si la première machine d'état (gauche) essaie de passer le contrôle à gauche, il n'existe pas de solution. Comme étape de prétraitement, toutes les variables sont triées en tenant compte du nombre de leurs apparitions dans une formule donnée. Cet ordre statique est utilisé pour organiser les variables dans la chaîne série

Les principaux inconvénients de l'architecture proposée dans [49] sont : une fréquence d'horloge faible (comprise entre 700 KHz et 2 MHz selon les formules traitées) et un temps de compilation très long (jusqu'à plusieurs heures sur un Sun 5 / 110MHz / 64MB). Dans le but d'améliorer les performances, la mise en œuvre matérielle du backtrack non chronologique a été proposée dans [50]. Cela n'a toutefois pas conduit à des améliorations significatives et a donc motivé la révision des décisions de conception initiales. Par conséquent, une structure d'interconnexion à anneau régulière a été utilisée au lieu de lignes globales irrégulières, réduisant essentiellement le temps de compilation à l'ordre de quelques secondes et augmentant la cadence d'horloge (jusqu'à 20-30 MHz) [51, 52]. En outre, une technique permettant de générer et d'ajouter des clauses de conflit a été proposée. La conception décrite dans [18, 36]] est différente de [17], avec les clauses (toutes de même taille) disposées dans des éléments de

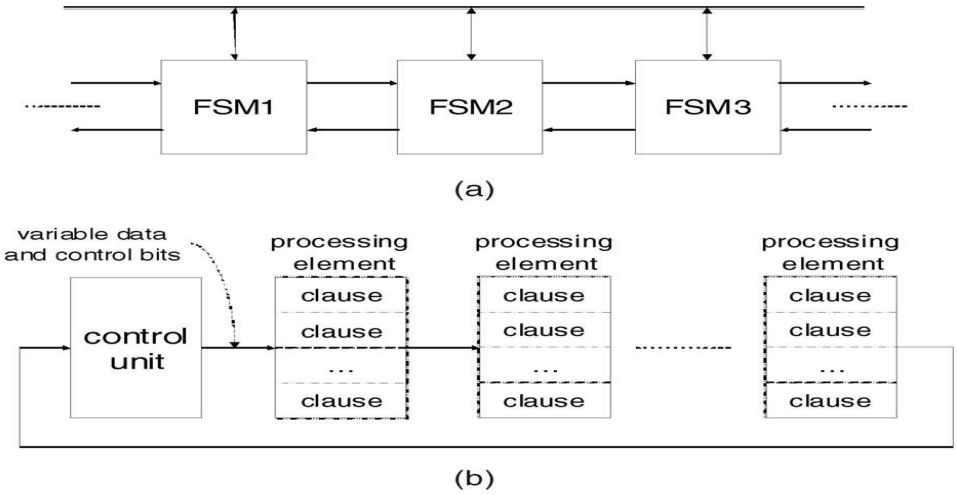

FIGURE 3.3 – Architecture de Zhong et al.

traitement répartis le long d'un réseau de communication en pipeline, comme cela est montré dans la figure 3.3(b). L'unité de commande principale conserve l'état actuel du solveur SAT et surveille le réseau pour détecter les changements de valeur et les conflits de la variable.

Pour traiter les grandes instances de problèmes qui ne logent pas dans un seul FPGA, les auteurs proposent d'utiliser plusieurs FPGAs interconnectés (les architectures [49] et zhong1998solving sont facilement évolutives). Les résultats expérimentaux sont basés à la fois sur la mise en œuvre matérielle (sur un émulateur IKOS contenant un certain nombre de matrices de FPGA) et sur la simulation. Les accélérations réalisées sur le logiciel GRASP [52] s'exécutant sur un Sun5 / 110MHz / 64MB (en mode restreint), y compris la compilation matérielle et le temps de configuration, sont d'un ordre de grandeur [36] pour un sous-ensemble du DIMACS SAT repères [27]. Il convient de noter que les architectures proposées par Zhong et al. sont parmi les plus largement connues et ont servi de prototypes pour un certain nombre de logiciels reconfigurables SAT développés par la suite.

3.2.3 Platzner et al.

Le solveur SAT proposé par Platzner et al. [53, **?**] est similaire à celui de Zhong et al. [49]. Il se compose d'une colonne de machines à états finis, d'une logique de déduction et d'une unité de contrôle globale comme présenté dans la figure 3.4. Il met en œuvre un algorithme basé sur DP pour les formules CNF. La logique de déduction calcule le résultat de la formule à partir de l'affectation de la variable partielle courante. Toutes les assignations de variables sont essayées dans un ordre fixe. Initialement, toutes les variables sont non assignées et l'unité de contrôle active la première FSM. Celle-ci essaie d'assigner « 0 » à sa variable et la logique de déduction calcule le résultat. Si la formule est évaluée à « 1 », une solution a été trouvée. Sinon, si la formule est évaluée à « 0 », la FSM complète sa valeur. Si la formule est évaluée à « x », la FSM suivante est activée. Si une FSM tente les deux assignations de variable et la formule s'évalue toujours à « 0 », la FSM réinitialise sa valeur et passe le contrôle à la FSM précédente.

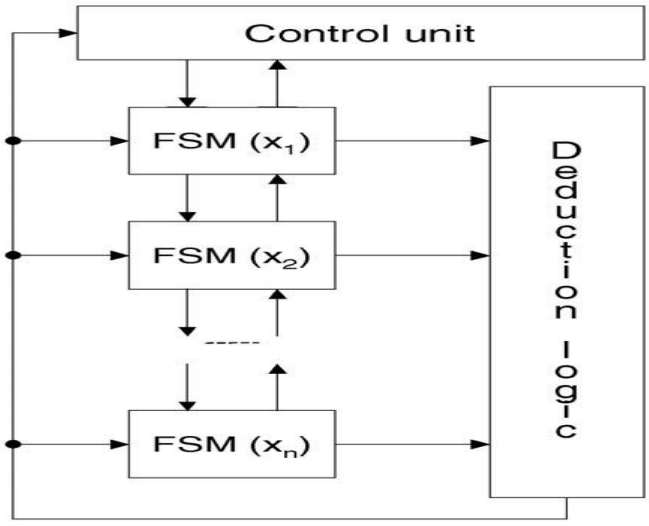

FIGURE 3.4 – Architecture de Platzner et al.

Les auteurs ont mis en œuvre un prototype d'accélérateur sur la base d'une carte Pamette contenant quatre FPGA Xilinx XC4028. Les accélérations obtenues pour les repères trou6 à trous10 de DIMACS SAT [27], y compris la compilation matérielle et le temps de configuration, s'échelonnent de 0.003 à 7.408 comparé à GRASP exécuté sur un PII / 300MHz / 128MB [19]. La fréquence d'horloge dans [53] varie de 65 MHz (pour le trou6) à 27 MHz (pour le trou10). Le temps de compilation matériel domine le temps d'exécution du matériel pour toutes les instances de problème considérées et, par conséquent, constitue la principale limitation de cette architecture. En outre, aucune solution n'est proposée pour le cas où une formule donnée n'est pas compatible avec FPGA choisi.

3.2.4 Abramovici et al.

Abramovici et Saab [54] ont appliqué une technique de modélisation d'une formule booléenne donnée (pas nécessairement au format CNF) par un circuit arbitraire. Le solveur SAT conçu est basé sur l'algorithme PODEM [55] qui est généralement utilisé pour résoudre les problèmes de génération de test. Ici, l'objectif est de définir la sortie primaire d'un circuit logique combinatoire (qui représente une fonction booléenne à satisfaire) à « 1 » en trouvant une affectation appropriée des entrées primaires. Un concept important de cet algorithme est un « objectif », qui est l'affectation souhaitée d'une certaine valeur à un signal ayant initialement une valeur inconnue [54]. Un objectif ne peut être atteint que par des assignations d'entrées primaires. Une procédure de backtrack est utilisée pour propager un objectif le long d'un chemin vers les entrées primaires et détermine l'affectation d'entrée primaire qui est susceptible d'aider à atteindre l'objectif. Pour atteindre cet objectif, deux modèles du circuit ont été construits : un modèle avancé pour la propagation des affectations d'entrée primaire et un modèle en arrière pour les objectifs de propagation (voir dans la figure 3.5(a).

Dans [56], une architecture améliorée est proposée, représentée dans la figure 3.5(b), qui utilise l'algorithme DP pour les formules CNF (modélisées par un circuit à deux niveaux) et

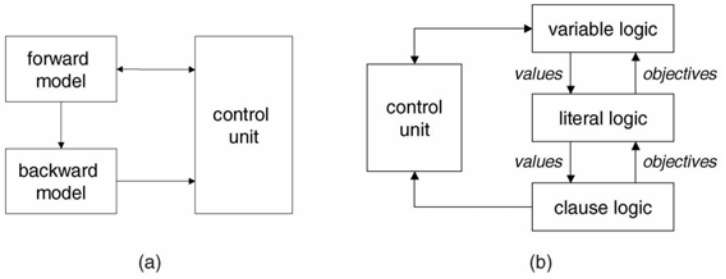

FIGURE 3.5 – Architecture de Abramovivi et al.

implémente une stratégie de sélection de variable améliorée. Le bloc logique variable stocke les valeurs courantes de toutes les variables. Ces valeurs sont envoyées au bloc logique littéral, qui les distribue entre les clauses. Le bloc logique de clause évalue chaque clause et la formule entière et détermine également les valeurs souhaitées pour les littéraux (objectifs). Les objectifs sont renvoyés au bloc logique littéral, qui combine les objectifs issus de différentes clauses en un seul objectif pour chaque variable. Enfin, le bloc logique variable transforme les objectifs en affectations variables (implications ou décisions). L'unité de contrôle déclenche le retour en arrière lorsque la formule est évaluée à « 0 ». Il est important que tous les objectifs se propagent simultanément le long de tous les chemins possibles, ce qui permet l'attribution simultanée de plusieurs variables.

Pour la mise en œuvre du matériel, Abramovici et al. ont suggéré de créer une bibliothèque de modules de base ayant un emplacement et un routage interne prédéfinis et devant être utilisés pour n'importe quelle formule. Le circuit solveur est construit à partir de modules, ce qui réduit le temps de compilation à de l'ordre de quelques minutes. Les auteurs ont mis en place des circuits simples sur le FPGA XC6264 et ont simulé les plus grands. Pour un circuit occupant toute la zone du XC6264 FPGA, la fréquence d'horloge était d'environ 3,5 MHz. Dans [56], Abramovici et Sousa rapportent des accélérations brutes (ne comprenant pas les coûts généraux de compilation et de configuration) de 0,01 à 7000 (après ajustement de l'unité de temps) obtenues sur GRASP [52] pour un sous-ensemble de repères DIMACS SAT [27]. Dans [56], un système de logique virtuelle a été proposé permettant de construire des circuits pour résoudre des instances de problème SAT plus grandes que les ressources matérielles disponibles. C'est obtenu en décomposant une formule en sous-formules indépendantes qui peuvent être traitées dans des FPGA séparés, soit simultanément, soit séquentiellement. La caractéristique importante de la méthode suggérée est que les sous-formules peuvent être résolues dans n'importe quel ordre où les signaux entre FPGA ne sont pas nécessaires. Toutefois, lorsque les ressources matérielles disponibles sont trop insuffisantes pour la taille du problème d'origine, la décomposition peut prendre un temps excessif et produit trop de sous-problèmes.

3.2.5 Dandalis et al.

Dandalis et al. [57, 58] ont proposé une architecture pour évaluer les clauses en parallèle pendant la phase de déduction de l'implication. Une caractéristique distinctive de leur approche est que le circuit évolue dynamiquement lors de l'exécution, en essayant d'atteindre un

niveau de parallélisme plus adéquat et, par conséquent, d'optimiser les performances. Dans l'architecture proposée, toutes les clauses d'une formule CNF sont divisées en p groupes, ce qui déduit les implications en parallèle. Ainsi, les implications sont résolues de manière indépendante dans chaque groupe et, par la suite, un processus de fusion combine tous les résultats permettant la création de la prochaine variable. Si un conflit est détecté, le processus de déduction est terminé et le retour en arrière est effectué. Si aucune implication ne se produit, une nouvelle variable de décision est sélectionnée et une valeur est attribuée. Sinon, l'assignation de variable résultante est renvoyée à l'évaluateur de clause et l'ensemble de la procédure est répété jusqu'à ce qu'aucune autre implication ne soit déduite ou qu'un conflit ne soit détecté. Un exemple de l'architecture de l'évaluateur de la clause pour $p = 3$ est représenté dans la figure 3.6. Les processus de décision et de retour sont supposés être exécutés par un ordinateur hôte.

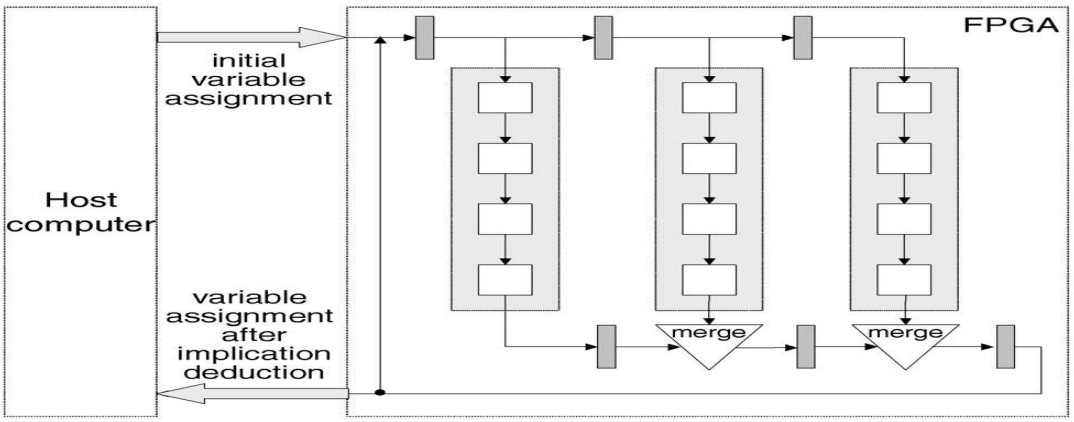

FIGURE 3.6 – Architecture de Dandalis et al.

Chaque groupe de clauses est semblable à celui proposé par Zhong et al. dans [51] [57] et est organisé comme un tableau linéaire de modules fonctionnant en pipeline comme cela est montré dans la figure 3.6. Chaque module correspond à une clause avec un nombre limité de littéraux et la structure d'un module est la même pour toutes les clauses. Les variables associées à une clause donnée sont stockées dans la mémoire locale du module respectif. Pour mettre à jour le contenu des mémoires locales (afin de correspondre à une instance de problème particulière), l'utilisation d'une reconfiguration partielle a été suggérée.

Lors de la résolution de problèmes, le circuit est reconfiguré pour implémenter des moteurs de déduction avec différents niveaux de parallélisme p. Dans cette tâche, étant donné un ensemble de circuits matriciels avec un nombre différent de modules par groupe, l'objectif est de trouver un modèle qui minimise le retard de propagation d'implication moyenne. Évidemment, ce temps dépend aussi de la répartition des clauses en groupes et de leur ordre relatif au sein du groupe. Toutefois, ces questions n'ont pas été examinées. Au lieu de cela, un algorithme heuristique gourmand a été proposé qui est exécuté par une machine hôte et essaye tous les modèles disponibles à tour de rôle, en commençant par celui qui a le nombre minimum de modules par groupe. Différents modèles sont testés alors que toute amélioration de la performance est

détectée. Ainsi, l'évolution de l'architecture consiste à trouver le nombre optimal de groupes pour une instance de problème donnée. L'exécution de l'algorithme SAT n'est pas redémarrée pour chaque modèle, c'est-à-dire après la commutation des modèles, le processus de recherche se poursuit avec l'affectation de la variable partielle précédemment trouvée. Les auteurs rapportent des accélérations de 1 à 6 fois [57] par rapport au cas où $p = 1$ pour un certain nombre de repères DIMACS SAT [27]. Ces résultats ont été obtenus à l'aide d'un simulateur de logiciel et ne reflètent pas les reconfigurations matérielles requises. On a également supposé que les ressources FPGA sont suffisantes pour résoudre une instance SAT.

3.2.6 Leong et al.

Leong et al. ont étudié la possibilité et l'efficacité de la réalisation d'algorithmes SAT complets [58] et incomplets [59, 60] dans des matériels reconfigurables [61]. En particulier, les boucles internes des algorithmes GSAT [58] et WSAT [60] pour le 3-SAT ont été partiellement associées au FPGA.

Dans l'architecture implémentant l'algorithme GSAT [59] (voir figure 3.7, une affectation de variable initiale est générée de manière aléatoire par un logiciel et téléchargée dans un FPGA Xilinx XC6200. Le matériel retourne successivement chaque variable et évalue la formule CNF, en essayant de maximiser le nombre de clauses satisfaites. La variable conduisant au nombre maximal de clauses satisfaites est envoyée au logiciel, qui calcule alors la prochaine affectation de variables en basculant cette variable.

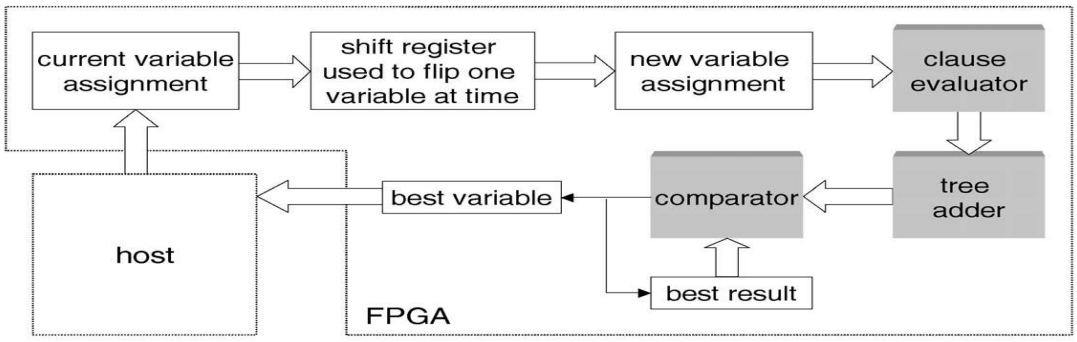

FIGURE 3.7 – Architecture de Leong et al.

Le circuit SAT a été synthétisé à partir de la description VHDL, à l'exception de l'évaluateur de clause, qui est un composant dépendant du problème et est donc personnalisé directement pour chaque instance de problème individuelle (en utilisant la possibilité de reconfiguration dynamique partielle, fournie par les périphériques de la famille XC6200). La conception a été testée en matériel pour les petites formules CNF (jusqu'à 50 variables et 80 clauses) et les résultats étaient soit pires soit comparables à la mise en œuvre logicielle de l'algorithme GSAT [59].

Dans [60], une implémentation similaire de la boucle interne de l'algorithme WSAT est décrite. La conception est destinée aux FPGA de la série Virtex et est basée sur un modèle

supportant au plus 50 variables et 170 clauses. Un circuit correspondant à une formule 3-SAT particulière est généré par un programme logiciel qui personnalise le fichier de flux de bits modèle. Une manipulation directe d'un flux binaire est beaucoup plus rapide qu'un cycle complet de synthèse, de mappage, de placement et de routage, qui est généralement effectué par des solveurs SAT spécifiques à l'instance. Mais les contraintes imposées au nombre maximal de variables et de clauses limitent l'application du circuit à des formules CNF assez simples.

Les résultats obtenus sur un FPGA XCV300 sous une fréquence d'horloge de 33 MHz pour un nombre d'instances SAT de DIMACS [27] montrent une accélération de 0,1 à 3,3 par rapport à la mise en œuvre de l'algorithme WSAT en logiciel (sur Sun SparcStation 20) [60]. Ces résultats incluent à la fois le temps nécessaire pour personnaliser et télécharger en FPGA le flux de bits modèle et le temps passé dans les communications entre le processeur hôte et le FPGA. Cependant, rien n'a été proposé pour résoudre des instances de problème dont les tailles dépassent les dimensions du FPGA.

3.2.7 Soussa et al.

De Sousa et al. [62, 63, 64] ont mis en place un algorithme de recherche basé sur DP pour le 3-SAT, doté d'un moteur de diagnostic qui est activé en cas de conflit et tente d'identifier les variables de décision directement responsables du conflit occurrence. L'analyse du conflit est également utilisée pour construire et ajouter de nouvelles clauses à une formule CNF, accélérant ainsi le processus de recherche. Il a été décidé de répartir le travail entre le logiciel et le matériel reconfigurable avec les tâches les plus intensives en calcul (telles que les implications informatiques et le choix de la prochaine variable de décision) affectées au matériel, tandis que les tâches orientées sur le contrôle (analyse des conflits, La gestion de base de données de la clause) sont effectués dans un logiciel.

Le solveur SAT suggéré a une architecture spécifique à l'application qui utilise des registres de configuration pour l'instanciation de la formule CNF [62, 63]. Afin de traiter les instances qui dépassent la capacité matérielle disponible, un schéma de matériel virtuel avec commutation de contexte a été proposé. Dans ce cas, les données de configuration du solveur SAT sont organisées en pages, qui sont stockées dans la mémoire embarquée. Chaque page configure un pipeline de clause, tel qu'illustré dans la figure 3.8. Les données correspondant aux variables sont conservées en deux blocs mémoire. Les variables sont lues séquentiellement à partir d'un bloc de mémoire, traitées dans la pipeline de la clause et écrites dans l'autre bloc de mémoire. La page de matériel suivante est alors chargée et les variables sont traitées lors du déplacement du deuxième bloc de mémoire vers le premier. Le processus se poursuit alors qu'il ya des implications générées et qu'aucun conflit n'est détecté.

Dans [63], on décrit une implémentation matérielle du solveur SAT basé sur la carte PCI RC1000 de Celoxica contenant un FPGA Xilinx XCV2000E et quatre banques SRAM (2 Mo chacun). Les résultats préliminaires montrent que ce système peut fonctionner à 47 MHz et devrait permettre de traiter des formules ayant jusqu'à 7 680 variables et 214 304 clauses. Cependant, l'interface avec le logiciel n'a pas encore été implémentée, de sorte que les temps d'exécution réels ne sont pas disponibles.

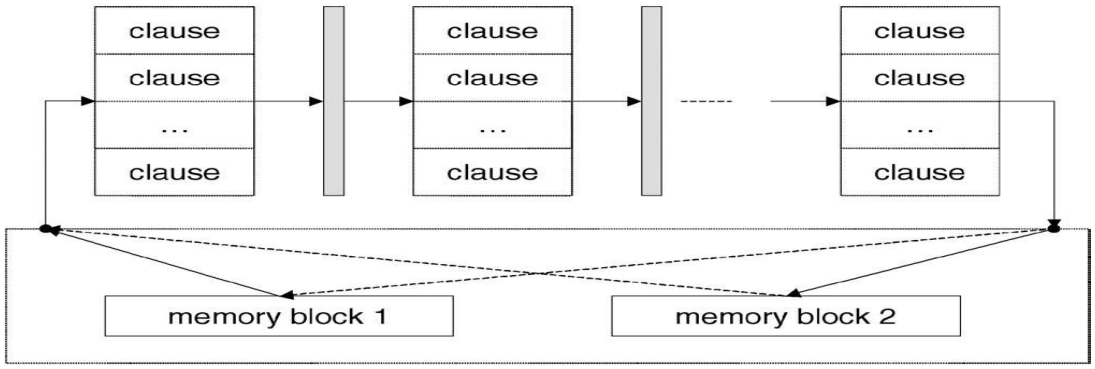

FIGURE 3.8 – Architecture de Soussa et al.

3.2.8 Skliarova and Ferrari

Skliarova et Ferrari [65] ont proposé et mis en œuvre un solveur SAT spécifique à l'application basé sur l'algorithme DP. Le problème a été formulé sur une matrice ternaire en établissant une correspondance entre des clauses et des variables d'une formule CNF et des lignes et des colonnes de la matrice et en trouvant un vecteur ternaire v qui est orthogonal à chaque ligne de la matrice construite. Si le vecteur v ne peut pas être trouvé, alors la formule est insatisfaisante. D'autre part, si le vecteur v existe, alors les zéros et les uns correspondent aux variables qui doivent recevoir respectivement les valeurs «1» et «0» pour satisfaire à la formule CNF.

L'architecture de satisfaction proposée est représentée dans la figure 3.9. La séquence des opérations matérielles est gérée par une unité de contrôle centrale utilisant une mémoire de pile pour supporter le processus de retour arrière. Il existe quatre blocs mémoire stockant la matrice initiale et sa transposition. Les matrices ne sont pas modifiées pendant le processus de recherche. Toutes les modifications possibles (telles que la suppression de lignes et de colonnes) sont reflétées dans les registres. L'ALU exécute différentes opérations sur des lignes et des colonnes de matrices telles que le comptage du nombre de uns, etc. Pour résoudre diverses instances de problème, il suffit de télécharger les données de matrices respectives. Tous les autres composants du solveur restent inchangés. Cela permet d'utiliser la reconfigurabilité locale et de réduire la surcharge de configuration.

Dans [29], une architecture SAT satisfier améliorée a été proposée la mise en œuvre d'un algorithme hybride. L'algorithme suggéré nécessite la construction d'une liste de favoris, qui ne contient que des attributions de variables partielles autorisées. Ensuite, l'algorithme DP est appliqué et, à chaque nœud de l'arbre de décision, l'assignation de la variable partielle actuelle est vérifiée pour la cohérence avec le contenu de la liste des favoris. Lorsqu'une incohérence est détectée, un retour arrière prématuré est effectué, ce qui permet de réduire le nombre de nœuds visités dans l'arbre de décision.

Le problème SAT est divisé entre le logiciel et le matériel reconfigurable de telle sorte qu'un FPGA est seulement responsable du traitement des sous-problèmes qui apparaissent à différents niveaux de l'arbre de décision et satisfont aux contraintes matérielles imposées

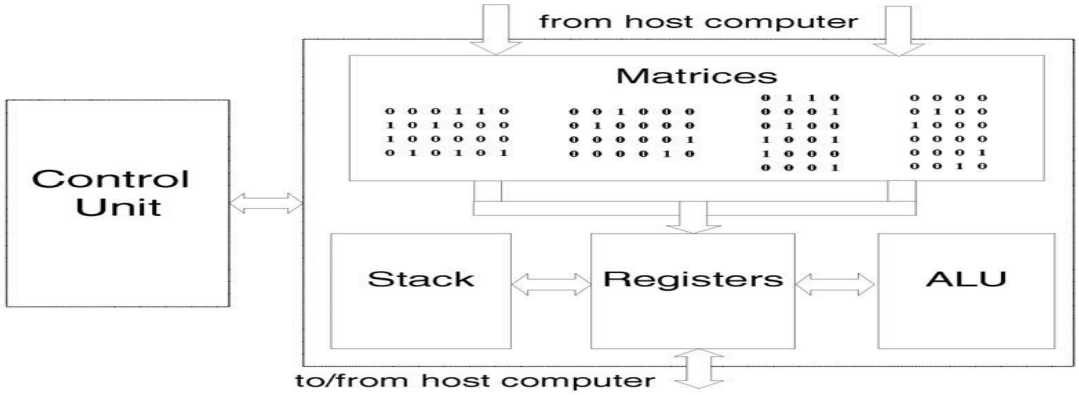

FIGURE 3.9 – Architecture de Skliarova et Ferrari.

(telles que le nombre maximum autorisé de lignes et de colonnes dans la matrice). Cette technique permet de traiter des formules qui dépassent les ressources du matériel reconfigurable disponible. Cependant, l'efficacité du partitionnement logiciel / matériel dépend fortement des caractéristiques d'une instance de problème particulière.

Le solveur SAT a été implémenté sur une carte ADM-XRC PCI d'Alpha Data contenant un XCV812E Virtex-EM FPGA (fonctionnant à 40MHz). Les accélérations obtenues pour les repères trou x de DIMACS SAT [27], y compris la configuration FPGA et le temps de communication matériel / logiciel, atteignent deux ordres de grandeur par rapport à GRASP [52] s'exécutant sur un AMD / Athlon / 1GHz / 256MB [72]. Pour d'autres benchmarks de la même suite [27], qui sont facilement résolus par GRASP (en fractions de seconde), le matériel SAT satisfier [65] [72] ne fournit pas d'accélérations utiles.

3.2.9 Limites des architectures

3.2.9.1 Tableau comparatif

Les différentes architectures proposées et présentées dans les sections précédentes utilisent toutes l'algorithme DP avec une sélection statique ou dynamique des instances. Ceci est bien détaillé dans le tableau 3.1. Ces architectures se basent sur une instance spécifique ce qui limite leur utilisations pour des traitements plus poussés. Cependant l'utilisation du hardware leur a permis de gagner dans le temps de résolution. Notre objectif alors est de booster ces temps de résolution, tout en s'orientant vers la voie d'une architecture pure matérielle, indépendante de l'instance.

3.2.9.2 Choix de notre approche pour le solveur

Le choix de la méthode à adopter pour notre solveur s'est basé sur la recherche arborescente comme c'était le cas de toutes les architectures proposées dans l'état de l'art. Les différentes architectures proposaient des algorithmes DP ou DPLL, utilisant des instances spécifiques avec

TABLE 3.1 – Caractéristiques principales des solveurs.

Solveur SAT	Issues algorithmique	Model de programmat-ion	Model d'execution	Model de reconfigurat-ion	Capacité logique	Performance (t_{total})
Suyama et al.	Comme Algorithme DP avec une sélection dynamique	instance-specifique	hardware seul	statique	système multi-FPGA	$t_{comp} + t_{conf} + t_{ex\_h}$
Zhong et al. [12]	Algorithme DP avec une sélection statique, non-chronologique, backtrack, analyse des conflits	instance-specifique	hardware seul	Dynamique (global)	système multi-FPGA	$t_{comp} + t_{conf} + t_{comm} + t_{ex\_h}$
Platzner et al.	Basé sur l'algorithme DP avec une sélection statique	instance-specifique	hardware seul	statique	use larger device	$t_{comp} + t_{conf} + t_{ex\_h}$
Abramovici et al. [15]	Basé sur l'algorithme DP avec une sélection d'optimisation statique	instance-specifique	hardware seul	Dynamique (global)	logic partitioning in sub-formulae	$t_{comp} + t_{conf} + t_{comm} + t_{ex\_h}$
Sousa et al.	DP-based algorithm with dynamic selec-tion and conflict analysis (in software)	application-specifique	software/hardware partitioning according to computational complexity	Dynamique (partiel)	virtual hardware scheme	$t_{conf} + t_{comm} + t_{ex\_s} + t_{ex\_h}$
Skliarova et al.	DP-based algorithm with dynamic selection	application-specifique	software/ hardware partitioning according to logic capacity	Dynamique (partiel)	software/ hardware parti-tioning	$t_{conf} + t_{comm} + t_{ex\_s} + t_{tex\_h}$

un modèle d'exécution hardware uniquement. Ce qu'on a rapporté par rapport aux différentes architectures c'est que notre architecture présentera un nouvel algorithme basé sur la recherche arborescente sur laquelle se sont basés les algorithmes DP et DPLL mais avec une utilisation différente de cette dernière qui nous permettra de gagner en performance totale en temps de chargement et résolution du problème. En plus de ceci notre solveur sera indépendant de

l'instance et non plus spécifique à l'instance donnée. Notre solveur adoptera également une conception purement matérielle orientée vers implémentation FPGA. Ce qui caractérisera aussi notre architecture c'est qu'elle sera localement synchrone et globalement asynchrone, car elle sera composée de plusieurs unités ou chacune fonctionnera de manière synchrone mais communiqueront et échangeront toutes entre elles de façon asynchrone (au sens des d'échanges de données parties et non des fréquences d'horloge).

3.3 Une architecture de solveur purement matérielle

3.3.1 Principes de base

Le choix de l'arbre de recherche et de l'algorithme de parcours dans cette arbre est déterminant dans la conception de l'architecture matérielle capable d'en prendre la charge.

Notre choix de représentation de l'arbre de recherche s'est porté vers la structure que nous avons présenté dans la figure 2.4 de cette section. Cet arbre est défini de manière statique par le choix d'ordonnancement des indices des variables. Il présente des propriétés intéressantes et utiles décrites dans la section 2.5.1.

La stratégie de parcours que nous avons choisie, et qui constitue en bonne partie l'originalité de notre architecture, est intermédiaire entre DPS (recherche en profondeur d'abord) et BPS (recherche en largeur d'abord). Par défaut, le mode privilégié est le DPS. Mais, chaque fois que ce mode est susceptible d'entraver la progression de la recherche (pénurie de données pour cause d'accès mémoire lents), la recherche bascule temporairement en mode BPS.

L'architecture proposée peut être paramétrée pour ne rechercher qu'une solution (la première trouvée) où l'intégralité des solutions. Du point de vue pratique il s'agit de pouvoir identifier dans le graphe, un ou tous les nœuds qui correspondent à une solution satisfaisant la formule CNF du problème SAT.

Pour les variables, nous avons choisi une représentation à trois états qui correspondent respectivement à 0 (« Faux »), 1 (« Vrai ») et U (« Non Assignée »). Dans la structure d'arbre choisie, à chaque nœud correspond une configuration unique de l'assignation des variables. À cette configuration, correspond donc un état de la formule CNF : « satisfaite » (la formule vaut 1, ce que l'on notera « SAT »), « non satisfaisable » (la formule vaut 0, ce que l'on notera « UNSAT ») et « indécidable » (aucune clause de la formule ne vaut 0 mais il en existe au moins une que l'on ne peut évaluer à 1 à cause des variables à l'état U, la formule étant alors évaluée à U). Cependant, l'état d'un nœud ne peut être connu qu'après qu'il a été évalué. Pour représenter cela, nous avons introduit un état additionnel « inconnu » (que l'on notera Z) pour signifier que l'état d'un nœud n'a pas encore été évalué.

Au commencement du parcours de recherche, tous les nœuds sont à l'état Z (« inconnu »). Au fur et à mesure des évaluations, des nœuds vont pouvoir basculer définitivement de cet

état vers l'un des trois autres états. Comme il avait été indiqué dans la section 2.5.1, l'état de certains nœuds peuvent être déterminés indirectement, sans être évalués, de la connaissance d'autres nœuds. Pour rappel, pour ce type d'arbre, tous les nœuds situés dans le sous-arbre gauche ou droit d'un nœud portant la valeur 0 ou 1 porte la même valeur. Lors de l'évaluation, pendant le parcours, d'un nœud portant la valeur 0, un backtrack est naturellement appliqué. Mais il est important de noter que, dans le cadre d'une recherche exhaustive de toutes les solutions, un backtrack peut également étre appliqué sur la valeur 1. Bien sûr, l'interprétation en est différente : dans le cas 0, les sous-arbres ne contiennent aucune solution, mais à l'opposé dans le cas 1, il ne contiennent que des solutions. En pratique donc, la descente d'un nœud vers ses sous-arbres n'a de sens que si ce nœud est à l'état U.

Dans notre approche, l'indépendance du solveur par rapport à l'instance SAT traitée est indispensable. Cela est possible en « matérialisant » les clauses sous forme de données et non de circuit. Le circuit doit pouvoir traiter n'importe quelle clause d'arité k (paramètre de l'architecture) indépendamment de la clause elle-même. De ce fait, les clauses sont représentées dans le solveur sous forme de donnée (à l'instar de ce qui se passe dans les solveurs logiciels).

L'architecture doit être capable d'évaluer en parallèle et en une seule étape un nombre important de clauses (paramètre de l'architecture). Mais, ce nombre est à priori dans la majorité des cas, bien inférieur au nombre de clauses du problème SAT à traiter. Ceci à une conséquence importante lors de l'évaluation de l'état d'un nœud : elle ne peut être faite en une seule étape. Plusieurs cycles sont nécessaires pour traiter les clauses, par groupes de clauses. Cette approche nécessite de pouvoir de disposer d'accès rapides à la mémoire pour changer de groupe de clauses. Alors que ceci est une opération qui peut s'avérer très lente, faisant perdre par les attentes induites, tous les bénéfices de l'évaluation parallèle des clauses.

Une première architecture que nous avons développée, présentée dans [66], a permis de mettre en évidence le problème de limitation sévère des performances que constitue la nécessité pour une architecture de solveur d'utiliser les données représentant les clauses en y accédant à partir d'une RAM. En effet, la largeur d'une RAM reste limitée à quelques dizaines de bits, voir une centaine de bits. Ceci limite sévèrement le débit de données auquel il est possible d'accéder, limitant d'autant la capacité de traitement d'une architecture.

C'est ici qu'intervient notre stratégie de parcours en BPS. Au lieu d'attendre de pouvoir disposer d'un nouveau groupe de clauses, on suspend l'évaluation du nœud courant pour commencer l'évaluation d'un autre nœud. Il est à priori évident que cela n'a pas de sens de sélectionner un nœud situé dans l'un des sous-arbres du nœud courant. De ce fait, le parcours est susceptible d'abandonner, tout du moins momentanément, la stratégie de parcours en DPS.

Notre proposition d'architecture découle de ces choix et d'éléments de réflexion.

3.3.2 Architecture du solveur

Notre solveur adopte une approche purement matérielle, orientée vers une implémentation sur FPGA. Son architecture est composée de trois parties principales, les unités SAT_CACHE, SAT_CORE et SAT_JOURNAL, selon l'organisation représentée dans la figure 3.10. Chacune des parties est synchrone. Si globalement, une seule horloge pilote l'ensemble des unités, les échanges de données entre unités s'effectuent selon des protocoles asynchrones.

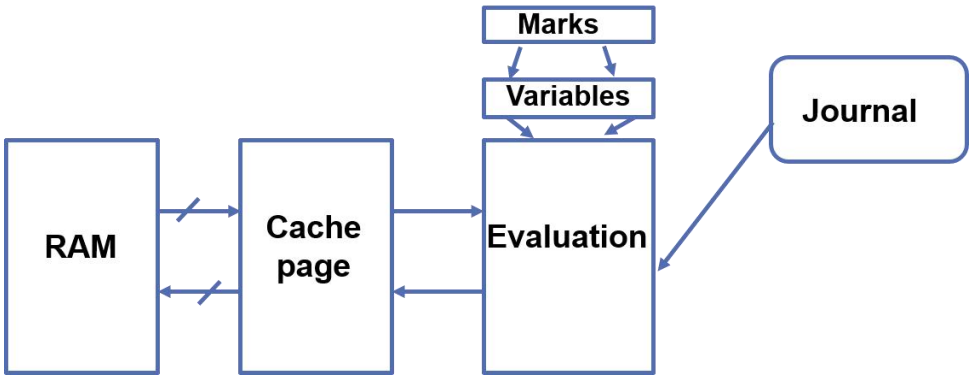

FIGURE 3.10 – Vue globale contextualisée des différentes unités du solveur SAT.

La RAM qui y figure ne fait pas partie de l'architecture. Elle contient initialement les données qui représentent les clauses sur lesquels le solveur doit travailler, celui-ci étant indépendant de l'instance traitée. Les clauses sont stockées sous forme compacte. Pour tout problème k-SAT, une clause est stockée sous forme de k paires (indice,coefficient). L'indice indique le numéro de la variable et le coefficient permet de connaître la forme du littéral, c'est-à-dire, comment la variable intervient dans la clause.

Comme indiqué précédemment, la récupération des données à partir d'une RAM est un processus lent et le traitement des clauses directement sur la RAM entraînerait une limitation trop importante des flux de données pouvant être traités simultanément, phénomène connu sous le nom de « famine de données ». Pour circonvenir ce problème, les données sont progressivement récupérées, séquentiellement, par « pages » de clauses, dans l'unité SAT_CACHE. Sa structure permet, lorsqu'une page de clauses est disponible, un transfert parallèle en un seul cycle vers l'unité suivante, SAT_CORE.

C'est l'unité SAT_CORE qui est proprement dit en charge l'algorithme de résolution. Il apparaît dans la figure 3.10 comme constitué des sous-blocs « évaluation », « variables » et « marques ». Son fonctionnement est décrit plus loin.

Le dernier bloc, SAT_JOURNAL, est l'élément permettant de suspendre et différer l'évaluation en cours d'un nœud au profit d'un autre. En effet, comme indiqué précédemment, toutes les clauses ne sont pas disponibles simultanément dans le solveur pour effectuer une évaluation complète de la valeur d'un nœud. S'il fallait attendre la disponibilité d'une nouvelle page de clauses pour continuer l'évaluation déjà entamée d'un nœud (avec les clauses disponibles),

l'architecture pourrait se trouver en « famine de donnée » entrainant une durée importante non productive. En suspendant l'évaluation d'un nœud pour entamer celle d'un autre nœud avec les clauses déjà disponibles, ont réduit d'autant les risques de famine. Une évaluation suspendue et différée est enregistrée dans SAT_JOURNAL. Elle n'en ressort qu'ultérieurement, lorsque l'évaluation peut reprendre.

Il est à noter que les risques de famine ne sont pas totalement éliminés. Ils dépendent principalement de deux facteurs : la taille du journal et la vitesse à laquelle il se remplit et se vide. Comme indiqué précédemment, seules les évaluations partielles en 1 nécessitent d'être poursuivies. En effet, une évaluation partielle en 0 n'a pas besoin d'être poursuivie (puisque le résultat ne pourra être différent de $Faux$). De même, une évaluation partielle en U n'a pas forcément besoin d'être poursuivie puisqu'elle ne peut plus aboutir à 1.

3.3.2.1 Unité SAT_CACHE

L'unité SAT_CACHE, représentée dans la figure 3.11, est en charge du préchargement d'une page de clauses, pour une utilisation future par l'unité SAT_CORE. Le préchargement est effectué séquentiellement dans un registre à décalage (structure de couleur mauve), ce qui évite toute structure coûteuse d'adressage. Un générateur circulaire d'adresses pour la RAM peut être utilisé, mais il est éventuellement plus simple de confier la génération des adresses à une partie logicielle externe (l'accès à la RAM n'étant alors pas direct). Les demandes d'accès peuvent alors être à des ressources externes, par exemple via un bus de données tel que USB. Le préchargement se poursuit tant que SAT_CACHE n'est pas plein.

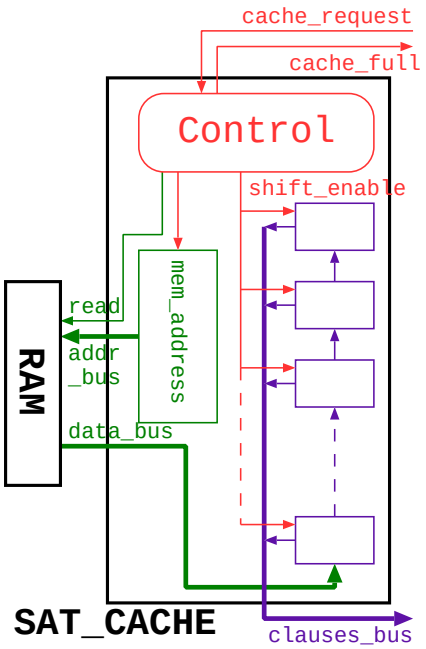

FIGURE 3.11 – Unité SAT_CACHE.

Le processus de chargement peut être plutôt long, surtout lorsque la taille de la page est grande, ce qui est souhaitable car cela favorise le parallélisme de traitement dans SAT_CORE.

Alors que le chargement de SAT_CACHE est séquentiel, l'envoi des données vers SAT_CORE s'effectue par un transfert parallèle en un seul cycle d'horloge. Le transfert de données entre les deux unités est contrôlé par les signaux $cache\_full$ et $cache\_req$. Comme les noms le laissent supposer, le premier signal indique que SAT_CACHE est plein, alors que le deuxième signale une requête de la part de SAT_CORE. Un transfert à lieu lorsque les deux signaux sont simultanément à 1. SAT_CACHE se retrouvant vide suite au transfert, le préchargement reprend alors naturellement.

Le comportement de ST_CORE étant assez trivial, la machine d'état qui pilote l'unité est simple et ne nécessite pas de précisions supplémentaires.

3.3.2.2 Unité SAT_CORE

L'unité SAT_CORE constitue le cœur de l'architecture du solveur. C'est elle qui est en charge à la fois du parcours de l'arbre de recherche et de l'évaluation de la valeur des nœuds parcourus. Sa structure est représentée dans la figure 3.12.

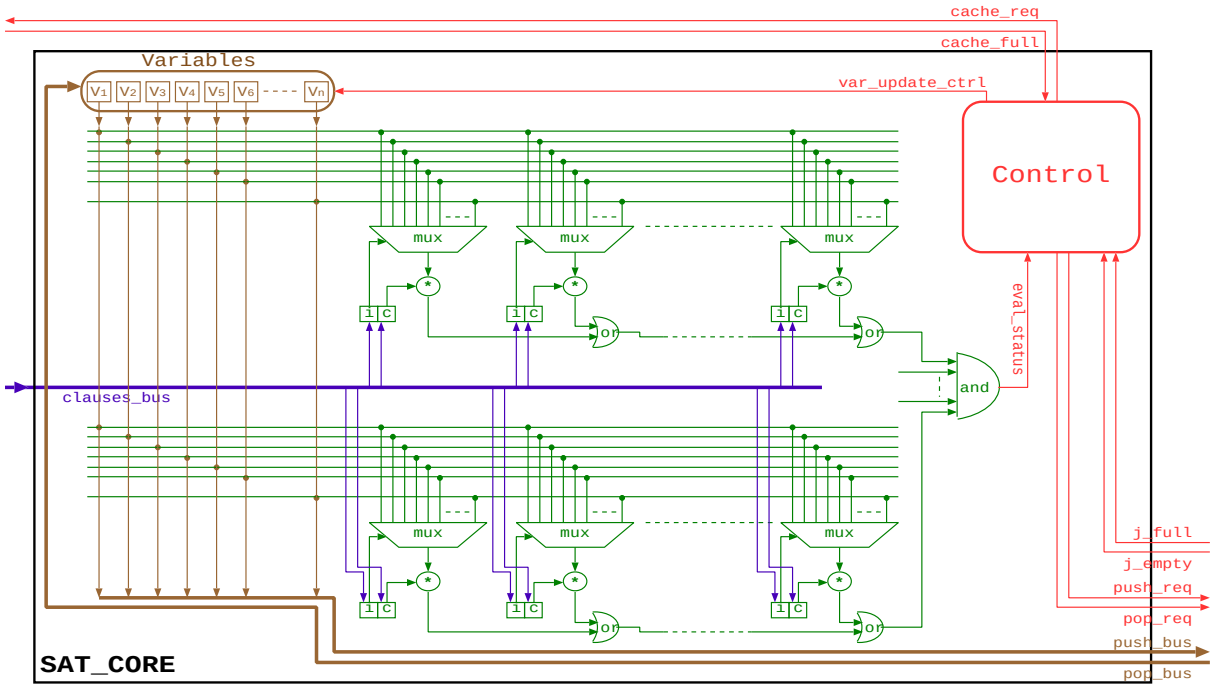

FIGURE 3.12 – Unité SAT_CORE.

Elle est constituée de plusieurs parties : le bloc « variables et marques » (représenté en marron dans la figure, sous la dénomination *variables*), le bloc « évaluation » (en vert) et le

bloc « contrôle » (en rouge).

Le bloc « variables et marques » est constitué d'un structure de registres contenant d'une part les valeurs assignées aux variables pour le nœud courant (nœud en cours d'évaluation), ainsi qu'un jeu de marques associées au variables permettant d'annoter le parcours de l'arbre ainsi de faciliter la détermination du prochain nœud à atteindre. Les valeurs de variables sont accessibles en parallèle par le bloc « évaluation ».

Le bloc « évaluation » est constitué d'un ensemble de registres contenant les valeurs des clauses de la page actives (clauses de la page présente dans SAT_CORE et disponibles évaluation) ainsi que les structures de calcul permettant d'évaluer en un seul cycle d'horloge, la valeur que la page active de clauses prend pour la valeur actuelle des variables. Cette valeur est utilisée par le bloc contrôle dans la décision de l'action suivante pour le parcours de l'arbre.

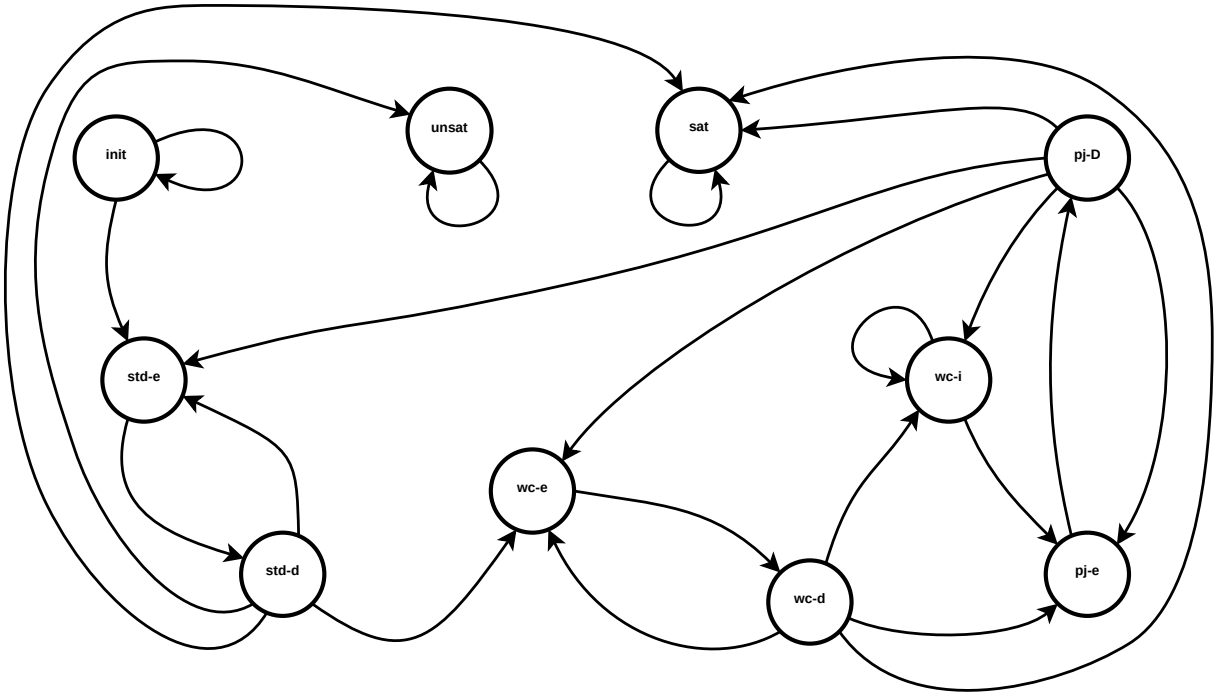

FIGURE 3.13 – Graphe de contrôle de la machine d'états finie de l'unité SAT_CORE.

Le bloc « contrôle » est constitué d'une machine d'états finie. Son graphe de contrôle est représenté de manière simplifiée dans la figure 3.13 (les conditions de transition entre états ont été omises sur les arcs par soucis de lisibilité). Il correspond à un fonctionnement du solveur où le parcours se limite à la recherche d'une solution. Le graphe correspondant à la recherche exhaustive de toutes les solutions est quasiment le même, l'état *sat* n'étant simplement plus un état final. Il s'agit d'un graphe synchrone : à chaque cycle d'horloge, un arc doit être franchi.

La résolution d'un nouveau problème débute à l'état *init*. C'est état est un état indispensable d'attente : il s'agit d'attendre que le chargement de la page de clauses se termine au niveau de

SAT_CACHE pour pouvoir la récupérer en tant que page active. Lorsque c'est enfin le cas, le traitement proprement dit peut enfin commencer avec une transition vers l'état *std-e*.

L'évaluation et parcours de l'arbre de recherche est matérialisé dans le graphe de contrôle à la base par des paires d'états *évaluer* et *décider*, ce qui requiert deux cycles. Ces paires d'états peuvent être regroupées en trois groupes.

Le premier groupe, constitué des états *std-e* et *std-d* correspond à une situation où l'évaluation et le parcours du graphe peut être poursuivi sans nécessiter une page de clauses autre que la page active (celle actuellement présente dans SAT_CORE). L'évaluation du nœud courant est faite pendant l'état *std-e* et la décision concernant la poursuite du parcours pendant l'état *std-d*. Aucune nouvelle page n'étant nécessaire pour un backtrack ou pour une descente dans l'arbre, dans ce cas on peut continuer à alterner entre ces deux états. Par contre, une évaluation à 1 de la page actuelle, lorsqu'il ne s'agit pas de la dernière page restant à évaluer, requiert le chargement d'une nouvelle page. Si c'est possible (SAT_CACHE étant pleine et un transfert pouvant avoir lieu entre SAT_CACHE et SAT_CORE), l'alternance enre *std-e* et *std-d* se poursuit. Dans le cas contraire, l'état actuel des variables doit être mis dans le journal de SAT_JOURNAL (pour une poursuite d'évaluation différée) et une transition vers la paire d'état *wc-e* et *wc-d* a lieu.

Le groupe d'états *wc-e* et *wc-d* correspond à une situation où le processus d'évaluation et le parcours du graphe se poursuivent (sur la page active de clause) alors que l'on est en attente d'une nouvelle page de clauses. Il est cependant important de noter que toutes les combinaisons de variables qui se trouvent dans le journal (c'est à dire, les nœuds dont la poursuite d'évaluation est différée) ont déjà été évaluées avec la page active.

Lorsque une nouvelle page de clauses devient disponible, il est nécessaire temps de reprendre l'évaluation des nœuds en attente dans le journal. La paire *pj-e* et *pj-d* correspond au fonctionnement impératif tant qu'il reste des nœud dans SAT_JOURNAL à évaluer sur la page active.

Le nœud *wc-i* correspond à une situation d'attente imposée par la non disponibilité de place dans SAT_JOURNAL conjointement à la non disponibilité d'une nouvelle page de clauses.

Les état de décision *std-d*, *wc-d* et *pj-d* peuvent tous trois conduire vers l'un des états *SAT* et *UNSAT*, correspondant respectivement à la découverte d'une solution, ou la constatation de l'absence de solution (backtrack remontant jusqu'à la racine de l'arbre).

3.3.2.3 Unité SAT_JOURNAL

L'unité SAT_JOURNAL est représentée dans la figure 3.14. Elle est constituée de deux séries de registres à décalage de type FIFO (*First In First Out*) et d'un bloc de contrôle les pilotant.

La série de registres (*push FIFO buffer*) est utilisée pour recevoir les combinaisons de variables des nœuds dont l'évaluation est suspendue/différée et qui ont déjà été évalués sur la

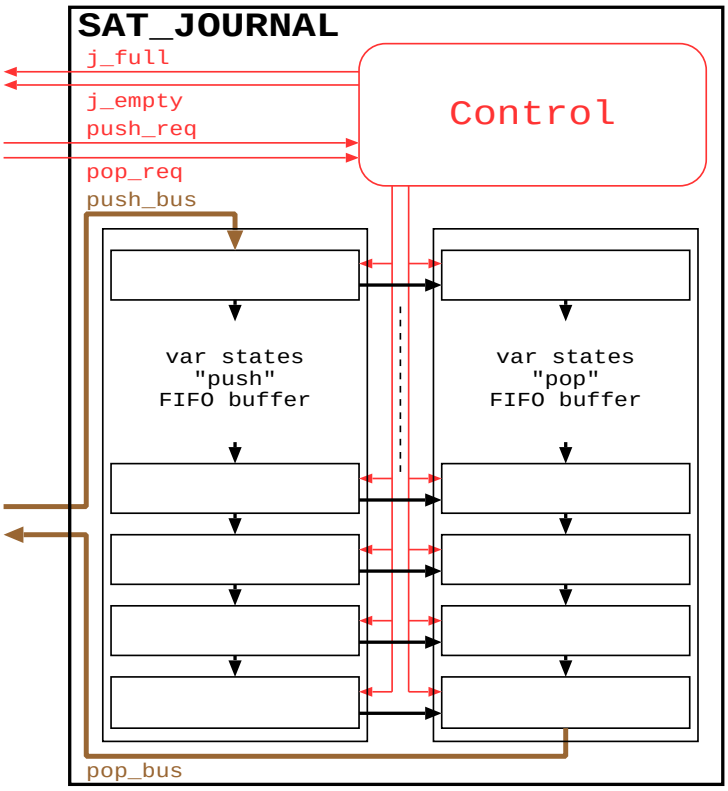

FIGURE 3.14 – Unité SAT_JOURNAL.

page active. La série de registres (*pop FIFO buffer*) contient quant à elle, les combinaisons de variables des nœuds dont l'évaluation est suspendue/différée et qui doivent encore être évalué sur la page active. Les nœuds présents dans le *push FIFO buffer* ne peuvent être transféré dans le emphpop FIFO buffer qu'un fois ce dernier vide et qu'une nouvelle page de clause et disponible et devient active.

Le contrôleur de SAT_JOURNAL est trivial et ne fait servir le fonctionnement de SAT_CORE. L'asservissement est assuré par l'intermédiaire des signaux *push_req* et *pop_req* qui explicitent les requêtes de SAT_CORE. Ce dernier a connaissance de l'état de SAT_JOURNAL par l'intermédiaire de *j_full* et *j_empty*.

3.3.3 Résultat d'implémentation

L'architecture de notre solveur SAT a été modélisée en VHDL au niveau RTL (*Register Transfer Level*). Il est configurable (paramètres génériques) à la synthèse par rapport aux paramètres suivants : v le nombre maximal de variables du problème SAT, k l'arité du problème ou taille maximale d'une clause (nombre maximal de littéraux dans une clause), p la taille d'une page de clause (nombre de clauses dans une page) et j la profondeur du journal.

D'autres paramètres sont également configurables mais ont peu d'impact sur l'utilisation

des ressource et la fréquence d'horloge maximale (comme la largeur du bus d'accès à la mémoire RAM). Parmi les paramètres de conception, seul v limite la taille du problème pouvant être traité, les autres paramètres n'ayant un impact que sur la vitesse de résolution et coût en ressource matérielles de l'architecture.

Les résultats du tableaux 3.2 ont été obtenus pour une implémentation sur la carte FPGA NexysVideo de chez Digilent (carte basée sur un FPGA Xilinx de la famille Artix-7) en utilisant la suite de conception Xilinx Vivado 15.4 pour la synthèse, le placement et le routage. Différentes combinaisons des paramètres génériques de conception (v, p, j) ont été testées, pour k fixé à 3, les performances étant mesurées en termes de ressources utilisées (ou consommées) et de vitesse de fonctionnement. La consommation de ressources est représentée par la paire (l, r) se rapportant respectivement aux nombres d'éléments logiques et de registres utilisés. La vitesse de fonctionnement est évalué par l'intermédiaire de la fréquence maximum de fonctionnement de l'horloge f_{clk}. Rappelons que deux cycles d'horloge sont requis au moins pour traiter un nœud sur la page de clauses active (traitement par une paire d'état *évaluer* et *décider*). Sur cette base, une colonne additionnelle cl/s indique le nombre maximal de clauses traitées par seconde (hors cycles d'horloge inactifs correspondant à l'état *wc-i*).

Avec l'accroissement des valeurs des paramètres de conception, le principal contributeur à la consommation de ressources logiques est, comme on pouvait s'y attendre, l'unité SAT_CORE. L'accroissement de la consommation de ressources est approximativement linéaire vis-à-vis de v ou de j. Cependant, lorsque ces paramètres atteignent des valeurs élevés, la tâche devient beaucoup plus difficile pour Vivado (la durée de synthèse passe de quelques minutes à plus de deux heures). Il est alors notable que pour ces cas, la consommation augmente aussi beaucoup plus rapidement. Une raison probable est la disponibilité locale limitée dans chaque partie du FPGA de lien d'interconnexion, nécessitant une duplication de la logique et des ressources de type registre. La dernière ligne du tableau correspond à une durée de synthèse de plus de 5 heures, synthèse qui s'est finalement interrompue par manque de mémoire (16 Go sur l'ordinateur utilisé).

Au stade actuel, la préparation et chargement des données représentant un problème SAT (les clauses du problème) est réalisé à la main, ce qui limite fortement les possibilités de test. Un développement d'interfaces logicielles de pilotages est actuellement en cours pour faciliter l'utilisation du solveur sur des problèmes de taille significative.

3.3.4 Pistes d'évolution

Dans cette architecture, au niveau de SAT_CORE, l'évaluation du nœud courant (évaluation de la configuration courante des variables) sur la page de clauses active est faite en un seul cycle d'horloge (cycle d'évaluation). Ceci est possible car toutes les variables sont accessibles simultanément, mais contribue largement au fait que la consommation des ressources logiques se situe essentiellement dans SAT_CORE. Une piste pour réduire la consommation de ressource consisterait à diviser les variables en sous-ensemble adjacents et à analyser pendant une cycle d'horloge, uniquement un sous-ensemble de variables. Un conséquence immédiate est que plus

TABLE 3.2 – Résultat d'implémentation pour le solver SAT.

Design parameters			Ressource usage		f_{clk}	cl/s
v	p	j	l	r	(MHz)	($\times 10^6$)
15	15	4	360	655	188.6	1414.5
15	15	8	483	751	188.6	1414.5
15	15	16	727	943	185.1	1388.3
15	15	32	1220	1327	185.1	1388.3
15	31	4	567	1240	172.4	2672.2
15	31	8	704	1344	172.4	2672.2
15	31	16	965	1552	172.4	2672.2
15	31	32	1491	1968	172.4	2672.2
20	20	4	448	995	166.6	1666.0
20	20	8	586	1051	166.6	1666.0
20	20	16	829	1243	166.6	1666.0
20	20	32	1319	1627	166.6	1666.0
22	31	4	623	1426	163.9	2540.5
22	31	8	764	1530	156.2	2421.1
22	31	16	1012	1738	156.2	2421.1
22	31	32	1567	2154	156.2	2421.1
31	31	4	626	1426	158.7	2459.8
31	31	8	742	1530	158.7	2459.8
31	31	16	943	1738	158.7	2459.8
31	31	32	1536	2154	156.2	2421.1
31	63	4	1008	2783	138.8	4372.2
31	63	8	1136	2896	136.9	4312.4
31	63	16	1354	3120	136.9	4312.4
31	63	32	2015	3567	128.2	4038.3
63	63	4	1154	3159	135.1	4255.6
63	63	8	1323	3269	111.1	3499.6
63	63	16	1599	3495	111.1	3499.6
63	63	32	22621	16011	68.0	2142.0
63	127	4	2248	6246	100.0	6350.0
63	127	8	2473	6371	85.4	5422.9
63	127	16	19132	10710	65.3	4146.6
63	127	32	31306	19379	45.4	2882.9
127	127	4	2349	7000	67.1	4260.8
127	127	8	7120	7391	63.6	4038.6
127	127	16	43907	20468	36.1	2292.3
127	127	32	Synthesis aborted			

d'un cycle est alors requis pour évaluer l'ensemble des variables (c'est-à-dire, l'état du nœud courant). De plus, le graphe de contrôle, et donc, l'unité de contrôle de SAT_CORE devient plus complexe.

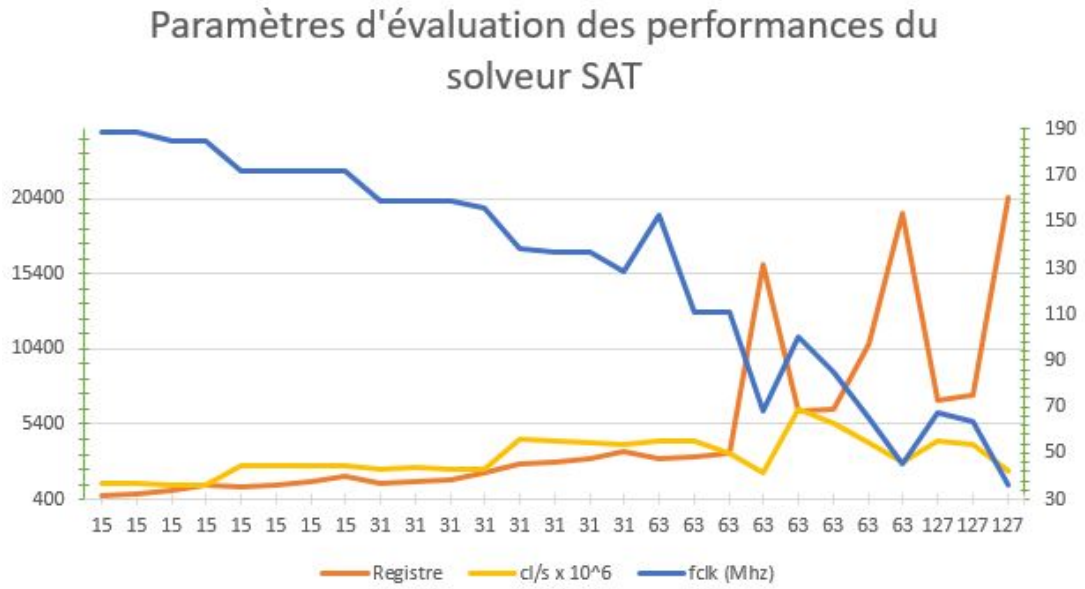

FIGURE 3.15 – Performances du solveurs SAT

Il est à noter que la préparation des clauses, et plus exactement, le partitionnement des clauses en pages, est d'une importance très élevé pour les performances du solveur. En effet, des pages contenant beaucoup de clauses « contradictoires », c'est-à-dire, introduisant beaucoup de contraintes entre elles, entraineront plus souvent des évaluations de nœuds à 0 sur la page active, ce qui, en induisant plus de backtrack, réduit la profondeur de l'arbre ayant besoin d'être exploré. Il serait intéressant d'étudier des stratégies de tri et regroupement de clauses selon un tel critère. Compte tenu du fonctionnement de notre solveur, cette opération préalable serait une stratégie statique, puisque réalisée avant d'entamer le parcours de l'arbre.

Une autre idée dont l'exploration serait intéressante est la décomposition d'un problème SAT en sous-problèmes SAT disjoint plus petits (en fixant les valeurs de quelques variables « pivots », permettant alors la résolution indépendante de chacun d'entre eux. Non seulement, ces sous-problèmes seraient à priori plus rapides à résoudre car moins grands et complexes, mais de plus, il serait possible d'en résoudre plusieurs simultanément avec plusieurs instances de notre solveur SAT.

3.4 Conclusion

Ce dernier chapitre était consacré aux architectures de solveurs SAT bénéficiant d'une accélération matérielle. Nous l'avons introduit par un état de l'art sur les approches architecturales hybrides (logiciel/matériel) ou matérielles, ciblant de plateformes FPGA ou multi FPGA, et basée pour l'essentiel sur les algorithmes DP et DPLL. Une étude comparative, résumée dans un tableau bilan, nous a conforté dans la volonté de concevoir un solveur dédié, purement

matériel et indépendant de l'instance traitée, ciblant une technologie FPGA et capable de fournir une capacité de traitement parallèle importante.

Dans la suite du chapitre, nous avons exposé les motivations et principes originaux de notre solveur qui présentaient l'étape importante de ces travaux de recherche dans la résolution des problèmes NP complet, reposant sur un mécanisme d'évaluation différée induisant une stratégies de parcours hybride (DFS et BFS) de de l'arbre de recherche. Nous avons ensuite détaillé l'architecture, composée des trois unités SAT_CACHE, SAT_CORE et SAT_JOURNAL, ainsi que le fonctionnement de celles-ci.

Les performances ont été évaluées et analysées par implémentation sur la carte NexyVideo de chez Digilent, carte basée sur un FPGA Xilinx de la famille Artix-7.

Conclusion générale et perspectives

Dans cette thèse, nous nous sommes intéressés à résolution, à l'aide de solveurs matériels dédiés, de problèmes issus du domaine industriel, dans le cadre de la gestion des ressources de production. Notre objectif étant de réduire significativement les temps de résolution et de permettre par conséquent, la prise en charge dans des laps de temps réduits ou tout du moins acceptables, de problèmes de taille significative issus du monde réel. Pour cela, nous avons choisi de proposer et développer une architecture de solveur matérielle dédiée ciblant des technologies de type FPGA.

Le formalisme choisi pour la modélisation mathématique, puis la résolution, est celui de la satisfiabilité propositionnelle, connu sous le nom de SAT (*SATisfiability*). Les problèmes SAT sont des problèmes fondamentaux de décision, de la classe des problèmes NP complet, liés à la théorie de la complexité et de la satisfiabilité booléenne. On peut en retrouver des exemples dans des domaines divers, tant académiques qu'issus du monde réel, tels que la recherche opérationnelle ou la cryptographie.

Nous avons tout d'abord abordé dans le chapitre 1 le domaine applicatif industriel, en situant clairement son contexte et l'objectif ciblé, à savoir, le développement et amélioration de la résolution de ces problèmes. Ceci a été réalisé par la présentation d'un état de l'art portant sur les concepts de risques, de dangers, de management des risques industriels, de gestion des ressources de production en s'appuyant sur les différentes normes édictées dans ce domaine. Afin de lever toutes ambiguïtés relatives aux problèmes industriels, des terminologies utilisées dans ces domaines ont été présentées en détails. Par la suite, après la présentation des différentes définitions relatives aux aspects industriels traités dans cette thèse, nous avons défini un problème industriel lié à la gestion des ressources de production tenant compte de divers risques industriels, contrairement à ce qui est habituellement proposé dans la littérature.

Pour ce problème, nous avons proposé une méthode de modélisation basée sur le formalisme ILP (*Integar Linear Programming*) ou programmation linéaire en nombres entiers. Ce formalisme est utilisé en pratique comme passerelle vers le formalisme SAT, plus complexe à gérer intellectuellement, et traité dans le chapitre 2

Le chapitre 2 a été consacré à SAT, problème central en théorie de la complexité et représentant phare des problèmes NP-complets. Après l'introduction des notions préliminaires de la logique et du langage propositionnel utilisé par ce formalisme, SAT adoptant une forme normale conjonctive (CNF), nous avons poursuivi en présentant les différents paradigmes

aidant à résoudre les problèmes NP-complet de type SAT. Ces paradigmes basés sur la recherche arborescente, qui se divisent en deux familles de méthodes : la recherche en profondeur d'abord (DFS, *Deepth First Search)* et la recherche en largeur de d'abord (BFS, *Breath First Search*). Ces deux stratégies de parcours de recherche se déclinent sur les méthodes de résolutions exactes et énumératives (complètes et incomplètes). Des algorithmes de résolution et heuristiques fondamentaux, ont été présentés, notamment DPLL et CDCL.

Le chapitre 3, consacré aux solveurs bénéficiant d'une accélération matérielle, a introduit dans un premier temps, un état de l'art consacré aux architectures matérielles et hybrides (logicielles/matérielle) visant à accélérer la résolution des problèmes SAT. Cette première partie se terminant par un bilan comparatif. La suite du chapitre a été consacrée à notre contribution à la résolution du problème SAT, ce qui constitue le cœur de notre travail.

Notre contribution, centrée sur le développement d'un solveur matériel dédié, propose une méthode originale, différente de celles employées par les solveurs logiciels ou hybrides. Sachant que l'accélération matérielle peut fournir une puissance de calcul parallèle massive, les approches hybrides restent généralement limitées par leur faible capacité d'accès aux données (bande passante de mémoire et latence de communication entre logiciel et matériel), les périodes d'inactivité induites par ces faibles débits de données étant en général nombreuses et longues. L'originalité de notre approche repose sur la capacité à réduire l'apparition de périodes d'inactivité liées à ce que l'on peut appeler « famine de données », en utilisant des techniques permettant de suspendre et reporter l'évaluation de nœud du parcours de recherche pour lesquels tous les données ne sont pas encore disponibles. Ce faisant, le solveur adopte une stratégie de parcours hybride entre DFS et BFS qui lui permet de tirer la pleine puissance de ses capacités de traitement massivement parallèle.

Destiné à une mise en œuvre sur FPGA, le solveur est constitué de trois unités spécialisées, SAT_CACHE, SAT_CORE et SAT_JOURNAL qui coopèrent entre elles via des échanges asynchrones (l'architecture est cependant synchrone au sens ou une seule horloge pilote l'ensemble). Le solveur est indépendant de l'instance SAT à traiter et est totalement configurable avant la synthèse en ce qui concerne les paramètres relatifs à l'arité du problème SAT, au niveau de parallélisme de calcul (nombre de clauses traitées simultanément), au nombre de variables supportés, à la taille des tampons de journalisation (dédiées au stockage temporaire des nœuds dont l'évaluation a été suspendue et reportée).

Les performances obtenues ont été évaluées par synthèse sur la carte NexysVideo de Digilent (basée sur un Xilinx FPGA de la famille Artix-7), en utilisant la suite de conception Xilinx Vivado 15.4. Pour explorer les performances (consommation de ressources et vitesse de traitement) de l'architecture, les tests ont été réalisés en variant, à la synthèse, les valeurs des plusieurs paramètres génériques de l'architecture. L'évolution de ces résultats lorsque l'on atteint les limites du FPGA ciblé et de l'outil de synthèse ont été analysés et des propositions suggérées pour circonvenir ces limites.

Les perspectives à court terme dans la poursuite du développement du solveur SAT se situent à différents niveaux :

— développement d'une plateforme logicielle permettant de doter le solveur d'un interface homme/machine un tant soit peu ergonomique afin d'en faciliter l'usage pratique ;

— développement des algorithmes de prétraitement des clauses permettant de les regrouper en pages de clauses « contradictoires », c'est-à-dire, où les contraintes entre clauses d'un même page favorise sur chaque page l'évaluation des nœuds à 0 ;

— découpage d'un problème SAT en plusieurs sous-problèmes SAT disjoints (par voie logicielle) et résolution indépendante de ces sous-problèmes (voire parallèle sur des instances multiples du solveur).

À plus long terme, l'exploration d'autres alternatives architecturales pour les trois unités SAT_CACHE, SAT_CORE et SAT_JOURNAL, voire même, d'autres organisations internets sont à envisager. L'algorithme de parcours implémenté dans le solveur, bien qu'utilisant une stratégie astucieuse de parcours de l'arbre par suspension/report d'évaluation de nœud, peut être largement améliorer en y intégrant des heuristiques nouvelles. En effet, l'application telles quelles des heuristiques traditionnelles de l'approche logicielle est peu approprié, ces heuristiques étant difficilement parallélisables sans risque d'encourir une « famine de données ».

[]

Bibliographie

[1] Yves MORTUREUX. *Analyse préliminaire de risques*. Editions TI, Techniques de l'Ingénieur, 2002.

[2] A. Heurtel. La gestion des risques techniques (sûreté de fonctionnement) et des risques de management. *CNRS IN2P3/LAL Version*, 2(2.1) :6, 2003.

[3] Mohamed Sallak, Christophe Simon, and Jean-François Aubry. Optimal design of safety instrumented systems : A graph reliability approach. In *7ème Congrès International Pluridisciplinaire Qualité et Sûreté de Fonctionnement, Qualita 2007*, pages 255–262. RUFEREQ, 2007.

[4] Jean-Claude Laprie. Sûreté de fonctionnement informatique : concepts, défis, directions. *Computing*, 1(1) :11–33, 2004.

[5] R Flanagan and G Norman. Risk management and construction blackwell science ltd. 1993.

[6] Souheil Ayed, Sofiène Dellagi, and Nidhal Rezg. Optimal integrated maintenance production strategy with variable production rate for random demand and subcontracting constraint. *IFAC Proceedings Volumes*, 44(1) :5207–5212, 2011.

[7] Sofiene Dellagi, Anis Chelbi, and Wajdi Trabelsi. Joint integrated production-maintenance policy with production plan smoothing through production rate control. *Journal of manufacturing systems*, 42 :262–270, 2017.

[8] Lahcen Mifdal, Zied Hajej, and Sofiene Dellagi. Joint optimization approach of maintenance and production planning for a multiple-product manufacturing system. *Mathematical Problems in Engineering*, 2015, 2015.

[9] Samir Haddad. *Modèle d'évaluation des performances d'une ligne de production d'une séquence de N produits distincts sur M machines avec (M-1) stocks tampons*. PhD thesis, Université Laval, 2012.

[10] Russell Impagliazzo, Shachar Lovett, Ramamohan Paturi, and Stefan Schneider. 0-1 integer linear programming with a linear number of constraints. *arXiv preprint arXiv :1401.5512*, 2014.

[11] Oumar Koné, Christian Artigues, Pierre Lopez, and Marcel Mongeau. Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources. *Flexible Services and Manufacturing Journal*, 25(1-2) :25–47, 2013.

[12] George Dantzig. *Linear programming and extensions*. Princeton university press, 2016.

[13] Ignasi Abío Roig. Solving hard industrial combinatorial problems with sat. 2013.

[14] António G Ramos and José Leal. Ilp model for energy-efficient production scheduling of flake ice units in food retail stores. *Journal of cleaner production*, 156 :953–961, 2017.

[15] Khadija Bousmar, Fabrice Monteiro, Sofiene Dellagi, Zineb Habbas, and Abbas Dandache. Modelling industrial manufacturing problem using ilp solver : Case of production analysis. In *Microelectronics (ICM), 2017 29th International Conference on*, pages 1–4. IEEE, 2017.

[16] Khadija Bousmar, Fabrice Monteiro, Zineb Habbas, Sofiene Dellagi, and Abbas Dandache. A pure hardware k-sat solver architecture using fpga for developing decision-support tools for industrial problems. In *6ème conférence international sur l'Innovation et Nouvelles Tendances dans les Systèmes d'Information INTIS 24-25 Novembre 2017, Casablanca, Maroc.*, 2017.

[17] Stephen Cook. The p versus np problem. *The millennium prize problems*, pages 87–104, 2006.

[18] Celina MH De Figueiredo. The p versus np–complete dichotomy of some challenging problems in graph theory. *Discrete Applied Mathematics*, 160(18) :2681–2693, 2012.

[19] Frédéric Lardeux. *Approches hybrides pour les problèmes de satisfiabilité (SAT et MAX-SAT)*. PhD thesis, Thèse de doctorat, 2005.

[20] Fadi A Aloul, Arathi Ramani, Igor L Markov, and Karem A Sakallah. Solving difficult sat instances in the presence of symmetry. In *Proceedings of the 39th annual Design Automation Conference*, pages 731–736. ACM, 2002.

[21] Luca FERRO, Samith KHIN, and Nader SALMAN. Résolution pratique de problèmes np-complets. Technical report, Technical report, 26 juin, 2005.

[22] Xili Wang. A novel approach of solving the cnf-sat problem. *arXiv preprint arXiv :1307.6291*, 2013.

[23] Olivier Fourdrinoy. Hybridation des méthodes de résolution pour sat. *7e Rencontres Jeunes Chercheurs en Intelligence Artificielle*, 2005.

[24] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7) :394–397, 1962.

[25] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2) :277–284, 1987.

[26] Ann Becker and Dan Geiger. A sufficiently fast algorithm for finding close to optimal clique trees. *Artificial Intelligence*, 125(1-2) :3–17, 2001.

[27] Iouliia Skliarova and Antonio de Brito Ferrari. Reconfigurable hardware sat solvers : A survey of systems. *IEEE Transactions on Computers*, 53(11) :1449–1461, 2004.

[28] Lintao Zhang and Sharad Malik. The quest for efficient boolean satisfiability solvers. In *International Conference on Computer Aided Verification*, pages 17–36. Springer, 2002.

[29] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3) :201–215, 1960.

[30] Chu Min Li and Anbulagan Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the 15th international joint conference on Artifical intelligence-Volume 1*, pages 366–371. Morgan Kaufmann Publishers Inc., 1997.

[31] Donald Loveland. Automatic theorem proving, 1978.

[32] Richard E Korf. Depth-first iterative-deepening : An optimal admissible tree search. *Artificial intelligence*, 27(1) :97–109, 1985.

[33] JP Marques Silva and Karem A Sakallah. Conflict analysis in search algorithms for satisfiability. In *Tools with Artificial Intelligence, 1996., Proceedings Eighth IEEE International Conference on*, pages 467–469. IEEE, 1996.

[34] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff : Engineering an efficient sat solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

[35] Narendhar Maaroju and Deepak Guide Garg. *Choosing the best heuristic for a NP-Problem*. PhD thesis, 2009.

[36] Lonlac Konlac and Jerry Garvin. *Contributions à la résolution du problème de la Satisfiabilité Propositionnelle*. PhD thesis, Artois, 2014.

[37] Carla P Gomes, Bart Selman, Henry Kautz, et al. Boosting combinatorial search through randomization. *AAAI/IAAI*, 98 :431–437, 1998.

[38] Lintao Zhang, Conor F Madigan, Matthew H Moskewicz, and Sharad Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 279–285. IEEE Press, 2001.

[39] Carla P Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability solvers. *Foundations of Artificial Intelligence*, 3 :89–134, 2008.

[40] Pascal Vander-Swalmen. *Aspects parallèles des problèmes de satisfaisabilité*. PhD thesis, Université de Reims-Champagne Ardenne, 2009.

[41] James M Crawford and Larry D Auton. Experimental results on the crossover point in random 3-sat. *Artificial intelligence*, 81(1-2) :31–57, 1996.

[42] Robert G Jeroslow and Jinchang Wang. Solving propositional satisfiability problems. *Annals of mathematics and Artificial Intelligence*, 1(1-4) :167–187, 1990.

[43] Makoto Yokoo, Takayuki Suyama, and Hiroshi Sawada. Solving satisfiability problems using field programmable gate arrays : First results. In *International Conference on Principles and Practice of Constraint Programming*, pages 497–509. Springer, 1996.

[44] Takayuki Suyama, Makoto Yokoo, and Hiroshi Sawada. Solving satisfiability problems on fpgas. In *International Workshop on Field Programmable Logic and Applications*, pages 136–145. Springer, 1996.

[45] Takayuki Suyama, Makoto Yokoo, and Hiroshi Sawada. Solving satisfiability problems using logic synthesis and reconfigurable hardware. In *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*, volume 7, pages 179–186. IEEE, 1998.

[46] Takayuki Suyama, Makoto Yokoo, Hiroshi Sawada, and Akira Nagoya. Solving satisfiability problems using reconfigurable computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1) :109–116, 2001.

[47] Takayuki Suyama, Makoto Yokoo, and Akira Nagoya. Solving satisfiability problems on fpgas using experimental unit propagation. In *International Conference on Principles and Practice of Constraint Programming*, pages 434–445. Springer, 1999.

[48] Jon William Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, University of Pennsylvania Philadelphia, PA, 1995.

[49] Peixin Zhong, Margaret Martonosi, Pranav Ashar, and Sharad Malik. Using configurable computing to accelerate boolean satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(6) :861–868, 1999.

[50] Peixin Zhong, Pranav Ashar, Sharad Malik, and Margaret Martonosi. Using reconfigurable computing techniques to accelerate problems in the cad domain : a case study with boolean satisfiability. In *Proceedings of the 35th annual Design Automation Conference*, pages 194–199. ACM, 1998.

[51] Peixin Zhong, Margaret Martonosi, Pranav Ashar, and Sharad Malik. Solving boolean satisfiability with dynamic hardware configurations. In *International Workshop on Field Programmable Logic and Applications*, pages 326–335. Springer, 1998.

[52] João P Marques-Silva and Karem A Sakallah. Grasp : A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5) :506–521, 1999.

[53] Oskar Mencer and M Plazner. Dynamic circuit generation for boolean satisfiability in an object-oriented design environment. In *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*, pages 8–pp. IEEE, 1999.

[54] Miron Abramovici and Daniel Saab. Satisfiability on reconfigurable hardware. In *International Workshop on Field Programmable Logic and Applications*, pages 448–456. Springer, 1997.

[55] Prabhakar Goel. An implicit enumeration algorithm to generate tests for combinational logic circuits. *IEEE transactions on Computers*, (3) :215–222, 1981.

[56] Miron Abramovici and Jose T De Sousa. A sat solver using reconfigurable hardware and virtual logic. *Journal of Automated Reasoning*, 24(1-2) :5–36, 2000.

[57] Andreas Dandalis and Viktor K Prasanna. Run-time performance optimization of an fpga-based deduction engine for sat solvers. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 7(4) :547–562, 2002.

[58] Mark Redekopp and Andreas Dandalis. A parallel pipelined sat solver for fpga's. In *International Workshop on Field Programmable Logic and Applications*, pages 462–468. Springer, 2000.

[59] Wong Hiu Yung, Yuen Wing Seung, Kin Hong Lee, and Philip Heng Wai Leong. A runtime reconfigurable implementation of the gsat algorithm. In *International Workshop on Field Programmable Logic and Applications*, pages 526–531. Springer, 1999.

[60] Philip Heng Wai Leong, CW Sham, WC Wong, HY Wong, Wing Seung Yuen, and Monk-Ping Leong. A bitstream reconfigurable fpga implementation of the wsat algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1) :197–201, 2001.

[61] CK Chung and Philip Heng Wai Leong. An architecture for solving boolean satisfiability using runtime configurable hardware. In *Parallel Processing, 1999. Proceedings. 1999 International Workshops on*, pages 352–357. IEEE, 1999.

[62] José T de Sousa, JM Da Silva, and Miron Abramovici. A configurable hardware/software approach to sat solving. In *Field-Programmable Custom Computing Machines, 2001. FCCM'01. The 9th Annual IEEE Symposium on*, pages 239–248. IEEE, 2001.

[63] NA Reis and José T de Sousa. On implementing a configware/software sat solver. In *Field-Programmable Custom Computing Machines, 2002. Proceedings. 10th Annual IEEE Symposium on*, pages 282–283. IEEE, 2002.

[64] RC Ripado and JT de Sousa. A simulation tool for a pipelined sat solver. 2001.

[65] Iouliia Skliarova and Antonio de Brito Ferrari. A software/reconfigurable hardware sat solver. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(4) :408–419, 2004.

[66] Khadija Bousmar, Fabrice Monteiro, Zineb Habbas, Sofiene Dellagi, and Abbas Dandache. A new fpga-based dpll algorithm to improve sat solvers. In *Microelectronics (ICM), 2015 27th International Conference on*, pages 287–290. IEEE, 2015.

[67] Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.

[68] Luc De Raedt, Anton Dries, Tias Guns, and Christian Bessiere. Learning constraint satisfaction problems : An ilp perspective. In *Data Mining and Constraint Programming*, pages 96–112. Springer, 2016.

[69] Harry R Lewis. Computers and intractability. a guide to the theory of np-completeness, 1983.

[70] Peter C Cheeseman, Bob Kanefsky, and William M Taylor. Where the really hard problems are. In *IJCAI*, volume 91, pages 331–340, 1991.

[71] Ravinderjit S Braich, Nickolas Chelyapov, Cliff Johnson, Paul WK Rothemund, and Leonard Adleman. Solution of a 20-variable 3-sat problem on a dna computer. *Science*, 296(5567) :499–502, 2002.

[72] Michael Buro and H Kleine Büning. *Report on a SAT competition*. Fachbereich Math.-Informatik, Univ. Gesamthochschule, 1992.

[73] Marco Platzner and Giovanni De Micheli. Acceleration of satisfiability algorithms by reconfigurable hardware. In *International Workshop on Field Programmable Logic and Applications*, pages 69–78. Springer, 1998.

[74] Iouliia Skliarova and António B Ferrari. A hardware/software approach to accelerate boolean satisfiability. In *Proc. IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop*, pages 270–277. Citeseer, 2002.

[75] Kanupriya Gulati, Mandar Waghmode, Sunil P Khatri, and Weiping Shi. Efficient, scalable hardware engine for boolean satisfiability and unsatisfiable core extraction. *IET Computers & Digital Techniques*, 2(3) :214–229, 2008.

[76] Mona Safar, M Watheq El-Kharashi, Mohamed Shalan, and Ashraf Salem. A reconfigurable, pipelined, conflict directed jumping search sat solver. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, 2011.

[77] Jason Thong and Nicola Nicolici. Fpga acceleration of enhanced boolean constraint propagation for sat solvers. In *Proceedings of the International Conference on Computer-Aided Design*, pages 234–241. IEEE Press, 2013.

[78] Jason Thong and Nicola Nicolici. Sat solving using fpga-based heterogeneous computing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 232–239. IEEE Press, 2015.

[79] Khadija Bousmar, Fabrice Monteiro, Zineb Habbas, Sofiene Dellagi, and Abbas Dandache. A pure hardware k-sat solver architecture for fpga based on generic tree-search. In *Microelectronics (ICM), 2017 29th International Conference on*, pages 1–5. IEEE, 2017.

[80] Khadija Bousmar, Fabrice Monteiro, Zineb Habbas, Sofiene Dellagi, Abbas Dandache, and Karim Tabba, Mohamed ans Alami. A pure hardware k-sat solver architecture using fpga for developing decision-support tools for industrial problems. In *International conference on complexe systems and logistics, ICOSYL'18, 11-13 Avril 2018, IUT de Montreuil-Paris*, 2018.