CENTRE DE RECHERCHE
EN INFORMATIQUE DE LENS

De la résolution du problème SAT à la résolution de problèmes autour de SAT

Mémoire d'

Habilitation à Diriger des Recherches

Université d'Artois

(Spécialité Informatique)

présentée et soutenue publiquement le 15 décembre 2000

par

Lakhdar SAÏS

Composition du jury

Rapporteurs :	Michel CAYROL	Professeur, Université Paul Sabatier, Toulouse
	Jean-Jacques CHABRIER	Professeur, Université de Bourgogne, Dijon
	Marie-Catherine VILAREM	Professeur, Université de Montpellier II
Examinateurs :	Éric GRÉGOIRE	Professeur, Université d'Artois (directeur de recherche)
	Gérard FERRAND	Professeur, Université d'Orléans
	Pierre MARQUIS	Professeur, Université d'Artois
	Pierre SIEGEL	Professeur, Université de Provence, Aix-Marseille I

UNIVERSITÉ D'ARTOIS – CENTRE DE RECHERCHE EN INFORMATIQUE DE LENS
rue de l'Université, SP 16 – F-62307 LENS CEDEX
Tél : +33 (0)3 21 79 32 74 – Fax : +33 (0)3 21 79 32 72

Remerciements

Merci à Michel CAYROL, Jean-Jacques CHABRIER et Marie-Catherine VILAREM d'avoir accepté d'examiner ce modeste travail, malgré vos nombreuses activités et responsabilités. Par votre présence dans les projets « *Bahia* » et « *Classes polynômiales* » de l'ex PRC-IA, j'ai énormément appris à la fois sur le plan scientifique, mais aussi humain.

Merci, à Gérard FERRAND, professeur d'informatique à l'Université d'Orléans de m'avoir fait l'honneur de participer à ce jury.

Merci à Éric GRÉGOIRE de m'avoir fait une grande place au sein de son laboratoire, d'avoir partagé avec moi l'aventure SAT, celle autour de SAT et pour bien d'autres choses. Je lui dois beaucoup.

Merci à Pierre MARQUIS pour m'avoir initié à la compilation des bases de connaissances, et surtout pour avoir enrichi après chaque discussion ma propre base de connaissances. Je lui suis reconnaissant.

Merci à Pierre SIEGEL de m'avoir permis de faire ma thèse d'Université dans sa jeune, dynamique et chaleureuse équipe « raisonnement et preuve » de l'Université de Provence, et de m'honorer de sa présence dans ce jury.

J'ai beaucoup aimé la « *pastaga* », en particulier la « *moresque* », surtout quand on la boit en bonne compagnie et en face de la « Bonne Mère ». À tous les anciens de l'UFR-MIM de l'Université de Provence, je vous dis tout simplement merci pour votre gentillesse, votre solidarité, et pour tous les moments inoubliables que nous avons passés ensemble.

Chèr(e)s ami(e)s marseillais(es), maintenant je supporte le Racing Club de Lens. Qu'il pleuve ou qu'il neige, que Lens gagne ou perde, il y a toujours au moins 30 mille spectateurs au stade Bollaert. Leur attachement indéfectible au RCL montre mieux que tout autre exemple les qualités des gens du Nord.

À Lens, j'ai découvert les vertus d'un bon digestif. Pour ne pas être pris pour ce que je ne suis pas, j'ajoute la mention « *à boire avec modération* ».

À tous les collègues du CRIL, et à tous les membres de l'IUT de Lens, de l'accueil, jusqu'à mon bureau, en faisant le tour par le quatrième département, merci. J'ajouterai une mention spéciale à Assef, Bertrand, Jean-Luc et Sylvie pour m'avoir soulagé ces derniers temps de mes tâches quotidiennes et d'autres plus conjoncturelles.

Bertrand, je tiens à t'exprimer ici ma profonde amitié. Comme je ne pourrai jamais te remercier assez, je te dis simplement merci et « *plus vite que ça !* ».

À Souhila, Ouamar et Massin, je vous demande pardon pour mes longues absences, et je vous promets que cela changera et, surtout, qu'après les fêtes « *adieu la cigarette* ».

Je termine par ceux qui m'ont soutenu malgré la distance, mes parents, mes sœurs et mon frère. Sachez que par mes pensées et mon cœur je suis souvent avec vous. Merci de m'avoir appris le métier de berger avant de rejoindre les bancs de l'école.

À mes grands-parents

Table des matières

Avant-propos **1**

Organisation du mémoire **3**

Partie I Synthèse des travaux de recherche **5**

1 Introduction **7**

2 Problème SAT **9**

2.1 Définitions de base et bref état de l'art 9

2.1.1 Les classes polynômiales du problème SAT 11

2.1.2 Algorithmes de résolution 11

2.1.2.1 Résolution 12

2.1.2.2 Énumération 12

Simplifications : 13

Heuristiques : 14

Traitement des échecs : 14

2.1.2.3 Recherche locale 14

2.1.3 Instances de SAT 15

2.1.4 Analyse de complexité 17

2.2 Exploitation de propriétés structurelles 17

2.2.1 Symétries 18

2.2.2 Équivalences 20

2.3 Aspects codage et extension du formalisme 21

2.3.1 Logique à cardinalité 21

2.3.2 Formalisme étendu 22

2.4 Aspects algorithmiques 24

2.4.1 Recherche locale 24

2.4.2 Recherche systématique 24

2.4.2.1 Généralisation du théorème de partition du modèle 24

2.4.2.2 Poids : apprentissage au cours de la recherche 25

2.4.3 Méthode mixte : coopération entre approche systématique et non systématique 26

2.5 Phénomène de seuil et génération aléatoire 27

3 Problèmes autour de SAT 29

3.1 Problèmes de satisfaction de contraintes 29

3.2 Compilation des bases de connaissances et représentation compacte de l'ensemble des solutions 31

3.3 Des techniques efficaces pour SAT au secours de la révision de croyances 32

3.4 Une méthode complète pour des bases de connaissances propositionnelles non monotones 32

3.5 Coopération consistante et non-monotonie 33

3.6 Premier ordre fini 33

4 Perspectives **34**

4.1 Problème SAT 34

4.1.1 Approches traditionnelles 35

4.1.1.1 Aspects algorithmiques 35

Raisonner : 35

Spécialiser : 35

Combiner : 35

Implémenter efficacement : 36

4.1.1.2 Fonctions booléennes générales et propriétés structurelles 37

4.1.1.3 Codage 38

4.1.2 Autres approches 38

4.1.2.1 Résolution étendue 38

4.1.2.2 De la structure dans les instances aléatoires ? 38

4.1.2.3 Résolution de SAT par comptage 39

4.1.2.4 Formule CNF et représentation en terme de graphes 39

4.2 Autour de SAT 39

4.2.1 Problèmes de satisfaction de contraintes 39

4.2.1.1 Filtrages et algorithmes de résolution 39

4.2.1.2 Heuristiques 40

4.2.1.3 Extension de la substituabilité de voisinage 40

4.2.1.4 Compilation et représentation d'ensembles de solutions 41

4.2.1.5 CSP n-aire 41

4.2.2 Procédures de preuve pour les logiques non standard 41

4.3 Applications 42

Bibliographie **43**

Partie II Curriculum Vitae **57**

5 État civil et cursus **59**

5.1 Notice individuelle 59

5.2 Diplômes 59

5.3 Activités professionnelles 60

6 Activités de recherche **61**

6.1 Encadrement de la recherche et animation scientifique 62

6.1.1 Activités d'encadrement 62

6.1.2 Participation à des jurys de thèse 62

6.1.3 Participation à des comités de programme et d'organisation 63

6.1.4 Participation à des projets de recherche 63

6.1.5 Relecture d'articles 63

6.2 Publications 63

6.2.1 Thèse 63

6.2.2 Édition d'actes de conférences 64

6.2.3 Ouvrages et chapitres d'ouvrages 64

6.2.4 Articles publiés dans des revues d'audience internationale avec comité de rédaction 64

6.2.5 Communications à des manifestations internationales avec comité de sélection 64

6.2.6 Conférences d'audience nationale avec comité de sélection 66

6.2.7 Tutoriels 67

6.2.8 Posters 67

6.2.9 Articles collectifs 67

6.2.10 Colloques et journées sur invitation 67

6.2.11 Rapports internes 68

6.2.12 Articles soumis 68

6.2.13 Travaux en cours 68

6.2.14 Rapport d'activités 68

6.2.15 Séminaires 68

7 Activités administratives **70**

7.1 Mandats électifs 70

7.2 Responsabilités diverses 70

8 Activités d'enseignement **71**

8.1 Synopsis 71

8.2 Contenu détaillé 72

8.3 Autres activités liées à l'enseignement 74

Partie III Publications jointes **75**

Tractability Through Symmetries in Propositional Calculus **79**

Two proof procedures for cardinality based language in propositional calculus **97**

Calcul propositionnel : Vers une extension du formalisme **113**

Tabu Search for SAT **125**

Boosting complete techniques thanks to local search methods **133**

About the Use of Local Consistency in Solving CSPs **151**

Tractable Cover Compilations **159**

Iterated syntax-based revision in a nonmonotonic setting **167**

Avant-propos

Avant de commencer la rédaction de ce document, j'ai naturellement feuilleté de nombreux mémoires d'habilitation à diriger des recherches. Je ne déroge pas à la tradition de situer dans le temps les environnements dans lesquels j'ai évolué, en profitant pour remercier les personnes que j'ai eu la chance de rencontrer. Je me limiterai dans cet-avant propos aux personnes ayant influencé directement mon itinéraire scientifique.

- je n'aurais pas fait d'études sans mon grand-père OUAMAR grâce à qui *l'école primaire* du village a vu le jour. Ce mémoire est pour toi !

- mes premiers pas en recherche ont été effectués en fin de cycle d'ingénieur à *l'Institut National d'Enseignement Supérieur en Informatique de l'Université de Tizi-Ouzou*. L'étude que j'ai réalisée sous la direction du Professeur Md Said AIDENE, concerne la *résolution des problèmes de programmation linéaire à grand nombre de paramètres (de grande taille)*, en utilisant différentes techniques de décomposition. Ce projet m'a permis de découvrir un domaine très riche "la recherche opérationnelle" qui a des liens étroits avec mes centres d'intérêts actuels. Merci MOHAND pour la documentation en russe que vous avez eu la gentillesse de me traduire.

- Merci au Professeur Henri MOREL (un grand Monsieur) pour avoir retenu ma candidature au DEA de Mathématiques et d'Intelligence Artificielle de *l'Université d'Aix-Marseille II, Luminy* et de m'avoir initié au problème SAT (Satisfiabilité d'une formule booléenne mise sous forme normale conjonctive), aux techniques de résolution (SL-Résolution, la procédure de Davis et Putnam), etc.

- Une belle aventure avec mon directeur de thèse Pierre SIEGEL a commencé en 1990 au *Laboratoire d'Informatique d'Université de Provence (LIUP)*, par une petite histoire de pigeons (tiroirs et chaussettes). C'était l'époque des puzzles logiques, des reines, des problèmes de Ramsey, pigeons, etc. Ces problèmes présentent pour une large part des structures très symétriques. La prise en compte de ces informations structurelles lors de la résolution a permis un gain considérable en efficacité. Conscient du lien entre codage et résolution de problèmes, nous avons proposé un formalisme (à cardinalité) permettant une expression plus naturelle et compacte des problèmes et de leurs propriétés de symétries. Cette étude a été conduite en collaboration étroite et quotidienne avec Belaid BENHAMOU que je remercie chaleureusement.
  Je raconte une partie de cette belle aventure qui est loin d'être terminée.
  Pierre, avec le recul, je m'aperçois que tu as représenté pour moi plus qu'un directeur de thèse. Tu fus un exemple pour tes qualités humaines difficiles à énumérer. Je t'exprime ici ma profonde gratitude !
  Je ne peux conclure l'étape marseillaise sans citer Antoine RAUZY, nos discussions souvent informelles et amicales ont été très riches. C'était la période des problèmes aléatoires difficiles, des phénomènes de seuil, des algorithmes de recherche locale, etc.
- Je mesure aujourd'hui ce que des projets comme *BAHIA (Booléens Algorithmes et Heuristiques pour l'IA), classes polynomiales de l'ex PRC IA* peuvent représenter pour un jeune chercheur. J'exprime ici, ma reconnaissance à mes aînés et à tous les membres que je continue à revoir avec beaucoup de

plaisir dans le cadre des *Journées Nationales sur la résolution Pratique de problèmes NP-Complet (JNPC)* et des réunions du *groupe de travail "Axe algorithme" du PRC GdR I3*.

– J'ai rejoint *le Centre de Recherche en Informatique de Lens (CRIL)* en septembre 94. Ces six années passées à Lens ont été extrêmement riches à la fois sur le plan personnel et professionnel. Une page ne suffirait pas à exprimer ma reconnaissance et mes remerciements. J'exprime ma reconnaissance à tous les collègues qui m'ont guidé dans la visite autour de SAT:
  – logiques non monotones, révision, fusion et validation des systèmes à base de connaissances (Éric GRÉGOIRE)
  – compilation des bases de connaissances (Pierre MARQUIS et Yacine BOUFKHAD)
  – problèmes de satisfaction de contraintes (Assef CHMEISS)

  J'ajoute une mention spéciale à mes étudiants Bertrand MAZURE et Laure BRISOUX (aujourd'hui tous deux maîtres de conférences) que j'ai eu la chance de co-encadrer sous la direction d'Éric GRÉGOIRE.

En résumé, j'ai appris tout au long de ces dix dernières années que la recherche est avant tout un travail d'équipe. Je peux même dire que je n'ai rarement travaillé seul, à l'exception de l'année 1993-1994 (année séparant la fin de ma thèse de ma nomination à l'IUT de Lens). Ce qui m'amène à dire que sans toutes ces personnes, ce document n'aurait pas vu le jour. C'est dans ce sens qu'il faut comprendre le terme *nous* très employé dans ce mémoire.
Dans celui-ci, j'ai choisi volontairement de privilégier la présentation d'un domaine qui a occupé une place importante dans mes activités de recherche pour lequel ma contribution personnelle a été la plus significative : la résolution du problème SAT et plus récemment les problèmes de satisfaction de contraintes (CSP). Dans la suite, les références concernant les travaux auxquels j'ai contribués seront mentionnées en caractères **Gras**.

Organisation du mémoire

Ce document est divisé en trois parties. La *première partie* est une synthèse de mes activités scientifiques. J'ai longtemps hésité quant à la forme et à l'organisation que je devais donner à cette partie centrale du mémoire. Étant co-responsable avec Philippe JÉGOU de l'axe algorithme du GdR I3, j'ai eu récemment l'occasion de rédiger une synthèse des activités du groupe de travail ; synthèse axée sur la réponse à trois questions :

- quelles sont les avancées scientifiques récentes (résultats établis) ?
- quels sont les verrous scientifiques actuels et les sujets ou thèmes de recherche émergeants ou qui devraient être lancés ?
- quelles sont les applications majeures en regard de ces sujets ou thèmes de recherche ?

Ce travail m'a aidé au choix de l'organisation de cette partie du mémoire. En effet, en regroupant les travaux récents de la communauté autour de quatre axes : codage, résolution, extensions, et applications, je me suis aperçu que j'ai moi même travaillé sur ces différents aspects. En outre, le titre qui a été donné par Pierre MARQUIS au sous-groupe de travail du GT 1.2 dont il est le responsable s'est pour moi révélé le meilleur titre reflétant mes travaux de recherche : « Utilisation des techniques de résolution du problème SAT à la résolution de problèmes autour de SAT ». Pour exprimer ma démarche dans le temps, j'ai légèrement modifié le titre précédent en : « De la résolution du problème SAT à la résolution de problèmes autour de SAT ». Ce qui m'a conduit a structurer la première partie de ce document en quatre chapitres :

- *Introduction* : dans ce chapitre je présente le contexte de mes travaux et la démarche scientifique adoptée.
- *Problème SAT* : ici sont exposés mes travaux sur la résolution de systèmes de contraintes booléennes. Après une description du problème, un *bref* état de l'art des travaux sur SAT est donné. Je décris ensuite mes travaux autour de quatre thèmes : exploitation de propriétés structurelles, aspects liés au codage et à l'extension du formalisme propositionnel, aspects algorithmiques et enfin génération aléatoire et phénomènes de seuils. Ces différents aspects sont bien évidemment très liés. Par exemple, les propriétés structurelles ont été étudiées dans le but d'améliorer la résolution de certains problèmes structurés et difficiles. On sait aussi que le formalisme utilisé et la manière de coder un problème ont une incidence directe sur la résolution proprement dite.
- *Problèmes autour de SAT* : comme je l'ai évoqué dans l'avant-propos, et afin de rester dans le cadre de la problématique de satisfaction de contraintes, j'ai fait le choix de privilégier la présentation de mes récents travaux relatifs à la résolution des problèmes de satisfaction de contraintes (CSP). Je laisse tout de même une place aux autres travaux : compilation et révision des bases de connaissances, fusion et validation des bases de connaissances et logique du premier ordre.
- Perspectives : dans ce chapitre, je mets en évidence mes perspectives de recherche à court et à moyen termes.

L'avantage de cette organisation est de permettre une bonne structuration. L'inconvénient est de ne pas permettre de voir clairement la démarche scientifique adoptée. Je ferai en sorte de minimiser la portée de cet inconvénient.

La *seconde partie* est un *curriculum vitae* composé de quatre chapitres. Après une brève description de mon état civil, de mon cursus et de mes activités professionnelles (chapitre 5), j'introduis au chapitre 6 mes activités de recherche (activités d'encadrement de la recherche, d'animation scientifique et liste de publications). Je présente ensuite dans le chapitre 7 mes activités administratives et autres responsabilités. Pour conclure cette partie une présentation de mes activités d'enseignement effectuées depuis septembre 1988 est donnée au chapitre 8.

La *troisième partie* de ce mémoire regroupe un sous-ensemble de mes publications parmi les plus significatives.

Première partie

Synthèse des travaux de recherche

Chapitre 1

Introduction

Dans l'introduction de ma thèse d'Université, j'ai écrit : « *La représentation des connaissances et mécanisation des raisonnements est un thème central en Intelligence Artificielle ... Cependant dans le cadre de l'IA, on attend des résultats pratiques : la déduction doit être programmée et permettre d'obtenir des résultats dans un temps raisonnable*». Ce paragraphe très largement inspiré de celui écrit par Pierre SIEGEL dans sa thèse d'État sur la représentation et l'utilisation de la connaissance en calcul propositionnel [Siegel 1987] se retrouve d'une certaine manière dans le texte fondateur du groupe de travail GT 1.2 [1] *(axe algorithmes)* du GdR I3 auquel je participe : « *il est essentiel que les formalismes utilisés pour la représentation des connaissances et la formalisation du raisonnement soient traitables pratiquement et/ou théoriquement... Cet axe couvre donc un champ qui va de l'étude théorique et pratique des problèmes NP-complets, aux procédures de preuve pour les logiques non classiques* ». L'étude théorique et pratique du problème SAT est essentielle si l'on veut disposer un jour de mécanismes de déduction efficace. Ce qui a conduit de nombreux chercheurs ayant travaillé sur les aspects modélisation du raisonnement à s'intéresser de plus près à la logique propositionnelle comme langage de représentation et au problème SAT par son lien à la déduction logique [2].

Je peux maintenir ce premier objectif aujourd'hui, d'autant que j'ai travaillé dans cette direction au cours de ces dernières années en collaboration avec Éric GRÉGOIRE et Pierre MARQUIS. Ainsi, les travaux sur la révision en logique propositionnelle non monotone, sur la compilation et la fusion de bases de connaissances intègrent parfaitement ce cadre. Mais au delà de ces objectifs, le problème SAT par sa place centrale (problème NP-complet de référence [3] [Cook 1971]) en théorie de la complexité rend son intérêt pratique encore plus important. On le retrouve en tant que sous-problème ou cas particulier dans de nombreux domaines comme la déduction automatique, les problèmes de satisfaction de contraintes, la vérification de circuit, la planification, la cryptographie, etc.

Les arguments ci-dessus montrent le caractère fondamental de ce problème. Il reste qu'on peut toujours s'interroger sur le choix d'un langage de représentation aussi rudimentaire qu'est la logique propositionnelle. Ce langage par sa simplicité permet de représenter une grande variété de problèmes et surtout d'étudier finement les algorithmes mis en œuvre. Les progrès récents obtenus autour de la résolution pratique du problème SAT (cf. chapitre 2) ont permis sans aucun doute de conforter cet argument. En effet, les succès croissants obtenus dans la résolution de problèmes réels [Reiter & Mackworth 1989, Kautz & Selman 1992, Zhang & Hsiang 1994, Crawford & Baker 1994, Larrabee 1992, Silva & Sakallah 2000, Biere *et al.* 1999, Massacci & Marraro 2000] mettent en évidence les premiers signes d'un passage à l'échelle industrielle et

1. Ce groupe de travail fait partie du thème 1 « modèles de raisonnement et algorithmes ».
2. $F \models f$ si et seulement si $(F \wedge \neg f)$ est insatisfiable.
3. SAT est le premier problème à avoir été montré complet pour la classe NP (Non déterministe Polynômial).

commerciale (« *Greentech Computing Ltd* » (UK) [Gre], « *Prover Technology AB.* » (Suède) [Prover], etc.). Pour s'en convaincre, il suffit de constater l'évolution quantitative et qualitative des bases de tests utilisées ces dernières années pour évaluer les algorithmes de résolution [Dimacs 1993, Hoos & Stützle 2000]. On peut également trouver des échos de ces résultats dans la presse « grand public » comme l'illustre un article paru dans le « New-York Times Magazine » [Johnson 1999].

Ce saut qualitatif ouvre et/ou remet au goût du jour de nombreuses perspectives (cf. chapitre 4) qui vont du codage à la résolution, en passant par l'étude des extensions possibles des cadres existants et de l'utilisation des techniques de SAT à la résolution de problèmes autour de SAT. C'est dans ce contexte que se situe une grande partie de mes travaux de recherche.

Le premier objectif de mes travaux concerne l'élargissement de la classe des instances traitables : il s'agit de pousser aussi loin que possible les limites des algorithmes existants. L'approche utilisée consiste d'une part à ajouter de nouveaux principes aux algorithmes, en exploitant par exemple, les propriétés structurelles et syntaxiques des problèmes traités, et d'autre part, à améliorer le formalisme d'expression des problèmes tout en restant dans le cadre du calcul propositionnel. Comme je l'ai déjà évoqué, il s'agit ensuite d'étendre et/ou d'utiliser certains résultats obtenus sur SAT à d'autres domaines comme la compilation et la révision des bases de connaissances ou encore les problèmes de satisfaction de contraintes.

Dans la suite, je détaille mes recherches selon deux axes :

- travaux sur le problème SAT (exploitation des propriétés structurelles, aspects codage, aspects algorithmiques, phénomène de seuil et génération aléatoire) ;
- travaux autour de SAT (problèmes de satisfaction de contraintes, compilation et représentation de l'ensemble des solutions, révision et fusion de bases de connaissances).

Chapitre 2

Problème SAT

Sommaire

2.1 Définitions de base et bref état de l'art 9
2.1.1 Les classes polynômiales du problème SAT 11
2.1.2 Algorithmes de résolution 11
2.1.3 Instances de SAT 15
2.1.4 Analyse de complexité 17
2.2 Exploitation de propriétés structurelles 17
2.2.1 Symétries 18
2.2.2 Équivalences 20
2.3 Aspects codage et extension du formalisme 21
2.3.1 Logique à cardinalité 21
2.3.2 Formalisme étendu 22
2.4 Aspects algorithmiques 24
2.4.1 Recherche locale 24
2.4.2 Recherche systématique 24
2.4.3 Méthode mixte : coopération entre approche systématique et non systématique 26
2.5 Phénomène de seuil et génération aléatoire 27

2.1 Définitions de base et bref état de l'art

Une *variable booléenne* est une variable prenant ses valeurs dans l'ensemble $\{vrai, faux\}$. Soit $X = \{x_1, x_2, \ldots, x_n\}$ un ensemble de n variables booléennes. Une *affectation (partielle)* des variables de X est une fonction (partielle) $I : X \rightarrow \{vrai, faux\}$. On distingue pour chaque variable x deux littéraux, x (*un littéral positif*) et $\neg x$ (*un littéral négatif*) où $\neg$ est le symbole de négation logique. Un littéral x (resp. $\neg x$) est vrai dans I si et seulement si $I(x) = vrai$ (resp. $I(x) = faux$).

Une *formule booléenne mise sous forme normal conjonctive (CNF)* est une conjonction de m clauses [1] : $\mathcal{F} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$. Chaque *clause* C_i est un ensemble de littéraux combinés par l'opérateur logique *ou* ($\vee$). Une affectation (interprétation) I est dite solution ou *modèle* de $\mathcal{F}$ si elle satisfait chaque clause de $\mathcal{F}$.

1. On désigne souvent une formule CNF comme un ensemble de clauses.

Le problème SAT est un problème de décision qui consiste à déterminer s'il existe un modèle de la formule d'entrée $\mathcal{F}$. La formule $\mathcal{F}$ est dite satisfiable si la réponse est positive et insatisfiable sinon.

On désignera par $\mathcal{F}(x)$ la formule $\mathcal{F}$ simplifiée par l'affectation de x à $vrai$. $\mathcal{F}(x) = \{C|C \in \mathcal{F}, \{x, \neg x\} \cap C = \emptyset\} \cup \{C \backslash \neg x | C \in \mathcal{F}, \neg x \in C\}$. L'ensemble de variables de $\mathcal{F}$ est noté $var(\mathcal{F})$. La taille d'une clause C_i (notée $|C_i|$) est égale au nombre de littéraux de la clause. Une clause de taille 2 (resp. 1) est dite binaire (resp. unitaire). Un littéral est dit *littéral unitaire* ou *mono-littéral* (resp. *littéral pur ou monotone*) s'il apparaît dans une clause unitaire (resp. si son opposé n'apparaît pas dans la formule)[2]. La *propagation unitaire* (resp. *des littéraux purs*) désigne le processus consistant à simplifier la formule en répétant l'opération de satisfaction des littéraux unitaires (resp. des littéraux purs). On note $k - SAT$ les instances dont les clauses sont toutes de longueur k. En fait, toute formule booléenne peut être transformée en temps linéaire en un ensemble de clauses équivalent du point de vue de la satisfiabilité. Cette transformation nécessite l'ajout de variables supplémentaires. De la même manière une formule CNF peut également se réduire à une instance de 3-SAT (une restriction de SAT qui reste NP-complète [Cook 1971]). Enfin, la taille d'une formule est donnée par $|\mathcal{F}| = \sum_{i=1}^{m} |C_i|$.

Exemple :
Donnée : Une formule booléenne mise sous forme CNF :
$\mathcal{F} = (a \vee b \vee c) \wedge$
$(\neg a \vee b) \wedge$
$(\neg b \vee c) \wedge$
$(\neg c \vee a)$
Question : Est-ce que la formule $\mathcal{F}$ admet au moins un modèle?
Réponse : Pour cet exemple, la réponse est oui : l'affectation $a = vrai, b = vrai, c = vrai$ satisfait la formule $\mathcal{F}$. Comme $\mathcal{F}$ admet une seule solution, on a $\mathcal{F} \wedge (\neg a \vee \neg b \vee \neg c)$ est insatisfiable.

La première méthode qui vient à l'esprit (sans être un spécialiste de SAT) pour décider si une formule $\mathcal{F}$ est satisfiable ou non est celle qui consiste à passer en revue les 2^n affectations possibles. Ce qui donne une complexité en temps dans le pire cas de l'ordre de 2^n. A l'heure actuelle, il n'existe pas de méthode permettant de décider de la satisfiabilité de toute formule de type $\mathcal{F}$ en temps polynômial en fonction de n. L'existence d'un tel algorithme est peu vraisemblable, même si sa non-existence n'a pas été prouvée. En effet, la découverte fondamentale de S.A. COOK au début des années 70 est que de nombreux problèmes (e.g. en recherche opérationnelle et en théorie des graphes) peuvent être réduis au problème SAT. Cette notion de réduction entre différents problèmes est centrale en théorie de la complexité. M.R. GAREY et D.S. JOHNSON [Garey & Johnson 1979] dans leur ouvrage présentent un large catalogue de problèmes NP-complets. La résolution efficace d'un de ces problèmes aura une incidence immédiate sur l'ensemble du catalogue.

Vu le nombre important de travaux de recherche réalisés depuis de nombreuses années sur SAT, il est très difficile, voire impossible, de réaliser un inventaire complet. Ce qui dépasserait le cadre de cette synthèse. Je me limiterai à un exposé des travaux importants et/ou en relation directe avec mes travaux de recherche. Le lecteur intéressé par un panorama et une liste bibliographique assez complète peut se reporter au récent « Survey » traitant du problème SAT [Franco *et al.* 1997].

Pour "résoudre SAT", on ne peut donc que chercher des heuristiques et de nouveaux raffinements permettant d'accélérer en pratique les traitements ou exhiber des restrictions pour lesquelles on peut garantir une résolution en temps polynômial.

2. Si x est un littéral unitaire ou pur dans $\mathcal{F}$, alors $\mathcal{F}$ et $\mathcal{F}(x)$ sont équivalents du point de vue de la satisfiabilité.

2.1.1 Les classes polynômiales du problème SAT

Une classe polynômiale est un ensemble de formules qu'il est possible de reconnaître et de résoudre en temps polynômial. Une classe polynômiale admet d'autant plus d'intérêt qu'elle recouvre une large classe de problèmes réels. Outre l'intérêt théorique de ce type d'étude, un autre bénéfice concerne l'exploitation de ces classes pour mieux traiter le cas général [Gallo & Urbani 1989] ou dans d'autres domaines comme la compilation des bases de connaissances [Selman & Kautz 1991, Marquis 1995].
Parmi les « nombreuses » classes polynômiales du problème SAT, 2-SAT [Cook 1971] (instances formées de clauses binaires) et Horn-SAT (instances formées de clauses ayant au plus un littéral positif) restent sans aucun doute parmi les plus intéressantes. En effet, les clauses binaires se retrouvent très fréquemment dans les problèmes réels, un traitement particulier sur les clauses binaires améliore considérablement l'efficacité des algorithmes de résolution. On sait aussi que les clauses de Horn sont à la base des langages de programmation logique, etc. On retrouve ensuite les formules Horn-renommables (des formules qui se réduisent à Horn par un renommage adéquat des littéraux) et les q-Horn (une généralisation de Horn-SAT et 2-SAT). D'autres restrictions polynômiales ont été exhibées soit par l'introduction de différentes formes de hiérarchies d'instances (e.g. hiérarchie de G. Gallo et M.G. Scutellà [Gallo & Scutellà 1988], hiérarchies de Dalal-Etherington [Dalal & Etherington 1992]), soit en restreignant le nombre d'occurrences des variables (e.g. r,r-SAT de Tovey [3] [Tovey 1984, Dubois 1990], formules dont les variables apparaissent au plus deux fois [Tovey 1984]). D'autres classes ont été obtenues grâce à des formulations en programmation linéaire (e.g. les formules équilibrées [Conforti & Cornuéjols 1992], généralisation de clauses de Horn [Chandru & Hooker 1990]). Plus récemment, en se basant sur la constatation qu'une utilisation appropriée de la procédure de Davis et Putnam [Davis *et al.* 1962] permet de récupérer une bonne partie des classes polynômiales connues, d'autres généralisations ont été obtenues, la reconnaissance dans ces cas est basée sur un algorithme plutôt que sur les propriétés de la formule (e.g. les travaux d'A. RAUZY et R. GÉNISSON [Rauzy 1994, Rauzy 1995b, Génisson & Rauzy 1996], la classe SLUR (Single-Lookahead Unit Resolution) [Schlipf *et al.* 1995]).

2.1.2 Algorithmes de résolution

De nombreux algorithmes ont été proposés pour résoudre SAT. Il serait présomptueux de prétendre réaliser un inventaire complet de ces méthodes. Les principes à la base de ces méthodes sont très variés : résolution [Robinson 1965, Kowalski & Kuehner 1971, Loveland 1978], réécriture [Boole 1854, Shannon 1938], énumération [Davis *et al.* 1962, Jeroslow & Wang 1990], comptage du nombre de solutions [Iwama 1989], recherche locale [Selman *et al.* 1992], programmation linéaire en nombre entiers [C.E. Blair & Lowe 1986, Hooker 1988, Bennaceur & Plateau 1992, Bennaceur & Plateau 1995], diagrammes binaires de décision [Akers 1978, Bryant 1992, Uribe & Stickel 1994], algorithmes génétiques et évolutionnistes [Hao 1995], etc. On peut distinguer parmi ces différentes méthodes deux catégories : les algorithmes complets et les algorithmes incomplets. Un algorithme pour SAT est *incomplet*, si pour toute formule pour laquelle la réponse est « oui », celle-ci est satisfiable, et *complet* si en plus la réponse « oui » (resp. « non ») est retournée (en un temps fini) pour toute formule satisfiable (resp. insatisfiable).

J'ai essentiellement travaillé et/ou utilisé les algorithmes basés sur la résolution, l'énumération et la recherche locale. Sans viser à être exhaustif, dans la suite, je limite la présentation aux algorithmes utilisant ces trois principes. Les techniques à base d'énumération et de recherche locale restent de loin les plus efficaces en pratique. En outre, je pense que la résolution, par son histoire, sa complémentarité aux autres approches et la puissance de plus en plus croissante des machines jouera un rôle encore plus important dans la prochaine génération d'algorithmes.

3. r,s-SAT désigne l'ensemble des formules où les clauses ont exactement r littéraux et chaque variable admet au plus s occurrences.

2.1.2.1 Résolution

Le principe (ou règle) de résolution consiste à produire à partir de deux clauses $C_1 = (l \vee V)$ et $C_2 = (\neg l \vee W)$, une clause (appelée *résolvante*) $Res_l(C_1, C_2) = (V \vee W)$. La *résolution* consiste à répéter l'application de principe jusqu'à la génération d'une clause vide (auquel cas la formule est insatisfiable). On appelle réfutation, toute suite de clauses générées par ce processus, la dernière étant une clause vide[4]. Une réfutation peut être représentée par un *arbre binaire de résolution* dont la racine est la clause vide, tout nœud interne est étiqueté par la résolvante produite à partir de ses deux fils, les feuilles sont étiquetées par des clauses de la formule.
La résolution dont l'origine remonte à Blake [Blake 1937] (voir aussi une variante de la résolution, la version originale de la procédure de Davis et Putnam [Davis & Putnam 1960]) est souvent attribuée à Robinson [Robinson 1965] qui a proposé une procédure de preuve en calcul des prédicats. Cette procédure permet de combiner le principe de résolution et d'unification [Robinson 1965]. Cette méthode est relativement inefficace car le nombre de résolvantes à engendrer est souvent important. De nombreuses stratégies ont été proposées afin de limiter le nombre de clauses produites ; parmi elles, citons la résolution linéaire[5] et ses diverses variantes (SL-résolution [Kowalski & Kuehner 1971], SLRI (SL-résolution avec Remontée des Impasses) [Cubbada & Mouseigne 1988]), la résolution régulière[6] [Tseitin 1968], la résolution étendue[7] [Tseitin 1968], ..., la procédure de Davis et Putnam (DP) [Davis & Putnam 1960]. La version originale de DP est une variante de la résolution, elle effectue l'élimination successive des variables, en générant pour chaque variable choisie par une heuristique[8], l'ensemble des résolvantes possibles et en éliminant les clauses mentionnant une telle variable. A chaque étape, le sous-problème généré contient une variable en moins, mais un nombre quadratique de clauses supplémentaires. DP exploite en plus les littéraux purs et unitaires.

2.1.2.2 Énumération

L'idée de base des algorithmes énumératifs est la construction d'un arbre binaire de recherche où chaque nœud représente une instanciation partielle des variables. À chaque nœud de l'arbre, on procède généralement à un filtrage (simplification) de la formule courante. Le choix de la prochaine variable à instancier fait l'objet en général d'une heuristique. Un exemple typique est la procédure de Davis, Logemann et Loveland (communément appelée DPL) [Davis *et al.* 1962] : la simplification s'appuie sur la propagation des littéraux purs et des mono-littéraux ; le choix de la prochaine variable à affecter est fait suivant une heuristique (e.g. choix de la variable figurant le plus souvent dans les clauses les plus courtes). La différence entre les deux procédures DP et DPL réside dans le fait que la première procède par élimination de variables en remplaçant le problème initial par un sous-problème plus large, alors que la seconde procède par séparation[9], en remplaçant le problème original par deux sous-problèmes de taille plus réduite. Ces deux procédures admettent des comportements différents et sont incomparables du point de vue de l'analyse de la complexité [Goldberg 1979b, Dechter & Rish 1994]. La procédure DPL est plus souvent implantée et utilisée pour résoudre SAT que la version originale DP ; car la règle d'élimination de variables admet plusieurs inconvénients : elle est plus difficile à programmer que la règle de séparation ; elle tend très rapidement à augmenter le nombre de clauses ; elle génère de nombreuses clauses redondantes ou sous-sommées[10].

4. La résolution est complète pour la réfutation.
5. La résolution est dite linéaire, si à chaque étape (sauf la première), une résolvante est produite à partir de la résolvante courante et d'une résolvante précédente ou d'une clause de la formule.
6. La résolution est dite régulière, si pour tout chemin dans l'arbre de résolution, une variable est éliminée au plus une fois.
7. La résolution étendue, utilise en plus de la règle de résolution, le principe *d'extension*, qui consiste à introduire de nouvelles variables représentant des formules intermédiaire (lemmes).
8. Dans la version originale de DP, la prochaine variable à éliminer est choisie dans la première clause de longueur minimale.
9. Soit $\mathcal{F}$ un ensemble de clauses et p une variable de $\mathcal{F}$, $\mathcal{F}$ est satisfiable si et seulement si $\mathcal{F}(p)$ ou $\mathcal{F}(\neg p)$ est satisfiable.
10. Une clauses C sous-somme C' si et seulement si $C \subset C'$.

Ci-dessous, une description détaillée de la procédure DPL :

Algorithme 2.1 DPL

```
Function DP : boolean
Input  : un ensemble  F de clauses
Output : true si  F est consistant, false sinon
Begin
    F=Simplifier( F) ;                %% propagation des littéraux purs et unitaires
   if ( F contient la clause vide) then return false ;
   elif ( F==∅) then return true ;
   else
     l=Heuristique_de_branchement( F) ;
     return DP( F(l)) || DP( F(¬ l)) ;
   fi
End
```

Par sa simplicité, la procédure DPL est l'une des plus utilisée pour résoudre SAT et est actuellement la plus performante en pratique. Les deux points clés de cette procédure sont la simplification et l'heuristique de branchement utilisée. La première permet de transformer la formule en une formule plus simple équivalente pour SAT ; la seconde a une incidence directe sur la taille de l'arbre de recherche. De nombreuses améliorations lui ont été apportées, elles concernent généralement un (ou plusieurs) des points suivants :

1. simplification de la formule ;
2. choix de la prochaine variable à affecter ;
3. traitement des échecs.

On distingue en général deux types d'approches, les approches prospectives (de type « *look-ahead* ») qui permettent de détecter de futures situations d'échec, et les approches rétrospectives (de type « *look-back* ») qui essayent d'apprendre à partir des situations d'échec. Ce qui correspond respectivement aux améliorations apportées à l'étape de simplification et au traitement des échecs.

Simplifications : Une technique de simplification est intéressante si elle permet non seulement de réduire le nombre de nœuds de l'arbre de recherche, mais également le temps de calcul. En plus de la propagation des littéraux purs et unitaires, de nombreuses techniques de simplification ont été introduites et utilisées, soit comme pré-traitement à la formule, soit au cours de la recherche. Parmi les plus intéressantes, on peut citer :

– la *résolution restreinte* qui consiste à ajouter des résolvantes de taille bornée. Cette technique est très souvent utilisée comme pré-traitement à la formule et permet en particulier de détecter certaines inconsistances locales, et de mieux traiter certaines classes de formules [Boufkhad 1996]. Différentes formes de résolution restreinte plus ou moins complexes ont été exploitées ces dernières années [Billionnet & Sutter 1992, Génisson & Siegel 1994, Van Gelder & Tsuji 1996], etc.
– Les *traitements locaux par propagation unitaire* ont été introduits dans CSAT par Olivier Dubois et al. [Dubois *et al.* 1996]. Ces traitements, appliqués pendant la résolution, permettent de produire par propagation unitaire des littéraux impliqués ou de détecter des inconsistances locales. Cette idée d'exploiter au maximum la propagation unitaire a été utilisée dans la plupart des implémentations performantes actuelles de la procédure DPL (Tableau, POSIT, Satz, etc.).

– L'*exploitation des classes polynômiales* au cours de la résolution a fait l'objet de nombreux travaux. Parmi les restrictions polynômiales connues, 2-SAT et Horn SAT restent les plus exploitées dans le cadre général [Gallo & Urbani 1989, Buno & Buning 1993, E. Boros & Kogan 1994, Larrabee 1992]. Ceci s'explique an partie par le fait que dans de nombreux problèmes réels, certaines contraintes peuvent être codées par des clauses binaires ou des clauses de Horn.

Heuristiques : Du fait de la relation forte entre l'ordre d'affectation des variables et la taille de l'arbre de recherche développé, une « bonne » heuristique de branchement est déterminante pour l'efficacité d'un algorithme de recherche. De nombreuses heuristiques ont été proposées dans la littérature, les plus populaires aujourd'hui ne sont que des variantes de celles suggérées dans [Davis *et al.* 1962, Goldberg 1979b, Jeroslow & Wang 1990] et décrites par PRETOLANI [Pretolani 1993] comme l'heuristique « *Moms* » : choix de la variable ayant le maximum d'occurrences dans les clauses les plus courtes. La première amélioration significative a été proposée dans [Dubois *et al.* 1996], elle se base sur une nouvelle fonction de pondération des clauses et sur l'ajout d'un facteur additionnel pour équilibrer les arbres de recherche. On peut trouver ensuite d'autres variantes, obtenues en exploitant par exemple le traitement local par propagation unitaire [Freeman 1995, Li & Anbulagan 1997], en estimant à l'avance l'effet de la propagation unitaire [Boufkhad 1996] ou encore en considérant les occurrences dans les clauses non Horn [Zhang 1997, Crawford & Auton 1996].

Traitement des échecs : En analysant les situations d'échec, les approches rétrospectives permettent de réaliser un retour-arrière non chronologique. Ce type d'approches a été étudié et appliqué dans de nombreux domaines de l'IA, comme dans les systèmes de maintien de vérité *ATMS* [McAllester 1980, Stallman & Sussman 1977], les problèmes de satisfaction de contraintes [Dechter 1989, Ginsberg 1993, Schiex & Verfaillie 1993] et en démonstration automatique [Bruynooghe 1980]. Elles se distinguent essentiellement par les techniques utilisées pour analyser les échecs et pour éviter de rencontrer les mêmes situations au cours de la recherche. Dans le cadre de SAT, les deux meilleures techniques sont GRASP [Silva & Sakallah 1996] et Relsat [Bayardo Jr. & Schrag 1997]. L'efficacité de ce type d'approches dépend à la fois des problèmes traités et des heuristiques de branchement utilisées. Une bonne heuristique de branchement tend à diminuer la hauteur du saut ou du retour-arrière. L'expérience montre que ce type d'approches a pour effet de corriger les conséquences d'un mauvais choix de variables; ce qui n'est pas exclu lorsqu'on utilise une heuristique, très souvent basée sur des arguments syntaxiques. On peut ranger aussi dans ce cadre l'évaluation sémantique d'OXUSOFF et RAUZY [Oxusoff & Rauzy 1989] et la généralisation proposée par Thierry CASTELL [Castell 1996]. Ces différentes techniques obtiennent leur meilleure performance sur certaines classes de problèmes réels. Il est inutile de les appliquer pour résoudre des instances aléatoires.

2.1.2.3 Recherche locale

Les méthodes considérées ici, contrairement aux algorithmes énumératifs, considèrent des « configurations », c'est-à-dire des instanciations complètes de toutes les variables de la formule. Au départ, ce sont des méthodes d'optimisation (on peut par exemple chercher à maximiser le nombre de clauses satisfaites). Sauf adaptation, ces méthodes sont incomplètes, et ne garantissent donc pas l'obtention d'une solution. En particulier, elles ne permettent pas de prouver qu'une instance est insatisfiable.

Le principe de base des méthodes de recherche locale consiste à se déplacer judicieusement dans l'espace des configurations en améliorant la configuration courante. On peut grossièrement résumer le calcul d'une solution par une méthode de recherche locale de la manière suivante :

Algorithme 2.2 RL

```
1.  Function RL : boolean
2.  Input  :  F un ensemble de clauses, deux entiers : nbEssais et nbDeplacements ;
3.  Output : true si  F admet un modèle, false s'il est impossible de conclure ;
4.  Begin
5.     for i from 1 to nbEssais
6.     do
7.         I = interprétationInitiale( F) ;            %% e.g. générée aléatoirement
8.         for j from 1 to nbDeplacements
9.         do
10.           if (I est un modèle de  F)
11.           then
12.              return true ;
13.           else
14.              do
15.                 P = meilleursVoisins(I,  F) ;          %% e.g. « min-conflict »
16.                 v = choix(P) ;      %% e.g. choisir une variable au hasard parmi P
17.              done
18.              I = inverser(I,v) ;               %% inverser la valeur de v dans I
19.           fi
20.        done
21.    done
22.    return false ;                                          %% modèle non trouvé
23. End
```

En général, les algorithmes de recherche locale sont incomplets ; ils ne permettent donc pas de prouver l'inconsistance et aboutissent généralement a un optimum local. On savait que ce type de méthodes sont très efficaces pour résoudre des problèmes d'optimisation combinatoire. La surprise est venue de ce qu'elles sont aussi très efficaces pour traiter SAT, où une solution exacte est demandée. En effet, au début des années 90, les raffinement proposés par [Minton *et al.* 1990, Chabrier *et al.* 1991, Sosic & Gu 1991], et ensuite par [Selman *et al.* 1992] ont montré leur efficacité pour résoudre certains problèmes difficiles qu'aucune méthode complète n'est capable de traiter encore à l'heure actuelle[11]. Les méthodes de recherche locale ont été appliquées avec succès sur des instances de très grande taille du problème des reines et sur d'autres problèmes structurés, alors que GSAT de B. SELMAN a été appliqué avec succès sur des instances aléatoires difficiles. Ce qui explique à mon avis, le large écho accordé dans la littérature à GSAT comparativement aux autres méthodes. Cette efficacité remarquable a motivé la mise au point de nouvelles variantes [Morris 1993, Gent & Walsh 1993, Selman *et al.* 1994, Cha & Iwama 1995, Hao & Tétard 1996, McAllester *et al.* 1997]. Leurs différences se situent essentiellement dans la fonction de sélection de la prochaine variable à réparer[12] et dans la stratégie utilisée pour s'échapper des minima locaux.

2.1.3 Instances de SAT

L'évaluation d'un algorithme est une étape aussi indispensable que sa mise en œuvre. Comme il semble improbable de trouver un jour un algorithme en temps polynômial pour SAT, déterminer ce qu'est un « bon »

11. Voir [Franco *et al.* 1997], pour une note historique (discutable) sur les techniques de recherche locale, En effet, on peut la compléter par les travaux de HANSSEN et JAUMARD sur Max-Sat [Hansen & Jaumard 1990], ceux de CHABRIER et al. [Chabrier *et al.* 1991] et enfin par une méthode plus ancienne appelée « *Inversion method* » et proposée dans [Dunham & Wang 1976].

12. Réparer une variable consiste à inverser sa valeur de vérité dans l'affectation courante, on utilise aussi le terme « *flipper* ».

algorithme est en soi une question très difficile. En effet, d'un point de vue théorique, l'analyse de sa complexité en moyenne est souvent difficile à mettre en œuvre, alors que d'un point de vue pratique, nous nous heurtons à la difficulté d'établir un jeu de tests «significatifs ».

Cependant l'évaluation expérimentale présente le mérite de permettre l'identification des propriétés des algorithmes et des problèmes traités, qui jusqu'à présent n'ont pas pu être analysés de manière rigoureuse. La recherche d'instances difficiles a pour objectifs de caractériser les limitations des algorithmes et de montrer là où des investigations supplémentaires sont nécessaires.

On peut regrouper les instances utilisées pour évaluer les algorithmes en trois catégories :

1. *problèmes structurés* : on retrouve dans cette catégorie, les puzzles logiques (e.g. problèmes de Lewis Caroll [Carrol 1966], les reines, le problème du zèbre, etc.), le problème des pigeons et les formules de Tseitin [Tseitin 1968], les problèmes de Ramsey [Ramsey 1930], coloriage de graphes, etc.
2. *problèmes aléatoires* : parmi les modèles de génération aléatoire proposés, deux modèles ont été principalement étudiés :
   - *modèle à densité fixée* [Goldberg 1979b] : étant donnés un nombre n de variables, un nombre m de clauses, et une probabilité p fixée. Chaque clause est générée indépendamment, en incluant avec une probabilité p chacun des $2n$ littéraux. Ce modèle génère des instances faciles en moyenne [Goldberg 1979b]. Franco et Paul [Franco & Paul 1983] ont montré que les résultats obtenus par GOLDBERG sont une conséquence directe du modèle utilisé. Ce qui a diminué considérablement la portée de ce modèle.
   - *modèle k-SAT (clause de taille fixée)* : étant donnés un nombre n de variables, un nombre m de clauses, et une longueur k des clauses fixée. Chaque clause est générée indépendamment, en tirant au hasard k variables parmi les n variables et en négativant chaque variable avec une probabilité de $\frac{1}{2}$. Un des résultats les plus importants de ces dernières années est le « phénomène de seuil » mis en évidence sur les données $k-SAT$ aléatoires [AI 1996]. Les expérimentations ont montré une séparation aiguë entre données satisfiables et données insatisfiables (la courbe de probabilité de satisfiabilité passe brutalement de 1 vers 0, lorsque le rapport du nombre de clauses sur nombre de variables augmente, le nombre de variables étant fixé). Le seuil est défini par la limite du rapport m/n quand n devient infini. Ce modèle permet pour la première fois de caractériser des instances SAT difficiles à résoudre en moyenne par toutes les méthodes développées. Ces instances se situent au seuil. Pour 3-SAT par exemple, la valeur expérimentale du seuil (la valeur du ratio m/n telle que la probabilité de tirer une instance satisfiable est égale à $\frac{1}{2}$) est proche de $4,25$. Ce phénomène de seuil observé expérimentalement et indépendamment par plusieurs auteurs [Dubois & Carlier 1991, Cheeseman *et al.* 1991, Mitchell *et al.* 1992, André 1993, Larrabee & Tusji 1993, Crawford & Auton 1993] a permis le développement de méthodes de résolution du problème SAT plus efficaces. Pour des instances $3-SAT$ générées au seuil les meilleures méthodes complètes ne peuvent résoudre des instances au-delà de 500 variables[13], alors que les méthodes incomplètes de type recherche locale sont capables de résoudre des instances satisfiables au seuil au-delà de 2000 variables. Sur le plan théorique, le calcul des seuils pour chaque classe $k-SAT$ en même temps que la preuve de l'existence du phénomène sont activement recherchés [Chvátal & Reed 1992, Goerdt 1992, Kamath *et al.* 1994, Frieze & Suen 1996, Dubois & Boufkhad 1997, Dubois 2000]. L'intérêt de ce modèle est aujourd'hui très largement admis. Cependant, les instances générées par ce modèle sont loin d'être proches des instances issues d'applications réelles. On peut aussi citer d'autres modèles de génération qui ont été mis au point soit dans le but de générer des instances encore plus difficile , soit pour obtenir des instances satisfiables difficiles pour la recherche locale, ou encore "plus proches" des instances qu'on trouve dans le monde réel (cf. section 2.5).
3. *problèmes réels* : Les instances issues d'applications réelles admettent leurs propres caractéristiques, qui peuvent être exploitées en vue d'un traitement efficace d'une application particulière. Ce qui peut

13. Prouver l'insatisfiablité des instances à 700 variables au seuil, est proposé comme un challenge dans [Selman *et al.* 1997].

donner ensuite des indications importantes pour une généralisation à la résolution d'autres domaines d'applications. Ces dernières années ont vu une effervescence particulière autour de la résolution d'applications réelles (e.g. vérification de circuits, planification, vérification de modèles (« *model checking* »), cryptographie, etc.). Cette tendance a été stimulée, d'une part par les progrès récents obtenus sur la résolution pratique de SAT, et d'autre part par les deux compétitions organisées ces dernières années. La première a été organisée en 1993 par le centre de recherche en mathématiques discrètes de Rutgers (DIMACS), la seconde par l'école d'aéronautique de Pékin en 1996. Ce qui a permis notamment de définir un format standard pour coder les instances SAT (format DIMACS) facilitant ainsi la mise en place de librairies de problèmes (DIMACS [Dimacs 1993], SATLIB [Hoos & Stützle 2000]).

2.1.4 Analyse de complexité

De nombreux résultats de complexité ont été obtenus pour certaines des méthodes citées précédemment. On a, par exemple pour la résolution, cherché à améliorer le meilleur minorant connu de la taille des preuves [Tseitin 1968, Galil 1977, Krishnamurthy 1982, Haken 1985, Urquhart 1987]. Ces différents résultats ont été obtenus sur des formules très fortement structurées comme celles codant le problème des pigeons [14], les formules de Tseitin et d'Urquahrt [15], les formules codant des instances du théorème de Ramsey [16] [Ramsey 1930], etc. Certaines de ces formules peuvent être résolues efficacement, en utilisant par exemple la résolution étendue (e.g. [Cook 1976]) ou la résolution avec symétries [Krishnamurthy 1985] (cf. section 2.2.1).

On peut également mentionner les résultats de GOLDBERG et al. [Goldberg 1979a, Goldberg *et al.* 1983] et de PURDOM [Purdom 1990] sur le modèle à densité fixé, montrant que la procédure de Davis et Putnam admet en moyenne un comportement polynômial, les résultats dans le pire cas obtenus par de nombreux auteurs pour les algorithmes énumératifs, de type DPL [Monien & Speckenmeyer 1985, Schiermeyer 1983, Schiermeyer 1996, Kullmann 1999a], ou encore le résultat important de Vašek CHVATAL et Endre SZEMEREDI [Chvátal & Szemerédi 1988] montrant que la résolution est exponentielle sur des instances k-SAT aléatoires. D'autres résultats plus récents ont été obtenus par VANGELDER [van Gelder 1999] et DANTSIN et al. [Dantsin *et al.* 2000].

Les résultats de complexité concernant les techniques à base de recherche locale sont plus difficiles à obtenir, on peut trouver une analyse dans le pire cas pour l'algorithme GSAT dans [Hirsch 2000].

2.2 Exploitation de propriétés structurelles

Pour certaines classes de problèmes réels ou structurés, une des voies pour réduire considérablement l'espace de recherche (cette réduction peut être exponentielle dans certains cas) est sans aucun doute la prise en compte des informations structurelles de la formule considérée. Les algorithmes cités précédemment ignorent pour la plupart ce type d'informations qui sont souvent à l'origine de la difficulté de résolution. Il est à noter que, selon le formalisme utilisé pour coder un problème, ces informations peuvent être plus ou moins apparentes et faciles à exploiter. On peut citer, par exemple, un domaine où la structure du problème joue un rôle considérable, il s'agit des problèmes de satisfaction de contraintes (CSP). En effet,

14. « *Pigeon-Hole problem* » en anglais et « *tiroirs et chaussettes* » en français. Le problème consiste à placer n pigeons dans $n-1$ pigeonniers, avec un et un seul pigeon par pigeonnier.

15. Ces formules se modélisent sous forme d'une conjonction de chaînes d'équivalences (Biconditional Normal Form) et donc résolvables en temps polynômial [Dunham & Wang 1976].

16. Pour donner une idée du théorème de Ramsey, partons de la division par excès $\frac{P}{k}$ = le plus petit entier p qui, multiplié par k, donne $p \times k \geq P$. D'une manière plus intuitive, si nous répartissons P objets dans k tiroirs, l'un au moins des tiroirs comprend au moins p objets.

la représentation en terme de (hyper)graphes d'un CSP offre un avantage certain. Citons, par exemple, les travaux développés autour des techniques de décomposition de CSP [Jégou 1993, Chmeiss 1996], la mise en évidence de classes traitables [Freuder 1982, Dechter 1990, Dechter & Pearl 1989], etc.

Les arguments développés ci-dessus expriment ma démarche scientifique. En effet, je me suis intéressé aux aspects liés au codage, après avoir travaillé sur l'exploitation des propriétés structurelles.

Les travaux autour de ces aspects ont été moins nombreux dans le cadre du problème SAT. On peut constater cependant un regain d'intérêt autour de ce type d'approches, motivé principalement par les applications réelles qui commencent à être résolues en utilisant les techniques de résolution de SAT. On peut imaginer différentes informations structurelles, je me suis intéressé particulièrement à deux types de structures : les symétries et les équivalences.

2.2.1 Symétries

La notion de symétrie est très présente dans la vie courante et dans de nombreux domaines comme la physique, la chimie, les mathématiques, etc. Cette notion, permet par exemple de simplifier certaines preuves mathématiques. Dans le même esprit, notre objectif premier est de voir dans quelle mesure il est possible d'exploiter ces régularités présentes dans de nombreux problèmes réels et structurés (les formules logiques qui les représentent restent invariantes sous certaines permutations de noms de variables) pour améliorer les performances des algorithmes de résolution.

En outre, les formules ingénieuses construites par Tseitin [Tseitin 1968] (le problème des pigeons, etc.) pour établir la complexité de la résolution, admettent des preuves polynômiales en augmentant la résolution soit par la règle d'extension (résolution étendue) proposée par le même auteur, soit par la règle de symétrie proposée par B. KRISHANMURTY [Krishnamurthy 1985]. Ce dernier a affirmé que pour la résolution, tout nouveau résultat de complexité passe nécessairement par l'utilisation du principe d'extension et de symétrie. Au début des années 90, Pierre SIEGEL [Siegel 1990] m'a expliqué les motivations liées à l'étude des symétries en calcul propositionnel. En effet, fort de l'idée que, lors de la résolution de problèmes difficiles, on peut tomber sur un sous-problème de type « *Pigeon-Hole* », cette étude peut aussi servir à mieux résoudre des problèmes qui peuvent ne pas présenter des symétries apparentes au départ.

Cette affirmation qui peut paraître au premier abord utopiste, je l'ai trouvée récemment dans le papier de Douglas D. EDWARDS [Edwards 1996] qui citait Bart SELMAN : « ..., Bart Selman (personal communication) believes that hidden pigeon-hole substructure is a common source of intractability in realistic UNSAT problems ». En se basant notamment sur les récents travaux de Craword et Baker [Crawford & Baker 1994] sur la résolution de problèmes d'ordonnancement (« *scheduling problems* »), des arguments ont été avancés pour justifier cette forte conjecture.

Dans tous les cas, la prise en compte des symétries par une méthode de résolution réclame un algorithme de *détection* de symétries d'une part, et une technique d'*exploitation* des symétries trouvées, d'autre part.

Une partie de notre travail est axée sur la détection des symétries et leur utilisation dans les algorithmes de résolution. Dans un premier temps, ce travail a abouti à l'implémentation d'un algorithme de détection des symétries dans une formule logique donnée sous Forme Normale Conjonctive **[Benhamou & Saïs 1991, Benhamou & Saïs 1992, Benhamou *et al.* 1992b]**. Du point de vue de la complexité, nous avons montré que le problème de la détection des symétries est polynômialement équivalent au problème de l'isomorphisme de graphes **[Saïs 1993]**. Nous avons ensuite introduit les symétries dans différentes méthodes : SL-Résolution, procédure DPL et Evaluation Sémantique. Les motivations précédemment développées nous ont conduits à opter pour une exploitation dynamique (au cours de la recherche) des symétries. La détection des

symétries étant un problème difficile, nous avons choisi d'utiliser une version incomplète de l'algorithme de détection. Des résultats meilleurs ont été obtenus sur la plupart des problèmes testés, en particulier sur les problèmes combinatoires du type recherche de nombres de Ramsey, coloriage de graphes, etc. Une preuve de complexité linéaire a été donnée pour le problème des pigeons. En outre, pour la première fois, le nombre de Ramsey $R(3,3,3)$ [17] a été trouvé par une méthode de déduction. Grâce à l'utilisation de conditions nécessaires à la présence de symétries, d'une stratégie intelligente et de structures de données adaptées, la recherche de symétries n'est que très peu coûteuse en pratique (surcoût de 10% environ quand le problème traité ne contient pas de symétries) **[Benhamou & Saïs 1992, Benhamou & Saïs 1994]**.

Par ailleurs, nous avons été conduits à introduire un concept particulier de symétrie qualifiée de "forte" qui vérifie la condition supplémentaire suivante : quand les k premiers littéraux d'un cycle de symétries C (ensemble de littéraux symétriques deux à deux) sont interprétés à $vrai$, le sous-ensemble de littéraux de C restant après chaque étape de l'interprétation est aussi un cycle de symétries. C est alors un cycle dit "k-fortement symétrique" ("fortement symétrique" lorsque k est égal au nombre de littéraux de C). Cette notion de symétrie forte permet la généralisation des propriétés des symétries simples. En effet, selon le théorème des symétries simples (développé dans ma thèse), si deux littéraux a et b d'une formule $\mathcal{F}$ appartiennent à un même cycle de symétries C, alors, $\mathcal{F} \wedge a$ est satisfiable si et seulement si $\mathcal{F} \wedge b$ est satisfiable. Généralisation : si A est un sous-ensemble de littéraux d'un cycle fortement symétrique C de $\mathcal{F}$, alors $\mathcal{F} \wedge A$ est satisfiable si et seulement si $\mathcal{F} \wedge B$ est satisfiable, pour tout sous-ensemble B de C. Cette nouvelle notion de symétrie est automatiquement utilisable dans le formalisme des formules de cardinalité ; elle permet la production immédiate d'une information supplémentaire précisément caractérisable par une formule de cardinalité (cf. section 2.3.1), donc de raccourcir considérablement la preuve de certains problèmes **[Benhamou *et al.* 1992b, Saïs 1993]**. En effet, si $\mathcal{F} \wedge A$ est insatisfiable, alors on peut déduire l'information suivante : "au plus $|A| - 1$ littéraux de $vrai$ parmi ceux de C".

J'ai effectué une évaluation expérimentale de la procédure DPL avec et sans symétries sur certaines classes de problèmes proposés lors du challenge international organisé par DIMACS. Des résultats intéressants ont été obtenus sur de nombreux problèmes, en particulier sur les formules de *Tseitin* et d'*Urquhart* intraitables par les algorithmes classiques. Notre méthode a été citée lors de ce challenge comme la seule à avoir été capable de résoudre de telles formules **[Saïs 1994b]**.

L'étude des symétries (ou d'autres formes plus faibles) a fait l'objet de nombreux travaux dans différents domaines comme la logique du premier ordre [Crawford 1992, Zhang & Zhang 1995, Slaney *et al.* 1993, Slaney 1994], les problèmes de satisfaction de contraintes [Benhamou 1994, Ellman 1993, Puget 1993, Benson & Freuder 1992], et bien évidement en calcul propositionnel [Aguirre 1992, Chabrier *et al.* 1995, Belleannée & Vorc'h 1994, Boy de la Tour & Demri 1995, Crawford *et al.* 1996, Chabrier 1997]. Je citerai d'une part les travaux de J. CRAWFORD sur les symétries en logique du premier ordre [Crawford 1992], que nous avons découverts pour la première fois lors du « *workshop on tractable reasoning, AAAI'92*» auquel nous avons participé. Dans son papier, CRAWFORD a obtenu un résultat particulièrement intéressant autour de l'utilisation du théorème de Polya [Polya & Read 1987] pour compter le nombre d'interprétations non isomorphes [Crawford 1992]. Dans [Crawford *et al.* 1996], il est proposé une technique d'élimination de symétries par ajout de nouvelles contraintes. Cette technique, peut être utilisée comme pré-traitement de la formule. Cependant, le nombre de contraintes à ajouter peut être prohibitif. D'autre part, un langage de très haut niveau a été mis au point au *LIRSIA* (Université de Bourgogne, Dijon), supportant une recherche locale complète appelé SCORE(FD/B) [Chabrier *et al.* 1996, Chabrier 1997]. Ce langage de représentation fournit des primitives, clauses de cardinalité (cf. section 2.3.1) et structures de contrôle, exploitées statiquement par le compilateur pour déterminer des propriétés des problèmes, par exemple les symétries.

17. $R(3,3,3) = 17$, le plus petit nombre de sommets à partir duquel il est impossible de colorier les arêtes d'un graphe complet avec 3 couleurs sans obtenir un triangle monochromatique. Un graphe à 16 sommets admet un tel coloriage.

2.2.2 Équivalences

L'opérateur d'équivalence ($\Leftrightarrow$) joue un rôle important en logique propositionnelle. En effet, en plus de son utilisation en tant qu'opérateur pour définir des fonctions booléennes quelconques, il est à la base du principe d'extension de *Tseitin* où il est utilisé pour représenter des formules intermédiaires dont la manipulation permet de raccourcir les preuves. Il est aussi très utilisé pour construire des formules très difficiles à la fois pour les algorithmes complets et incomplets. On peut citer les formules de *Tseitin* pour lesquelles la résolution est exponentielle, les formules codant le problème de parité (instances « Parity 32 » de DIMACS) très difficiles à la fois pour les algorithmes complets et incomplets, ou encore les formules difficiles pour la recherche locale, construites par Edward *Hirsch* [Hirsch 2000]. On retrouve ces équivalences dans de nombreux problèmes réels (problèmes de circuit, de model checking, de cryptographie, etc.).

Une des propriétés les plus intéréssantes est le fait que si on a deux littéraux équivalents ($a \Leftrightarrow b$), on peut substituer le littéral a par b partout dans la formule, en gardant une équivalence du point de vue de la satisfiabilité. Cela permet de réduire le nombre de variables et de produire éventuellement de nouveaux littéraux ou encore de détecter des contradictions.

Il faut aussi noter que parmi les formules difficiles citées précédemment, certaines peuvent être entièrement (ou partiellement) codées sous forme d'une conjonction de chaînes d'équivalences (Biconditional Normal Form (BNF)). Les formules de ce type peuvent être résolues en temps polynômial [Dunham & Wang 1976].

Ces remarques justifient à elles seules l'utilisation spécifique des équivalences pour améliorer la résolution de certaines classes de problèmes.

Comme pour les symétries, la prise en compte des équivalences pose le problème de leur *détection*, de leur *utilisation* et du *choix du formalisme* permettant une expression plus naturelle de ce type de structure.

En collaboration avec Laure BRISOUX et Éric GRÉGOIRE, nous avons opté pour une approche sémantique, par propagation unitaire. En effet, pour une variable donnée x, en analysant les littéraux produits par propagation unitaire de $\mathcal{F} \wedge x$ et de $\mathcal{F} \wedge \neg x$, on peut détecter des inconsistances locales et produire non seulement des littéraux, mais également des équivalences entre littéraux. Ce qui permet de généraliser les traitements locaux par propagation unitaire utilisés traditionnellement par de nombreux auteurs. Les littéraux équivalents peuvent ensuite être utilisés en effectuant des substitutions complètes ou partielles. Plus précisément, si $a \Leftrightarrow b$ dans $\mathcal{F}$, alors tout en obtenant une formule équivalente pour SAT, substituer a par b et $\neg a$ par $\neg b$ dans $\mathcal{F}$, ou encore limiter cette substitution de a par b uniquement aux clauses contenant à la fois a (ou $\neg a$) et b (ou $\neg b$). Ce qui revient dans ce dernier cas à satisfaire ou à raccourcir les clauses. Une description de ce travail peut être trouvée dans la thèse de Laure BRISOUX **[Brisoux-Devendeville 1999]**. Daniel LE BERRE propose d'étendre ce processus de détection en effectuant une double propagation unitaire, ce qui permettra de détecter des informations additionnelles. Cette généralisation pose des problèmes de coût ; une collaboration est actuellement en cours sur ce sujet.

Une autre voie consiste à faire des abstractions basées sur la syntaxe. En d'autres termes, reconnaître les chaînes d'équivalences de manière syntaxique à partir de la formule CNF. Une reconnaissance d'équivalences binaires simples a été utilisée par BOUFKHAD [Boufkhad 1996] comme pré-traitement de la formule considérée. Li CHU-MIN dans son papier [Chu-Min 2000] effectue à tous les nœuds de l'arbre de recherche une reconnaissance syntaxique de telles chaînes (limitée à des chaînes de longueur ≤ 3), et utilise un certain nombre de règles de déduction pour raisonner sur de telles formules. Sachant que la classe de formules BNF est stable par adjonction de littéraux unitaires, il est plus judicieux de faire cette reconnaissance sur la formule de départ et d'appliquer ensuite l'algorithme proposé par WANG [Dunham & Wang 1976] pour simplifier la formule.

L'un des inconvénients de la recherche locale, est certainement la non prise en compte de la structure de la formule (e.g. longueur des clauses, les équivalences). Dans **[Brisoux *et al.* 2000]**, nous proposons une utilisation du traitement local par propagation unitaire dans le cadre de la recherche locale. Ce processus de propagation unitaire est effectué d'une part comme pré-traitement de la formule, mais également pour exhiber les dépendances entre littéraux [18] qui sont ensuite utilisées par un algorithme de recherche locale. Signalons que la notion d'équivalence entre littéraux est un cas particulier de la notion de définissabilité définie dans [Lang & Marquis 1998].

2.3 Aspects codage et extension du formalisme

Résoudre efficacement un problème dépend à la fois du formalisme de représentation utilisé (e.g. calcul propositionnel, CSP, etc.) et du codage [19] proprement dit (e.g. les codages des problèmes de planification [Vidal 2000]). J'ai essentiellement travaillé sur les aspects liés à l'extension du formalisme.
Malgré les progrès récents obtenus sur la résolution du problème SAT et une utilisation de plus en plus fréquente d'une représentation CNF dans le cadre de la modélisation des problèmes réels (cf. introduction 1), l'utilisation de formalismes plus puissants et plus généraux reste encore une voie de recherche prometteuse. Ces travaux sont essentiellement motivés par des applications (cf. section 2.3.2) qui permettent une mise en évidence des limites des cadres existants. La tendance dans le cadre SAT, c'est d'aller vers des formalismes de manipulation de fonctions booléenes plus générales [Franco *et al.* 1997, Gu 1997, Groote & Warners 2000, Sheeran & Stalmarck 2000]. Dans le cadre des CSP, on commence à passer du cadre des CSP binaires au CSP n-aires [Régin & eds 1998, Bessière 1999, Smith *et al.* 2000].

2.3.1 Logique à cardinalité

Les travaux réalisés (cf. 2.2) autour de l'exploitation des propriétés structurelles nous ont conduits a rechercher un nouveau formalisme pour exprimer de manière naturelle certains problèmes et plus particulièrement les propriétés structurelles comme la symétrie.

L'approche qui consiste à rechercher une meilleure façon d'exprimer certaines contraintes particulières était très utilisée en programmation logique par contraintes (CLP). On peut citer, les contraintes symboliques (non numériques), par exemple sur des ensembles et leurs éléments (e.g. contraintes Tous Différents « *allDiff* », opérateur de cardinalité [Hentenryck & Deville 1990]). Les variables utilisées dans ce type de contraintes sont en général symétriques. L'utilisation de telles contraintes en CLP a souvent été motivée par des applications réelles (comme les problèmes de planification, d'ordonnancement, etc.) ou encore par le besoin d'exprimer des contraintes disjonctives.

L'utilisation d'opérateurs de cardinalité du type de celui qu'a introduit Pascal Van Hentenryck en programmation logique avec contraintes (CLP) [Hentenryck & Deville 1990] permet une généralisation naturelle du formalisme clausal. La connaissance est donc décrite par un ensemble de couples $(\alpha, \mathcal{L})$ où $\mathcal{L}$ est une liste de littéraux et α le nombre minimal de littéraux de $\mathcal{L}$ qui doivent être satisfaits dans tout modèle de $\mathcal{L}$. Ce formalisme permet l'expression des deux variantes suivantes : la formule $(\alpha, +, \mathcal{L})$: "au plus α littéraux de $\mathcal{L}$ doivent être interprétés à $vrai$", et la formule $(\alpha, e, \mathcal{L})$: "exactement α littéraux de $\mathcal{L}$ doivent être interprétés à $vrai$". Cette nouvelle représentation induit une meilleure expression (plus compacte et naturelle) des problèmes et de leurs propriétés de symétrie [20]; en outre, elle accroît la puissance de discri-

18. Pour chaque littéral l on calcule la liste des littéraux impliqués par propagation unitaire par $\mathcal{F} \wedge l$.
19. Voir compte rendu du sous-groupe de travail du GT12 "Influence du codage en calcul propositionnel" : http ://www.lim.univ-mrs.fr/ gtalgo.
20. Une formule de cardinalité est totalement symétrique.

mination des conditions préalables à la détection des symétries. Par conséquent, la détection des symétries y est moins coûteuse.

Nous avons associé à ce formalisme le développement d'une méthode de résolution (complète et décidable). A l'aide de cette dernière, nous avons donné une nouvelle preuve de complexité linéaire pour le problème des pigeons (sans utilisation des propriétés de symétrie) **[Benhamou *et al.* 1994]**.

De plus, une propriété d'équivalence entre la représentation clausale classique et celle des formules de cardinalité nous a permis d'adapter efficacement certains outils de la résolution classique (méthodes et raffinements) **[Saïs 1993]**.

L'efficacité de la procédure de Davis et Putnam, adaptée au cadre d'une représentation utilisant les formules de cardinalité, est sensiblement améliorée. En outre, l'introduction des symétries, dont les symétries fortes, a permis une amélioration supplémentaire. En effet, une preuve de complexité linéaire est donnée pour le calcul du nombre de Ramsey $R(3,3,3)$ (cf. **[Benhamou *et al.* 1992a, Benhamou & Saïs 1994]**).

On peut trouver dans [Graham *et al.* 1980] une description des nombres de Ramsey trouvés et encore recherchés. Les résultats que nous avons obtenus sur ce sujet grâce à une utilisation combinée des formules de cardinalité et des symétries ouvrent des perspectives quant à la recherche de tels nombres. Le problème qui se pose est certainement celui de trouver un codage plus efficace.

Dans le cadre du projet BAHIA [21] dont l'objectif était d'analyser et de comparer différentes approches (l'approche logique, CSP et programmation linéaire 0/1 (PL 0/1)), le formalisme à cardinalité a constitué une brique importante dans l'étude des pontages entre la programmation linéaire 0/1 et la logique propositionnel (voir **[Bahia 1995]** pour les pontages réalisés entre les trois formalismes). De nombreux travaux ont été réalisés par Hooker autour de l'utilisation des techniques d'optimisation combinatoire en logique (e.g. [Chandru & Hooker 1999]) et dans le cadre CSP (e.g. [Hooker 2000]). Le même auteur a proposé récemment une généralisation des formules de cardinalité, en une représentation sous forme de règles dont la partie gauche et droite sont des formules de cardinalité [22][Hooker & Yan 1999]. D'autres travaux ont été également réalisés autour de l'extension des algorithmes de SAT à la résolution de problèmes d'optimisation combinatoire [Walser 1997, Walser 1999] (pour la recherche locale) et [Barth 1995] (pour la procédure DPL).

2.3.2 Formalisme étendu

Une collaboration est née d'une discussion avec Antoine Rauzy sur l'intérêt de disposer d'un formalisme booléen plus expressif, en vue de modéliser et de résoudre des questions posées dans le cadre d'applications industrielles, en particulier les problèmes liés à l'analyse de sûreté de fonctionnement de systèmes industriels. Les problèmes étant de taille considérable, la représentation CNF s'est avérée très limitée.

Notre objectif est d'étendre au maximum le formalisme booléen tout en :

- restant dans le cadre du calcul propositionnel, lequel nous semble adapté à la résolution de nombreux problèmes pratiques ;
- gardant les mêmes structures de données, en tout cas les mêmes principes de mise en œuvre des algorithmes (algorithmes énumératifs, techniques de recherche locale et les diagrammes binaires de décision) ;

21. BAHIA : Booléen, Algorithmes et Heuristiques pour l'IA.
22. une règle de cardinalité de la forme suivante : $(k, x_1, \ldots, x_m) \Rightarrow (l, y_1, \ldots, y_n)$.

– gardant la même complexité algorithmique sur les formules étendues que sur les formules CNF et surtout la même efficacité pratique.

Nous avons montré que cela est possible sur des circuits combinatoires à plusieurs niveaux et construits avec des fonctions de seuil. Nous avons défini un nouveau formalisme généralisant les formules de cardinalité introduites dans ma thèse. Ce dernier étend les ensembles de clauses dans deux directions :

1. Des formules à deux niveaux (conjonctions de disjonctions) aux formules à un nombre arbitraire de niveaux.
2. Des clauses aux fonctions de seuil.

Plus précisément, une contrainte est représentée par l'équation (ou porte) suivante :

$$s \;=\; \#(\alpha_1.e_1, \ldots, \alpha_k.e_k) \in [\beta_{min}, \beta_{max}](P) \tag{2.1}$$

où s (sortie) et $\{e_1, \ldots, e_k\}$ (entrées) sont des variables propositionnelles, et $\{\alpha_1, \ldots, \alpha_k\}$ (poids) et $\{\beta_{min}, \beta_{max}\}$ (seuils) sont des entiers positifs ou nuls. P est $vrai$ si et seulement si la sortie s et la double inégalitée $\beta_{min} \leq \sum_{i=1,..,k} \alpha_i.e_i \leq \beta_{max}$ sont vérifiées ou infirmées simultanément. Une formule (circuit) est une conjonction d'équations (portes). Elle peut être vu comme un (hyper)graphe orienté acyclique dont les sommets sont les variables et dont les (hyper)arcs relient les entrées de chaque porte avec sa sortie.

Signalons quelques avantages de l'extension proposée sur les formules CNF :

– elle est beaucoup plus concise. Par exemple, le fait que k formules parmi n doivent être vraies se code en une seule équation ;
– elle permet facilement l'introduction de lemmes, c'est-à-dire de variables codant des formules ;
– elle permet de tirer partie facilement des symétries ou de toutes autres régularités apparaissant dans les problèmes, en les détectant automatiquement comme dans **[Benhamou & Saïs 1992]** ou en écrivant directement les formules de façon à en éliminer les redondances [23] ;
– elle permet de mettre en œuvre des procédures de réécriture des formules dans l'esprit du vérificateur de modèle INSTEP [Vlach 1993]. INSTEP implémente deux types de règles : les filtrages (ou simplifications immédiates) des formules et la décomposition de Shannon qui consiste à remplacer une formule $\mathcal{F}$ par la formule $(x \wedge \mathcal{F}_{x \leftarrow 1}) \vee (\neg x \wedge \mathcal{F}_{x \leftarrow 0})$.

Grâce à des structures de données adaptées, de nombreux algorithmes classiques (filtrages, énumération, recherche locale, diagramme binaires de décisions) ont été étendus sans difficultés particulières et surtout sans augmentation excessive de la complexité **[Rauzy *et al.* 1999]**.

Ce travail est un premier pas vers la mise en place d'un formalisme booléen plus général. Il reste beaucoup de travail à faire sur ce sujet, en particuliers sur :

– l'étude de techniques de simplification (pré-traitements) ;
– la mise en place d'heuristiques qui tiennent compte de la structure particulière du formalisme ;
– l'étude sur les aspects de compilation de formules booléennes (recherche de couverture d'impliquants (impliqués) premiers) ;
– l'extension des algorithmes pour résoudre des problèmes d'optimisation (ce que permettent les poids associés aux variables dans les équations).

23. Par exemple en imposant, dans un problème de Ramsey, des inéquations sur le nombre d'arêtes de chaque couleur.

2.4 Aspects algorithmiques

2.4.1 Recherche locale

Depuis le début des années 90, de nombreuses techniques à base de recherche locale ont été développées pour résoudre SAT. Mis à part le principe de base de l'algorithme, les stratégies utilisées pour échapper des minima locaux (e.g. « *break-out* » de MORIS[24] [Morris 1993], random walk[25] de SELMAN [Selman & Kautz 1993]) sont très différentes de celles utilisées en optimisation combinatoire. Cette constatation simple nous a conduits à étudier la possibilité d'adapter au cadre SAT des techniques de recherche locale développées en recherche opérationnelle (e.g. le recuit simulé, la méthode Tabou) et de les comparer aux autres approches à la GSAT. Par sa simplicité de mise en œuvre, notre choix s'est porté sur la méthode Tabou [Glover 1989]. La première version simple de la méthode Tabou que nous avons développée pour SAT (appelée TSAT) utilise une liste d'interdits de longueur fixe sans critère supplémentaire (e.g. critère d'aspiration) et s'est révélée plus efficace que les meilleures méthodes développées à ce moment là (GSAT random-Walk). La clé de l'efficacité de cette méthode réside dans le réglage de ses paramètres, c'est-à-dire la taille de la liste Tabou. Cette étude a donc permis de confirmer l'extrême sensibilité des techniques de recherche locale au réglage de leurs paramètres. Un autre résultat intéressant obtenu sur des instances aléatoires $k - SAT$, méritant une étude analytique approfondie, est celui montrant l'évolution linéaire de la taille optimale (obtenue empiriquement) de la liste tabou en fonction du nombre de variables. Nous avons aussi étudié l'influence de l'interprétation initiale sur l'efficacité de TSAT. Pour ce faire, l'interprétation initiale est générée en exploitant le déséquilibre entre les occurrences des variables. Des améliorations ont été obtenues sur les instances structurées. En revanche cette modification donne des résultats comparables à TSAT sur les instances aléatoires. Signalons que TSAT enlève le caractère aléatoire introduit dans GSAT, ce qui simplifie considérablement l'algorithme **[Mazure *et al.* 1997a]**.

L'efficacité de la méthode de recherche locale Tabou dépend considérablement de la taille de la liste d'interdits. Son réglage est difficile à obtenir particulièrement sur les instances structurées. Une voie que nous explorons actuellement concerne le réglage dynamique de ce paramètre crucial. Nous avons établi une relation entre la taille des cycles rencontrés au cours de la recherche et la longueur de la liste tabou. Des résultats prometteurs ont déjà été obtenus sur les instances aléatoires $k - SAT$. Notre approche exploite les informations liées à la présence et à la taille des cycles rencontrés au cours de la recherche. En effet, sur les instances $k - SAT$, la taille de la liste tabou converge vers la taille optimale obtenue précédemment. Malheureusement, ces informations restent insuffisantes pour établir le réglage optimal sur certains problèmes structurés.

2.4.2 Recherche systématique

2.4.2.1 Généralisation du théorème de partition du modèle

Comparativement aux améliorations obtenues par l'utilisation de filtrages plus puissants, d'heuristiques judicieuses, la recherche de propriétés permettant d'élaguer l'arbre de recherche a reçu moins d'attention.

Une des techniques les plus connue au moins dans la communauté française est sans doute le théorème de partition des modèles [Jeannicot *et al.* 1988, Oxusoff & Rauzy 1989]. Le théorème peut s'expri-

24. La méthode appelé Break-Out effectue une pondération des clauses. Cette pondération « poids » d'une clause est utilisée ensuite dans la fonction d'évaluation. Le poids d'une clause représente le nombre de fois qu'elle a été falsifiée depuis le début de la recherche.

25. Dans la stratégie Random-Walk, la prochaine variable à inverser est choisie parmi celles apparaissant dans les clauses falsifiées. Avec une probabilité p, celle améliorant le plus la fonction objective, et avec une probabilité $1 - p$ la variable est choisie au hasard.

mer comme suit : Si $\mathcal{F}_k \subset \mathcal{F}_j$, alors $\mathcal{F}_j$ est satisfiable ssi $\mathcal{F}_k$ est satisfiable ($\mathcal{F}_k$ et $\mathcal{F}_j$ étant des ensembles de clauses). Cette propriété a été intégrée dans une procédure DPL. Cette variante appelée « Evaluation Sémantique » a permis une amélioration sensible sur certaines classes de problèmes structurés.

Une généralisation immédiate consiste à remplacer l'inclusion syntaxique par la réciproque de la relation de déduction logique. Pour des raisons d'efficacité évidente, nous avons utilisé une relation plus faible (la déduction par propagation unitaire). Nous avons proposé deux façon d'intégrer cette notion dans la procédure DPL. Une première utilisation consiste à l'appliquer au moment du retour-arrière, la seconde avant le choix de la variable de branchement. Dans les deux cas, on peut intégrer la propriété de coupure dans un algorithme utilisant, soit les heuristiques traditionnelles, soit des heuristiques (du type de celles mises en place par OXUSOFF et RAUZY) spécifiques favorisant l'application du théorème. Une étude assez avancée autour de ces questions est actuellement en cours. Une autre généralisation consiste à associer (au cours de la recherche) pour chaque (sous-)clause de la base le niveau (ou profondeur de l'arbre) à partir duquel elle est apparue ou "touchée". Cet étiquetage des clauses de la formule au cours de la recherche permet de savoir à tout moment le niveau de l'arbre où une sous-clause a été touchée (apparue). Cette information est ensuite prise en compte lors de l'application du théorème. Plus précisément, lors de la production d'une sous-clause par propagation unitaire, nous avons montré qu'en analysant le graphe d'implications, on peut déterminer le niveau de l'arbre à partir duquel la sous-clause est produite. Ce qui nous permet de réaliser un retour-arrière non chronologique et de produire de nouvelles sous-clauses, importantes pour la suite de la recherche.

Ce travail a été développé en collaboration avec Éric GRÉGOIRE, Pierre MARQUIS et Bertrand MAZURE. Une description plus détaillée des différents points développées ci-dessus peut être trouvée dans **[Mazure 1999]**. La partie expérimentale est actuellement en plein développement (cf. chapitre 4).

2.4.2.2 Poids : apprentissage au cours de la recherche

De nombreuses techniques ont été proposées pour améliorer les algorithmes énumératifs en essayant d'analyser les causes d'échec. Elles font appel souvent à l'ajout de nouvelles clauses, ou à des techniques sophistiquées d'analyse de conflits, qui induisent souvent un sur-coût qui peut être non négligeable. En outre, la plupart des heuristiques traditionnellement développées utilisent des fonctions de sélection basées sur des paramètres syntaxiques (longueur des clauses, nombre d'occurrences, etc.). Par exemple, le score d'une variable peut être calculé comme suit : $F(x) = f(x) + f(\neg x)$, avec $f(x) = \sum_{x \in C} w(C)$, et $w(C) = 2^{|C|}$. De plus, la même heuristique est ensuite utilisée tout le long de la recherche, bien que les sous-problèmes rencontrés (formule simplifiée par l'interprétation partielle courante) au cours de la recherche n'ont pas forcément les mêmes caractéristiques que le problème initial.

Notre objectif, qui sans nul doute peut paraître ambitieux, concerne la mise au point d'une heuristique adaptative évoluant au cours de la recherche, en mettant à profit ces échecs répétés.

Ci-dessous quelques arguments expliquant les motivations de cette étude. On sait que prouver l'inconsistance d'une formule est souvent plus difficile que de prouver la consistance. De nombreuses instances inconsistantes difficiles contiennent une (des) sous-formules inconsistante(s). Une heuristique n'étant pas à l'abri d'une erreur dans le choix des variables de branchement, elle peut conduire à la résolution répétée du même sous-problème inconsistant (cf. section 2.4.3).

En d'autres termes, notre objectif est de mettre au point une heuristique permettant d'éviter de rencontrer les mêmes situations d'échec. L'heuristique que nous avons mise au point utilise un facteur de pondération supplémentaire pour les clauses, en comptabilisant au cours de la recherche le nombre de fois que chaque clause est à l'origine de la contradiction (ce qui n'est pas très coûteux en pratique). Plus précisément, on

obtient pour la fonction $F(x)$ décrite précédemment, une nouvelle fonction $F'(x) = f(x) + f(\neg x)$ avec $f(x) = \sum_{x \in C} \alpha(c) \times w(C)$ et $\alpha(C)$ le facteur supplémentaire ajouté, qui représente le nombre de fois que la clause C a été la cause de la contradiction.

L'augmentation du poids d'une clause revient à rendre cette clause redondante et de plus en plus importante. Cette information est directement prise en considération par la fonction de choix.

Les expérimentations réalisées sur la base d'instances (DIMACS) montrent clairement son efficacité par rapport à une version de base **[Brisoux *et al.* 1998, Brisoux *et al.* 1999]**. Sur certaines classes de problèmes, les résultats obtenus sont meilleurs que ceux des meilleurs algorithmes connus.

Poursuivant le même objectif, Bruni et Sassano[Bruni & Sassano 2000] ont présenté un papier au dernier Workshop SAT 2000, qui utilise une idée similaire.

2.4.3 Méthode mixte : coopération entre approche systématique et non systématique

La recherche locale ne peut prouver l'inconsistance. Sur une instance consistante de grande taille et non élémentaire, à cause de sa rigidité, une recherche systématique a peu de chance de trouver un modèle. Ces différentes techniques montrent donc une certaine complémentarité qui peut être illustrée par le problème suivant :
énoncé du problème : J'ai perdu une clé dans la maison et j'ai besoin de la trouver rapidement. Quel type de méthode devrais-je utiliser ?
solution possible : On peut procéder à une fouille systématique de toute la maison en commençant par exemple par le garage. Cependant, l'expérience montre que ce type de recherche est le dernier recours, une recherche non systématique consistant à suivre son instinct permet souvent de trouver plus rapidement l'objet recherché.

Il est donc naturel de penser à une approche permettant de faire coopérer fortement les deux types de techniques. La question est : comment les faire coopérer ?

La solution du problème précédent montre qu'il est judicieux de commencer par une recherche locale (incomplète) avant de décider de lancer une recherche systématique. Dan un premier schéma, la recherche locale est utilisée comme pré-traitement. Ce premier schéma pose la question suivante : est-il possible de bénéficier du travail effectué lors de la première phase, afin d'accélérer la seconde phase ? Une réponse possible consiste à éviter lors de la seconde phase de visiter autant que possible les endroits où la recherche a été infructueuse lors de la première phase.

Un second schéma possible consiste à combiner plus fortement les deux types d'algorithmes : appliquer localement le premier schéma au niveau de chaque pièce visitée. Dans cette seconde approche, si l'ensemble des pièces de la maison ont été visitées, on obtient une méthode complète.

Préalablement à la mise en œuvre de tels schémas de coopération dans le cadre de la résolution du problème SAT, il faut trouver des informations pertinentes à capitaliser lors du processus de recherche locale, pour fournir ensuite des indications susceptibles d'améliorer la phase de recherche systématique.

On peut être tenté de répondre que les informations en question sont en fait les interprétations visitées lors de la recherche locale. Il est bien évident que fournir un tel ensemble n'est pas réaliste. Cependant, on sait qu'une clause C est falsifiée par $2^{n-|C|}$ interprétations possibles, on peut par conséquent relever que plus une clause est falsifiée au cours de la recherche locale, plus l'espace des interprétations falsifiant une

telle clause tend à être exploré. Cette information intéressante est plus facile à exploiter. En effet, on peut au cours de la recherche systématique essayer de satisfaire de telles clauses et ainsi éviter (partiellement) les zones déjà explorées par l'algorithme de recherche locale.

Un des objectifs de cette étude est la mise au point de techniques permettant de prouver efficacement différentes formes d'inconsistance, un problème central en représentation des connaissances et en démonstration automatique. En effet, l'inconsistance d'une base de connaissances est très fréquemment due à la présence accidentelle d'une information contradictoire sur un sujet donné. Cette inconsistance pollue globalement la base de connaissances (i.e. toute information ainsi que son contraire peuvent être inférés de cette base). Malheureusement, il n'existe pas de méthode universelle et efficace pour détecter et/ou localiser ces inconsistances.

Une technique de recherche locale (incomplète) ne peut prouver l'inconsistance. Cependant, elle peut fournir des indications pertinentes sur de telles inconsistances locales. En effet, en comptabilisant le nombre de fois que chaque clause est falsifiée au cours de la phase de recherche locale, on s'est aperçu expérimentalement (sur de nombreux problèmes de DIMACS [Dimacs 1993]) que les clauses les plus souvent falsifiées sont les plus susceptibles d'appartenir aux sous-ensembles minimalement inconsistants. Ce qui nous a permis de mettre en évidence une technique capable de localiser l'inconsistance. Le nombre de fois qu'une clause est falsifiée (« trace ») lors de la recherche locale constitue une information pertinente que nous avons utilisée dans le cadre des deux schémas de coopération précédents. Dans le premier schéma, cette trace est exploitée pour trouver un bon ordonnancement des variables pour la méthode complète. Dans le second schéma, nous avons utilisé dans la procédure DPL la recherche locale comme un oracle, qui permet non seulement d'étendre l'interprétation partielle courante, mais également de fournir en cas d'échec la prochaine variable à affecter (celle apparaissant le plus souvent dans les clauses falsifiées). Notons que les deux schémas proposés sont complets. Des améliorations surprenantes ont été obtenues sur de nombreux problèmes de la base de test de DIMACS **[Mazure 1995, Mazure *et al.* 1996, Mazure *et al.* 1998]**. D'autres schémas de coopération entre les algorithmes de recherche locale et les algorithmes complets de recherche systématique sont présentés dans **[Mazure 1999]**.

Il est évident que les problèmes les plus difficiles présentent une inconsistance globale (l'inconsistance est due à l'ensemble du problème au lieu d'un sous-ensemble de celui-ci) ; ces derniers sont généralement construits de manière artificielle ou aléatoire et ne se retrouvent pas dans les applications réelles. De plus, ce genre d'instances possèdent le plus souvent de très fortes régularités (problèmes des pigeons, etc.). Une étude comparative avec l'approche utilisant la résolution bornée comme prétraitement a été réalisée **[Mazure *et al.* 1997a]**. Ces travaux ont été réalisés en collaboration avec Éric GRÉGOIRE et Bertrand MAZURE.

D'autres types d'approches complètes utilisant la recherche locale ont été développées. On peut citer dans ce cadre l'algorithme SCORE [Chabrier *et al.* 1991], et SCORE(FD/B) [Chabrier *et al.* 1995]. D'autres algorithmes complets à base de recherche locale ont été obtenus par ajout de clauses [Ginsberg 1993, Hao & Tétard 1996]. Des approches similaires au premier schéma de coopération décrit précédemment ont été étudiées indépendamment par Thierry CASTELL [Castell 1997] et James M. CRAWFORD [Crawford 1993]. Un inventaire complet des méthodes mixtes dans le cadre des problèmes de satisfaction de contraintes est donné dans [Lobjois & Lemaître 1997].

2.5 Phénomène de seuil et génération aléatoire

Dès lors que l'intérêt de la génération aléatoire d'instances SAT est avéré, d'une part par le résultat de Chvàtal et Szemerédi montrant qu'elles sont difficiles pour la résolution et donc pour toute les méthodes énumératives du type procédure de Davis et Putnam, d'autre part par la mise en évidence de phénomènes

de seuil, plusieurs questions se posent :

1. Est-il possible de mettre en évidence l'existence de conditions nécessaires et/ou suffisantes à la caractérisation des ensembles de clauses encore plus difficiles ? En utilisant, par exemple, des conditions nécessaires à la présence de symétries ?
2. Les instances générées par le modèle standard contiennent environ 50% de clauses de Horn. Que se passe-t-il, si l'on fait varier la probabilité de négativer un littéral de 0.5 à 1 ?
3. Est-il possible de générer aléatoirement des instances difficiles toujours satisfiable ?
4. Est-il possible d'établir des modèles de génération aléatoire donnant lieu à des instances plus proches des problèmes réels ?

Avec Richard GÉNISSON, nous nous sommes particulièrement intéressés aux deux premières questions et plus accessoirement à la troisième.

Mes travaux de recherche précédents sur les symétries ont conforté ma conviction que les problèmes les plus difficiles présentent une structure très symétrique. Ce qui nous a conduits à répondre à la première question, en étudiant deux nouveaux modèles de génération d'ensembles de clauses régulières (toutes les variables apparaissent le même nombre de fois, les clauses ayant toutes la même longueur). Dans le premier modèle, tous les littéraux ont exactement le même nombre d'occurrences ; dans le second, les variables apparaissent le même nombre de fois, avec un déséquilibre entre le nombre d'occurrences des littéraux positifs et négatifs. Dans ces deux modèles, des conditions nécessaires pour l'existence de symétrie sont vérifiées. Pour ces deux modèles, nous avons montré expérimentalement que le seuil se situe pour 3-SAT au rapport $n/m = 3.6$ (chaque variables apparaît 6 fois positivement et 5 fois négativement). Les instances obtenues sont beaucoup plus difficiles que pour le modèle standard. Les instances satisfiables posent plus de problèmes aux algorithmes de recherche locale. D'autre part, l'étude que j'ai réalisée sur le modèle de génération standard en variant la probabilité de tirer un littéral négatif, nous a conduits à établir des liens entre la valeur du seuil et la distribution des occurrences des variables (déséquilibre entre littéraux positifs et négatifs, proportion de clauses de Horn). Ces différents résultats sont publiés dans **[Génisson & Saïs 1994, Mazure 1999]**.

D'autres auteurs ont étudié ensuite le modèle de génération d'ensembles de clauses régulières [Bayardo & Schrag 1996, Dubois & Boufkhad 1996]. On peut également trouver des générateurs aléatoires d'instances difficiles pour la recherche locale [Castell & Cayrol 1997, Rauzy 1995a, Achlioptas *et al.* 2000] et des tentatives de prise en compte de certaines caractéristiques de problèmes réels dans la génération aléatoire [Gent & Walsh 1994].

Chapitre 3

Problèmes autour de SAT

Sommaire

3.1 Problèmes de satisfaction de contraintes 29
3.2 Compilation des bases de connaissances et représentation compacte de l'ensemble des solutions 31
3.3 Des techniques efficaces pour SAT au secours de la révision de croyances 32
3.4 Une méthode complète pour des bases de connaissances propositionnelles non monotones 32
3.5 Coopération consistante et non-monotonie 33
3.6 Premier ordre fini 33

3.1 Problèmes de satisfaction de contraintes

Les logiciels commerciaux tels que « *Ilog Solver* » et les outils pour la recherche et l'enseignement comme « *CHOCO* » montrent la maturité des recherches conduites dans la cadre de la résolution des problèmes de satisfaction de contraintes. Le formalisme et le raisonnement par contraintes offrent un outil de modélisation et de résolution majeur pour de nombreuses applications aussi diverses que le traitement du langage naturel, la gestion et l'allocation de ressources, les emplois du temps, la vision, etc. Un réseau de contrainte est défini par un ensemble de n variables $X = \{x_1, x_2, \ldots, x_n\}$ prenant leurs valeurs dans des domaines finis et discrets $D = \{d_1, d_2, \ldots, d_n\}$. Le problème de satisfaction de contraintes consiste à décider s'il existe une affectation de valeurs aux variables de manière à satisfaire un ensemble de contraintes $C = \{c_1, c_2, \ldots, c_m\}$. Une contrainte c_i est définie sur un sous-ensemble de variables $c_i = \{x_{i1}, x_{i2}, \ldots, x_{in_i}\}$ de X, où n_i est l'arité de c_i. Une telle contrainte peut être spécifiée (en extension) par une relation $r_i = \prod_{x_i \in c_i} d_i$ des combinaisons de valeurs (ou tuples) autorisées pour les variables de c_i. Ces contraintes peuvent aussi être spécifiées en intension par une fonction caractéristique (prédicat, fonction, etc). Satisfaire une contrainte c_i revient à trouver les valeurs à affecter pour chaque variable de c_i de manière à satisfaire la relation r_i associée à la contrainte.

Le problème de satisfaction de contraintes (CSP) est donc une généralisation du problème SAT. En effet, toute instance de SAT peut être vue comme une instance de CSP dans laquelle toutes les variables ont pour domaine $\{0, 1\}$, et il y a autant de contraintes que de clauses. Toute instance de CSP peut également être transformée en temps polynômiale en une instances de SAT équivalente pour l'existence de solution (les deux problèmes sont NP-complets). Les CSP binaires (où toute les contraintes ont une arité égale à 2)

ont été les plus étudiés. Leur représentation en terme de graphes de contraintes (au lieu d'hypergraphes de contraintes pour les CSP n-aires) simplifie considérablement les algorithmes.

La description ci-dessus donne un aperçu des liens étroits entre SAT et CSP. Ce qui explique certaines similitudes entre les deux problèmes à la fois dans les questions posées, mais également dans les approches utilisées pour y répondre.

En effet, on retrouve par exemple de nombreux travaux traitant de l'amélioration de la procédure de résolution classique appelée généralement « *Backtrack* » (BT). Pour l'essentiel, ces différentes améliorations ont concerné les techniques de filtrage, les heuristiques de choix de variables et de valeurs et le traitement des échecs. Pour ce qui est des aspects algorithmiques, le meilleur algorithme de résolution de CSP (MAC, pour « *Maintaining Arc Consistency* ») est obtenu en maintenant dynamiquement dans « *Forward Checking* » (FC), un CSP arc consistant [1] à chaque nœud de l'arbre de recherche [Sabin & Freuder 1994]. L'heuristique proposée (minimum domaine/degré [2]) par Régin et Bessière [Bessière & Régin 1996] améliore considérablement l'algorithme. D'autres approches sont basées sur la décomposition du CSP. Ces techniques exploitent les propriétés structurelles des CSP [Dechter & Pearl 1989, Dechter 1990, Jégou 1993]. En outre, les propriétés structurelles du graphe de contraintes (e.g. largeur du graphe) et la sémantique des contraintes (contraintes fonctionnelles, etc.) sont à la base de la plupart des classes polynômiales exhibées dans le cadre CSP. Par exemple, la consistance d'arc (AC) suffit pour résoudre des CSP structurés en arbre, alors que les CSP 0/1/all [Cooper *et al.* 1994] peuvent être résolus polynômialement en utilisant la consistance de chemin (PC).

De nombreux travaux ont concerné l'étude des liens entre SAT et CSP. Dans le cadre du projet BAHIA **[Bahia 1995, Bahia 1992]**, on peut citer par exemple, le résultat de GÉNISSON et JÉGOU montrant l'équivalence entre la procédure de Davis et Putnam et l'algorithme de « *Forward Cheking* » (FC) [Génisson & Jégou 1996], les pontages entre les classes polynômiales [Génisson & Rauzy 1996], l'équivalence entre les littéraux purs et la substituabilité de voisinage [Bellicha *et al.* 1994] et par conséquent l'extension du théorème de partition du modèle au CSP ou encore le formalisme des CSP clausaux proposé dans [Castell & Fargier 1998], etc.

Avec Assef CHMEISS **[Chmeiss & Saïs 2000]**, nous nous sommes intéressés à l'étude des possibilités d'intégration forte des traitements locaux dans le cadre de la résolution de CSP. Ce travail a été essentiellement motivé par les améliorations algorithmiques obtenues dans le cadre SAT en utilisant d'une part des traitements locaux de manière plus agressive comme la propagation unitaire, et d'autre part par l'utilisation de techniques de simplification plus puissantes comme la résolution bornée. La puissance des machines fait qu'on n'hésite même plus à revenir aux vielles méthodes comme la résolution et la procédure de Davis et Putnam originale, longtemps écartée pour des raisons de complexité en espace (cf. chapitre 2).

Ce dernier point nous conduit à faire le parallèle avec la consistance de chemin qui peut être vue comme un processus de production et de vérification de contraintes. Par ailleurs, pour les mêmes raisons que pour la résolution, l'intégration de la consistance de chemin dans les algorithmes de résolution de CSP a été jusqu'à présent évitée. En effet, la complexité élevée du test de consistance de chemin (comparativement à la consistance d'arc) constitue un obstacle à sa mise en œuvre en pratique. Pour pallier cet inconvénient, nous avons proposé une exploitation partielle de cette forme de filtrage au cours de la résolution. La consistance de chemin (directionnelle) est utilisée au cours de la recherche pour effectuer des disconnections de certaines variables particulières : les variables « doubletons » (variables de degré 2). Nous avons intégré ce filtrage dans MAC en exploitant également les singletons degrés et les singletons valeurs. L'algorithme obtenu est appelé M(AC+).

1. Un CSP est k-consistent si et seulement si toute instanciation partielle de $k-1$ variables peut être étendue de manière consistante à toute $k^{ième}$ variable. La consistance d'arc (resp. la consistance de chemin) correspond à la 2-consistance (resp. 3-consistance).

2. Min dom/deg : consiste à choisir la variable maximisant le rapport entre la taille du domaine et le degré de la variable. Le degré d'une variable est égale au nombre de contraintes où elle apparaît.

Une autre technique de filtrage plus puissante que la consistance d'arc et qui nous semble particulièrement prometteuse pour aller plus loin dans la résolution de problèmes difficiles est la "singleton arc consistance" (SAC) proposée par DEBRUYNE et BESSIÈRE [Debruyne & Bessière 1997]. Plus précisément, en plus des possibilités de filtrage par SAC, nous montrons qu'il est possible de déduire des informations pertinentes supplémentaires, en analysant le processus de propagation. Nous proposons également différentes façons de l'utiliser en cours de résolution, par exemple comme moyen de raffiner les heuristiques de choix de variables.

Il reste beaucoup de travail à faire sur ce sujet. Une collaboration est actuellement en cours avec Christian BESSIÈRE (CR CNRS, LIRMM Montpellier).

3.2 Compilation des bases de connaissances et représentation compacte de l'ensemble des solutions

Afin de pallier la non-calculabilité pratique (« *intractability* ») de la déduction logique en calcul propositionnel, plusieurs approches ont été proposées. Elles s'appuient sur différents principes, en particulier :

- restriction (du langage utilisé, e.g. les clauses de Horn, ou de la relation de déduction employée, e.g. la résolution bornée) ;
- approximation (de la base de connaissances ou de la relation de déduction) ;
- compilation. L'idée est de construire une structure de données σ^* (une compilation de la base initiale σ) qui se comporte comme σ pour l'interrogation mais nécessite seulement un temps polynomial (en fonction de $|\ \sigma^*\ |$).

En collaboration avec Yacine BOUFKHAD, Éric GRÉGOIRE, Pierre MARQUIS et Bertrand MAZURE, nous nous sommes particulièrement intéressés à ce dernier principe. Nous avons proposé une nouvelle approche de compilation préservant l'équivalence logique avec la base de connaissances initiale et appelée couverture tractable (une couverture tractable est une disjonction de formules tractables). Cette nouvelle approche a été formalisée dans un cadre général. Deux cas spécifiques ont été considérés :

1. les interprétations partielles sont utilisées pour « raboter » ou simplifier la base de connaissances en un ensemble de formules traitables (simplifications traitables).
2. les interprétations partielles sont utilisées de manière à dériver un ensemble de formules Horn- renommables (hyper-impliquants).

Dans le second cas, nous avons prouvé que la taille de la couverture est strictement inférieure à celle d'une couverture par impliquants premiers. Il faut noter qu'en pratique les formules tractables ne sont pas représentées, seules les interprétations partielles (obtenues par la procédure de Davis et Putnam) nous permettant de les obtenir sont réellement sauvegardées. Des résultats expérimentaux ont montré l'efficacité d'une telle approche par rapport à l'interrogation directe de la base de connaissances. Un gain exponentiel en temps et en espace est obtenu par rapport à l'approche par impliquants premiers [Castell & Cayrol 1996, Schrag 1996]. Ce travail a été publié dans **[Boufkhad *et al.* 1997]**.

Dans les approches présentées ci-dessus, le raisonnement modulo une théorie [Marquis 1995] a été utilisé lors de la construction de la base compilée dans l'objectif de réduire la taille de cette dernière. C'est cette notion, qui a rendu possible le gain en temps et en espace par rapport aux approches par impliquants premiers. Pour de nombreux problèmes et en particulier les problèmes sous-contraints, la taille de la compilation peut être prohibitive et donc difficile à construire et à représenter. Trouver une représentation plus

compacte de la formule compilée présente par conséquent un intérêt pratique indéniable pour de nombreuses applications.

Dans un cadre restreint aux formules contenant des symétries, je me suis intéressé au problème du calcul d'une représentation plus compacte de l'ensemble des solutions. En effet, pour ces formules, l'ensemble des solutions peut être partitionné en classes d'équivalence. L'utilisation de ces symétries permet de représenter l'ensemble des solutions par un sous-ensemble de solutions caractéristiques (solutions non symétriques). Une étude détaillée a été publiée dans **[Saïs 1994a, Saïs 1994c]**.

3.3 Des techniques efficaces pour SAT au secours de la révision de croyances

De nombreuses approches de la révision de croyances ont été proposées cette dernière décennie. La plupart d'entre elles ont été étudiées principalement au niveau dit de la connaissance [Newell 1982]. En dépit de quelques exceptions notables (e.g. [Papini 1991, Nebel 1992, Liberatore & Schaerf 1996]), elles n'ont pas été implantées, pour des raisons évidentes de complexité dans le pire des cas. Cependant, de récents progrès dans la résolution pratique du problème SAT ouvrent des perspectives de calculs effectifs pour les approches syntaxiques de la révision de croyances (voir e.g. [del Val 1994, Makinson & Gärdenfors 1991, Moinard 1994, Zhang *et al.* 1997]). En collaboration avec Éric GRÉGOIRE, Pierre MARQUIS et Brigitte BESSANT, nous avons proposé une nouvelle approche de la révision syntaxique des croyances dans un cadre non monotone. Motivée par des considérations pratiques cette approche est basée sur une approximation de la réunion des sous-ensemble minimaux inconsistants obtenus grâce à l'application des techniques complètes à base de recherche locale. Cette approche comporte un processus de résolution d'inconsistance en deux étapes. En premier lieu, une utilisation disciplinée d'ingrédients non monotones (des propositions d'anormalité qui permettent de représenter des règles de défaut et des exceptions) est offerte à l'ingénieur de la connaissance pour éviter de nombreuses inconsistances qui apparaîtraient si une représentation et une interprétation logique standard des croyances étaient réalisées. Les inconsistances restantes sont supposées imprévisibles et révisées par affaiblissement de formules apparaissant dans toute sous-base minimalement inconsistante, comme si elles représentaient des cas exceptionnels qui en fait n'ont pas lieu. Ce dernier point nous a conduit a utiliser les mêmes ingrédients que dans la première étape, pour affaiblir et rétablir la consistance. L'affaiblissement d'une formule est obtenue par ajout disjonctif de nouvelles propositions ; ceci offre des perspectives intéressantes en ce qui concerne les révisions itérées.

Alors que le calcul de bases de connaissances révisées demeure intraitable dans le pire des cas, notre approche bénéficie d'une technique heuristique efficace de recherche locale qui est souvent viable en pratique. Dans le même état d'esprit, des techniques de recherche locale spécifiquement adaptées pour MAX-SAT sont étudiées pour implanter une politique de révision alternative qui minimise le nombre de formules à affaiblir afin de rétablir la consistance **[Bessant *et al.* 2000, Bessant *et al.* 1998]**.

3.4 Une méthode complète pour des bases de connaissances propositionnelles non monotones

Nous avons proposé une nouvelle approche pour calculer de façon effective des inférences au sein d'une logique propositionnelle non monotone simple, fondée sur un concept de modèles préférés. L'approche que nous proposons est originale à deux égards au moins. Premièrement, elle se base sur des techniques de recherche locale tout en préservant la complétude logique. En second lieu, elle est efficace d'un point de vue

expérimental pour une classe importante de grandes bases de connaissances non monotones. Plus précisément, nous étendons à un cadre non monotone des résultats heuristiques récents liés à la résolution pratique du problème de satisfiabilité d'un ensemble de clauses (SAT). De fait, la procédure de preuve employée se base sur l'utilisation d'une méthode de recherche locale pour SAT et sur une heuristique efficace lorsque cette recherche locale n'a pu exhiber un modèle. Nous l'employons au sein d'un formalisme de représentation permettant des règles de raisonnement par défaut avec priorités exprimées à l'aide de propositions d'anormalités à la McCarthy. Un domaine d'applications typique concerne les formes de raisonnement révisable qui peuvent être tenues au sujet d'un modèle « profond » d'un système ou d'un appareillage complexe, où les propositions d'anormalité sont utilisées pour représenter les défaillances possibles de composants et où un nombre limité de pannes peuvent survenir simultanément **[Grégoire *et al.* 1998]**.

3.5 Coopération consistante et non-monotonie

Notre expérience dans le cadre de la détection et de la localisation des inconsistances nous a conduits à envisager l'extension et l'application de ces techniques pour résoudre les problèmes liés à la fusion et à l'interaction des bases de connaissances dans un cadre de travail coopératif. En effet, le problème de maintien de la consistance lorsque l'on fait interagir différentes sources d'informations est fondamental dans de nombreuses applications : bases de données distribuées, systèmes multi-agents, travail coopératif, etc. Dans ce cadre, nous avons montré que les techniques que nous avons développées dans le cadre du problème SAT peuvent être utilisées pour localiser les inconsistances qui peuvent apparaître lors de la fusion de bases de connaissance. Cette étude a été réalisé dans le cadre du projet inscrit au sein du contrat de plan État Région « Ganymede : Travail coopératif et communications avancés ». Une partie de cette étude est publiée dans le journal « Int. Journal of Cooperative Information Systems » **[Mazure *et al.* 1997b]**.

3.6 Premier ordre fini

Les remarquables progrès obtenus dans la résolution du problème SAT conduisent naturellement à l'extension et l'application de ces techniques dans le cadre de la logique du premier ordre. Une approche naïve consiste à appliquer les algorithmes de SAT sur une instanciation complète de la formule du premier ordre. Cette méthode peut provoquer une explosion combinatoire. En effet, le nombre d'instanciations possibles peut être extrêmement large et même infini. Dans ce cadre, on peut citer par exemple la méthode « *ModGen* » développée par S. KIM et H. ZHANG [Kim & Zhang 1994] pour résoudre des formules du premier ordre fini. Cette technique se compose de deux modules, le premier effectue une transformation de la formule du premier ordre en formule propositionnelle sous forme CNF ; le second module utilisant les techniques de SAT pour trouver un modèle de la formule. Dans le cadre du premier ordre fini restreint aux interprétations de Herbrand, nous nous sommes intéressés au problème suivant : soit BC une base de connaissances consistante et f une formule, prouver la consistance de $BC \cup f$. Nous avons proposé une technique efficace qui effectue une instanciation partielle et incrémentale de la base à partir des prédicats de f. Cela permet d'éviter dans de nombreuses situations d'instancier complètement la base. En d'autres termes l'approche que nous avons proposée se limite à l'instanciation d'un sous-ensemble de BC en relation avec f (partageant des prédicats). La résolution se fait également de manière incrémentale au fur et à mesure du processus d'instanciation. Ces travaux ont été réalisés en collaboration avec Éric GRÉGOIRE et Laure BRISOUX **[Brisoux *et al.* 1998, Brisoux *et al.* 2000]**.

Chapitre 4

Perspectives

Sommaire

4.1 Problème SAT 34
4.1.1 Approches traditionnelles 35
4.1.2 Autres approches 38
4.2 Autour de SAT 39
4.2.1 Problèmes de satisfaction de contraintes 39
4.2.2 Procédures de preuve pour les logiques non standard 41
4.3 Applications 42

Dans ce chapitre je présente mes perspectives de recherche à court et à moyen terme. Les pistes et les détails techniques donnés dans certains cas montrent l'état de maturité et le caractère à court terme de certaines de ces perspectives.

4.1 Problème SAT

Comme dans tout domaine de recherche, les résultats obtenus montrent à la fois les progrès mais également les limites des approches utilisées. Il est plus facile de continuer à améliorer et à étendre les approches traditionnellement utilisées que de trouver de nouvelles pistes originales qui permettront éventuellement d'aller plus loin et de répondre aux limites des approches traditionnelles.

Sans prétendre à la paternité exclusive de ces pistes, je présente mes perspectives de recherche sur SAT en deux axes :

– l'extension des approches traditionnelles ;
– l'étude d'autres approches.

4.1.1 Approches traditionnelles

4.1.1.1 Aspects algorithmiques

Parmi les algorithmes de recherche systématiques à la Davis et Putnam, on peut grossièrement distinguer deux catégories d'algorithmes :

- les algorithmes prospectifs : ils obtiennent leurs meilleures performances sur les problèmes aléatoires difficiles au seuil (e.g. CSAT, POSIT, Sato, Satz, CSAT, Tableau, etc.).
- les algorithmes de type rétrospectif : il faut plutôt les utiliser pour résoudre des problèmes réels ou structurés (e.g. Relsat, Grasp, etc.).

Raisonner : Pour résoudre des problèmes réels et des applications, il me paraît primordial de s'orienter vers le développement de méthodes intégrant des formes de raisonnement (e.g. utilisation de lemmes, de règles de déduction). L'analyse des échecs est une forme de raisonnement. En effet, à l'opposé des problèmes aléatoires, les instances codant des applications réelles présentent des structures particulières sur lesquelles il faut raisonner. En d'autres termes, éviter dans ces cas là d'utiliser des algorithmes dits de « force brute ». Je pense qu'à l'avenir on trouvera plus d'algorithmes appartenant à la seconde catégorie ou encore des techniques intégrant avantageusement les deux principes.

Spécialiser : On sait que chaque algorithme admet ses formules privilégiées, et ses formules difficiles. Il semble impossible de mettre au point un algorithme qui obtiendrait les meilleures performances sur tous les problèmes. Il me paraît plus raisonnable d'essayer de développer un algorithme qui résout efficacement une classe de formules. Une technique qui résout efficacement les problèmes de circuits par exemple, mais qui peut présenter un comportement catastrophique sur d'autres types de formules peut rester pour moi un bon algorithme. Les compétitions organisées dans le cadre SAT (DIMACS et Beijin) ont largement contribué aux développement des recherches. En plus du comportement général d'un algorithme, il me paraît nécessaire de prendre en considération le comportement spécifique d'un algorithme (e.g. quelle est la meilleure méthode disponible pour résoudre des problèmes de planification ?).

Après la découverte du phénomène de seuil, Olivier DUBOIS proposait l'idée de spécialiser les algorithmes soit dans la recherche d'une solution, soit dans la recherche d'une contradiction. En effet, un algorithme peut avoir des comportements très différents selon que l'on suppose qu'il existe ou non une solution. Cette idée intéressante ne me paraît que très partiellement explorée et nécessite de l'être systématiquement. Le challenge 1. proposé dans [Selman *et al.* 1997] concerne la résolution d'instances aléatoires insatisfiables de 700 variables au seuil par une méthode complète. Cette question est très difficile (voir "autres approches"). En revanche, résoudre des problèmes satisfiables de 700 variables au seuil par des techniques complètes (spécialisées dans la recherche de solutions) me paraît dans une certaine mesure plus facile (SAT est NP-complet et UNSAT est Co-NP-Complet). Elle présente plusieurs intérêts : elle permettra de se rapprocher plus des performances de la recherche locale et d'avoir indirectement des intuitions plus précises sur l'efficacité remarquable de la recherche locale sur de telles instances, etc. De plus, cette question est loin d'être évidente. Des travaux sont actuellement en cours dans cette direction.

Combiner : La mise au point de méthodes hybrides (en combinant différents algorithmes) est un thème de recherche actuel en optimisation combinatoire et en programmation logique par contraintes.

Nous avons montré dans le cadre de nos travaux précédents qu'il est possible de combiner avantageuse-

ment différentes approches (recherche locale et recherche systématique). Dans ce cadre, nous envisageons de continuer à explorer cette voie qui s'est déjà révélée viable. En particulier, proposer d'autres schémas de combinaison ou encore exploiter d'autre types d'informations. Par exemple, dans de nombreuses situations la recherche locale peut trouver une interprétation qui satisfait toute les clauses sauf une (« Critical Truth Assignment » ; notion caractérisant de nombreux problèmes difficiles). Cette clause particulière non satisfaite est à prendre en compte dans l'algorithme d'énumération. Le choix des algorithmes à combiner est aussi une question importante. Nous signalons que combiner efficacement les techniques de recherche locale et systématique correspond au challenge 7 proposé par SELMAN et al. [Selman *et al.* 1997].

La résolution a été pendant longtemps écartée pour des raisons de complexité spatiale (et donc temporelle). Les travaux récents de Laurent Simon et Philippe Chatalic [Chatalic & Simon 2000] et de Dechter [Dechter & Rish 1994] sur la procédure originale de Davis et Putnam (1960), montrent clairement qu'on a tout intérêt à revisiter ce type d'approches. On peut également noter que de nombreuses améliorations ont été obtenues grâce à l'utilisation de différentes formes limitées de résolution dans la procédure DPL (cf. section 2). Dans ce cadre, il me semble intéressant d'essayer de combiner plus fortement la résolution et l'énumération (DPL). Par exemple, on peut tenter de bénéficier des résultats obtenus en logique du premier ordre sur les stratégies de résolution (e.g résolution linéaire, SL-Résolution, etc.).

Pour réaliser une telle intégration, plusieurs questions se posent :

1. Quelle forme de résolution utiliser ? (e.g. résolution bornée, saturation sur les clauses binaires, résolution linéaire, etc.).
2. Quelles informations peut-on déduire ? (e.g. taille des résolvantes, leurs caractéristiques, etc.) et surtout quelles sont les plus intéressantes ? (e.g. binaires, tautologiques [1], etc.).
3. Quand et comment l'utiliser ? (e.g. de manière incrémentale, sur les clauses impliquant la variable de branchement, en limitant le nombre de résolvantes).

En outre, en collaboration avec Laure BRISOUX et Éric GRÉGOIRE, une étude préliminaire a été réalisée autour de l'hybridation de différentes techniques de recherche locale [Brisoux-Devendeville 1999]. Des résultats encourageant ont été obtenus grâce à une intégration de deux techniques de recherche locale aux comportements complémentaires (WSAT [Selman & Kautz 1993] et « *Break-Out method* » de MORISS [Morris 1993]).

Enfin, les résultats récents obtenus par GOMES et al. [Gomes & Selman 1997, Gomes *et al.* 1998] confirment l'importance de ce thème. Ces travaux ont été motivés par une observation simple liée au comportement des différentes techniques de résolution. En effet, sur de nombreux problèmes réels (e.g. planification, ordonnancement), les différentes méthodes de recherche présentent des variations importantes en terme de performances. Autrement dit, des différences significatives en temps d'exécution peuvent être constatées selon que l'on utilise différentes heuristiques sur le même problème, la même heuristique sur différents problèmes, ou encore différentes exécutions d'une même technique de recherche locale. Pour minimiser ce phénomène et améliorer la résolution de ce type de problèmes, ils proposent deux approches, la première est basée sur une intégration efficace de différents algorithmes (appelée « *Algorithm Portfolio Design* ») [Gomes & Selman 1997], la seconde est basée sur une exécution répétée d'un algorithme de recherche systématique, chaque exécution étant bornée par un nombre fixé de retour-arrière (appelé "cutoff") [Gomes *et al.* 1998]. Pour cette dernière approche, une perspective intéressante consiste à étudier les possibilités d'intégration de l'apprentissage des échecs (cf. section 2.4.2.2) pour améliorer les différentes étapes de résolution (les exécutions répétées).

Implémenter efficacement : La résolution de problèmes par ordinateur nécessite non seulement la mise au point d'un algorithme mais également le choix de structures de données permettant de coder efficacement

1. Une résolvante tautologique entre deux clauses binaires traduit une équivalence ou un ou-exclusif entre les deux variables.

les données. Une structure de données est efficace si elle permet de réduire (au moins en pratique) le coût des traitements effectués sur une telle structure. Sans minimiser la portée des heuristiques et des traitement additionnels, le choix d'une bonne structure de données constitue un point important dans la réalisation d'une implémentation efficace.

En outre, il suffit de parcourir les sources des meilleurs implémentations actuelles pour s'apercevoir des nombreuses astuces (certaines sont très discutables) utilisées [2] :

- réduire au maximum le coût des mises à jour en utilisant des structures de données supplémentaires (la complexité en espace étant un paramètre souvent négligé),
- retarder les modifications jusqu'au dernier moment,
- utiliser la programmation par effet de bord et les registres pour les variables locales les plus utilisées,
- privilégier la non-lisibilité du programme en utilisant des instructions les plus proches de la machine (e.g. opérations sur les bits),
- minimiser les allocations mémoires (utilisation de tableaux statiques),
- etc.

Toutes les implémentations que nous avons réalisées jusqu'à présent ne sont aucunement optimisées et n'utilisent pas ces différentes astuces de programmation. Dans ce contexte, la comparaison des méthodes devient malheureusement problématique sachant que le chronomètre est considéré comme l'arbitre suprême. De plus l'implémentation des différents algorithmes dans une même plate-forme est très difficile voire impossible à réaliser. Ce qui nous a conduits avec Bertrand MAZURE à développer une première version de DPL utilisant des structures de données efficaces et différentes astuces que nous avons jugées intéressantes. Nous avons développé une heuristique nouvelle générique et une étape de pré-traitement par résolution bornée très efficace. La prochaine étape consiste à intégrer les différents travaux précédemment réalisés ou en cours de réalisations.

4.1.1.2 Fonctions booléennes générales et propriétés structurelles

Pour se rendre compte des limites de la forme CNF, il suffit de regarder la structure des formules codant les problèmes réels. Je citerai par exemple les instances de circuits proposées par Joao SILVA [Silva & e Silva 1999, Silva & Sakallah 2000], les instances de model checking de A. BIERE [Biere *et al.* 1999], les problèmes de cryptographie de Massacci [Massacci 1999] (ces instances peuvent être téléchargées sur le site de SATLIB [3]). Le comportement des algorithmes développés sur SAT est assez décevant (en particulier la première classe d'algorithmes : les approches prospectives). Ce comportement s'explique en partie par la non prise en compte de la structure particulière de ces problèmes. En effet, de nombreuses clauses ne sont que l'expression d'une formule booléenne générale de la forme $y = f(x_1, x_2, \ldots, x_n)$. Cette constatation simple conduit à reconsidérer le formalisme étendu (cf. section 2.3.2) pour résoudre ce type de problèmes, en ne considérant que des contraintes de la forme $y = f(x_1, \ldots, x_n)$, où f est un connecteur logique ($\wedge$, $\vee$, $\Leftrightarrow$) et éventuellement des formules de cardinalité simple. Le but est d'essayer de mettre en œuvre des méthodes capables d'exploiter une telle structure. Il ne s'agit pas ici d'étendre les algorithmes classiques, mais de trouver des algorithmes plus adaptés et des techniques de simplification spécifiques.

2. Dans la description de Satz, ANBULAGAN affirme que l'utilisation des types « char » et « unsigned char » permet d'accélérer Satz de 25% [Anbulagan 1998] !

3. http://www.informatik.tu-darmstadt.de/AI/SATLIB

4.1.1.3 Codage

Améliorer la phase de codage de problèmes réels est une étape indispensable à la résolution efficace du problème proprement dit (voir par exemple, le challenge 8 proposé par Selman [Selman *et al.* 1997] et le thème 1 du GT 1.2 proposé par Daniel LEBERRE et Michel CAYROL [Leberre & Cayrol 1999]). Souvent, la manière de modéliser un problème (e.g. déterminer quels objets représenter et quels types de relations entres les objets) influe considérablement sur la difficulté de résolution du problème. Dans le cadre des applications il est important de considérer à la fois les améliorations au niveau algorithmique et du codage. Un bon codage est un codage qui favorise une résolution efficace. On sait par exemple que la notion de clauses redondantes a une influence sur l'efficacité des algorithmes de résolution (recherche locale et DPL). De nombreux nombres de Ramsey restent actuellement inconnus, on connaît pour certains des majorants et/ou minorants. Résoudre ce type d'instances nécessite en premier lieu de trouver un codage efficace. La notion de symétries fortes introduite dans ma thèse peut contribuer à atteindre un tel objectif.

4.1.2 Autres approches

4.1.2.1 Résolution étendue

Le principe d'extension a été introduit par TSEITIN [Tseitin 1968] : soient $\mathcal{F}$ une formule booléenne, α, β et γ des variables booléennes, où β et γ sont de variables de $\mathcal{F}$ et α une variable (supplémentaire) n'apparaissant pas dans $\mathcal{F}$, alors $\mathcal{F}$ est satisfiable si et seulement si $\mathcal{F} \wedge \{\alpha \Leftrightarrow (\beta \vee \gamma\})$ est satisfiable[4]. Par l'ajout d'une variable supplémentaire pour représenter une formule intermédiaire, on obtient une formule équivalente pour SAT. Cette opération d'ajout est appelée règle d'extension. La résolution étendue est obtenue en augmentant la résolution par la règle d'extension. Des preuves courtes ont été obtenues pour de nombreux problèmes difficiles pour la résolution : problème des pigeons [Cook 1976], formules de Tseitin [Tseitin 1968], problèmes de Ramsey (à deux couleurs) [Krishnamurthy 1985], etc. Ces preuves ont été obtenues grâce à une introduction appropriée de variables supplémentaires représentant des sous-formules. En utilisant ces variables intermédiaires, on évite de construire des résolvantes longues. Ce principe revient à introduire des lemmes pour raccourcir les preuves. Pour exploiter et implémenter une telle méthode, il est nécessaire de répondre à deux questions : « comment introduire de telles variables ? » et « quelle fonction intermédiaire représenter ? ». Ce qui n'est pas du tout évident. En effet, la puissance d'une telle méthode n'est plus à démontrer, mais sa mise en œuvre pratique reste une question difficile. Il suffit de voir le nombre d'auteurs (liste non exhaustive) ayant proposé cette perspective de recherche [Saïs 1993, Castell 1997, Selman 1995, Edwards 1996, Mazure 1999]. Une piste intéressante est d'exploiter les liens entre l'extension et des notions plus exploitables : la notion de clause bloquée[5][Kullmann 1999b] et les NPS[6] [Dubois & Boufkhad 1997].

4.1.2.2 De la structure dans les instances aléatoires ?

Les instances aléatoires $k - SAT$ ne présentent apparemment aucune forme de structure. Une conjecture forte : les instances aléatoires insatisfiables contiennent un noyau minimalement inconsistant contenant des régularités (e.g. symétries). On sait par exemple, que pour un nombre de clauses fixé, plus le nombre de variables augmente plus le seuil tend vers une fonction 0-1. En d'autres termes, le nombre

4. Plus généralement on peut ajouter à $\mathcal{F}$ l'ensemble de clauses codant une fonction booléenne plus générale $\alpha = f(x_1, x_2, \ldots, x_k)$ avec $\alpha \notin var(\mathcal{F})$ et $\forall x \in \{x_1, x_2, \ldots, x_k\}, x \in var(\mathcal{F})$. L'ensemble de clauses ajoutées est appelé extension.

5. Une clause C de $\mathcal{F}$ est dite bloquée si $\exists l \in C, \forall C' \in \mathcal{F}$, tel que $\neg l \in C'$, $Res_l(C, C')$ est tautologique.

6. Une solution s de $\mathcal{F}$ est dite NPS ssi l'interprétation s' obtenue en inversant n'importe quelle variable interprétée à $faux$ contredit $\mathcal{F}$.

de clauses redondantes diminue (pour une étude sur les clauses redondantes [Boufkhad & Roussel 2000, Mazure *et al.* 1998]). Je crois également que lors de la résolution de ce type de problèmes, il n'est pas exclu que les mêmes sous-problèmes à un renommage de variables près soient rencontrés. Ce qui me conduit à proposer ci-dessous une généralisation du théorème de partition du modèle via la symétrie : soient $\mathcal{F}_i$ et $\mathcal{F}_j$ deux ensembles de clauses et σ une application (bijective) de l'ensemble des variables de $\mathcal{F}_i$ dans l'ensemble des variables de $\mathcal{F}_j$, tel que $\sigma(\mathcal{F}_i) \subset \mathcal{F}_j$, alors si $\mathcal{F}_i$ est insatisfiable alors $\mathcal{F}_j$ est insatisfiable. Le problème est que trouver un tel σ est équivalent au problème d'isomorphisme de sous-graphes (un problème NP-Complet).

4.1.2.3 Résolution de SAT par comptage

Une autre technique qu'il est important de revisiter me paraît être la méthode basée sur le comptage des interprétations (non solutions) proposée par IWAMA [Iwama 1987, Iwama 1989]. Cette méthode, appelée aussi algorithme des ensembles indépendants, calcule le nombre d'interprétations supprimées et qui permet de décider de la satisfiabilité de la formule. Il a été montré d'une part que cet algorithme admet un comportement complémentaire aux autres techniques à base d'énumération, et d'autre part que sa complexité en moyenne sur les instances du modèle de probabilité fixé est polynômiale.

4.1.2.4 Formule CNF et représentation en terme de graphes

Il s'agit ici de voir les différentes représentations d'une formule CNF en terme de graphe (e.g. les sommets représentent les clauses et les arêtes représentent par exemple l'existence de résolvantes entre deux clauses, etc.) et d'exploiter ensuite une telle structure. Une voie possible consiste à étudier les possibilités d'adaptation de certains résultats de CSP, traitant de la décomposition de CSP (regroupement en arbres [Dechter & Pearl 1989], ensemble coupe-cycle [Dechter 1990], regroupement cyclique [Jégou 1990]), etc.). Plus généralement la notion de décomposition est liée à la difficulté du problème. En effet, les problèmes difficile tendent à être connexes (e.g. les instances aléatoires, les problèmes structurés), donc difficilement décomposables. Ce qui n'est à mon avis pas le cas des problèmes réels. Cette perspective admet des liens avec l'étude que nous avons commencée sur la généralisation du théorème de partition du modèle (cf. section 2.4.2.1).

4.2 Autour de SAT

4.2.1 Problèmes de satisfaction de contraintes

Dans ce cadre, nous envisageons de poursuivre notre travail dans plusieurs directions détaillées ci-dessous.

4.2.1.1 Filtrages et algorithmes de résolution

Il s'agit dans un premier temps de valider expérimentalement les deux approches proposées (cf. section 3.1). Un algorithme de type MAC est actuellement implémenté. La première approche utilisant une forme partielle de filtrage par consistance de chemin est également intégrée. Il reste à analyser son comportement

sur différentes classes de CSP, et plus particulièrement sur les CSP structurés. Pour ce qui est de la seconde approche (« Singleton arc consistance »), une étude est actuellement conduite en collaboration avec Assef CHMEISS et Christian BESSIÈRE.

4.2.1.2 Heuristiques

Nous pensons que des améliorations significatives peuvent encore être obtenues en utilisant des raffinements plus puissants des heuristiques classiques. En effet, sur de nombreux problèmes, les heuristiques traditionnelles ne permettent pas de discriminer suffisamment entre les différentes variables. Ce qui est le cas de l'heuristique de choix de la variable ayant le domaine de taille minimum (« *min dom* »), et à moindre échelle l'heuristique minimisant le rapport taille du domaine sur le degré de la variable (« *min (dom/deg)* »). La prise en compte du degré de la variable dans la fonction de sélection a permis une amélioration significative. Il s'agit d'aller plus loin en prenant en compte le voisinage immédiat de la variable. Les caractéristiques du voisinage peuvent être soit intégrées dans l'heuristique de départ, soit utilisées pour départager les variables (« *tie-breaker* »). L'idée de tenir compte du voisinage vient du fait que l'affectation d'une valeur à une variable a une incidence immédiate sur les variables voisines. Pour montrer l'intérêt de ce travail, nous avons commencé par établir un générateur de CSP réguliers (toutes les variables ont non seulement la même taille du domaine, mais également le même degré). Ce modèle permet de générer des instances beaucoup plus difficiles que celles obtenues par le modèle standard. Ceci s'explique par le fait que les meilleures heuristiques basées sur des paramètres syntaxiques comme la taille des domaines et le degré de la variable ne font aucune différence entre les variables. Au premier niveau de l'arbre, la taille du « *tie-breaker* » est égale au nombre de variables. Cette remarque suggère d'aller vers des fonctions de sélection prenant en compte la sémantique des contraintes.

J'ai essayé récemment de résoudre un CSP aléatoire généré au seuil en utilisant les différentes techniques de SAT sur le codage CNF de l'instance CSP. Le comportement des meilleurs algorithmes (Satz, CSAT, etc.) a été catastrophique, alors que l'algorithme MAC avec une heuristique « *min dom/deg* » a résolu le problème en moins d'une seconde. Ce comportement s'explique par le fait que MAC exploite la structure du graphe de contraintes, ce qui n'est pas le cas des algorithmes de SAT. Ce qui nous a conduits à étudier les pontages entre les heuristiques de SAT et celles des CSP. Des résultats très intéressants ont déjà été obtenus. L'équivalent de « *min dom/deg* » permet de généraliser l'heuristique « *Moms* » utilisée dans le cadre SAT. Il reste à étudier le comportement d'une telle heuristique. On peut déjà affirmer que cette nouvelle heuristique « retrouve » d'une certaine manière la structure du graphe.

4.2.1.3 Extension de la substituabilité de voisinage

La relation de *substituabilité du voisinage* (VS) sur les valeurs d'une variable a été définie dans [Bellicha *et al.* 1994] : pour deux valeurs a et b du domaine de la variable x_i, on dit que a est voisinage substituable par b si et seulement si l'ensemble des valeurs compatibles avec a est inclus dans l'ensemble des valeurs compatibles avec b. La valeur a peut être supprimée du domaine de x_i tout en obtenant un CSP équivalent pour la satisfiabilité. Ce qui permet de définir des techniques de filtrage par VS [Bellicha *et al.* 1994]. L'extension de VS (EVS) que j'ai obtenue est la suivante : a et b sont dits EVS ssi l'ensemble des valeurs compatibles avec a admet une intersection non vide avec l'ensemble des valeurs compatibles avec b sur chacune des variables y_j voisines de la variable x_i. Cette notion ne permet pas de supprimer la valeur a comme pour VS, mais elle peut être utilisée dans le cadre des algorithmes de recherche pour couper des branches de l'arbre. En effet, on peut considérer un seul tuple parmi ceux du produit cartésien des sous-domaines (valeurs appartenant à l'intersection) des variables voisines. Une utilisation dynamique de cette notion, permettra de couper des branches de l'arbre de recherche et d'éviter par conséquent une exploration redondante de l'espace de recherche. Il reste d'une part à valider expérimentalement cette approche et

d'autre part de voir si cette extension est équivalente ou non à la notion de NPS (« *Negative Prime Solution* ») proposée par Yacine BOUFKHAD [Boufkhad 1996] dans le cadre SAT. Ce dernier point mérite des investigations supplémentaires. Indépendamment, une étude sur l'extension des NPS au cadre CSP a été entamée par Yacine BOUFKHAD et Assef CHMEISS .

4.2.1.4 Compilation et représentation d'ensembles de solutions

Trouver toutes les solutions d'un CSP est une des questions que l'on peut se poser dans le cadre CSP, et qui admet une utilité dans de nombreuses situations : CSP dynamiques, résolution interactive de CSP, problèmes de configurations, problème de conception interactive, etc. (projet CSPFlex du PRC-IA [CSPFlex 1995]). En plus du problème de calcul d'un tel ensemble de solutions (qui peut être de taille exponentiel dans le pire cas), de nombreux auteurs ont proposé différents modes de représentation utilisant par exemple les automates [Vempaty 1992, Amilhastre 1999], les diagrammes de décision binaires [Bouquet & Jégou 1997], etc. Un sous-groupe de travail autour de ce thème a été proposé par J. ALMIHASTRE, P. JANSSEN et M.C. VILAREM dans le cadre du GT 1.2 du GdR I3.

Dans ce cadre, nous pensons utiliser l'algorithme M(AC+) [Chmeiss & Saïs 2000] (cf. section 3.1) pour calculer une représentation compacte de l'ensemble de solutions. En effet, chaque solution partielle trouvée par M(AC+) représente un sous-ensemble de solutions. De plus, le CSP simplifié par une telle solution est un CSP traitable. Une perspective consiste à étudier les extensions possibles des approches de compilation très largement développées dans le cadre propositionnel aux CSP. Par exemple, les approches développées dans [Boufkhad *et al.* 1997] (cf. section 3.2) peuvent être naturellement étendues au cadre CSP.

4.2.1.5 CSP n-aire

Dans le cas des CSP binaires, de nombreux travaux ont été réalisés autour des filtrages (e.g. consistance d'arc, consistance chemin, singleton arc consistance, etc), des classes traitables, etc. Pour obtenir des algorithmes efficaces dans le cas n-aire [Bessière 1999], une perspective intéressante consiste à étudier ces questions dans le cas n-aires. On peut noter d'ailleurs une nette orientation vers l'étude des CSP n-aires. Comme pour la forme CNF, cette tendance a été motivée par la résolution des problèmes réels dont les contraintes portent généralement sur plus de deux variables.

4.2.2 Procédures de preuve pour les logiques non standard

Les problèmes émanants de logiques non standard sont encore plus difficiles: leur complexité se situe généralement au deuxième niveau de la hiérarchie polynômiale. Cependant, comme pour le problème SAT les applications réelles sont loins de présenter le cas le plus défavorable. Dans la mise en œuvre de méthodes de preuves pour les logiques non standard, il est important de partir d'applications réelles, et de tenir compte des caractéristiques propres de telles applications, quitte à utiliser des techniques heuristiques. L'étude que nous avons réalisée dans ce cadre est un premier pas dans cette direction. Comme je l'avais signalé, c'est grâce à Eric GRÉGOIRE et à Pierre MARQUIS que j'ai fait mes premiers pas dans cette problématique. Il s'agit dans un premier temps d'essayer de progresser dans la compréhension des différentes approches développées dans ce domaine, et d'étudier ensuite les possibilités de mise en œuvre de telles procédures de preuve.

4.3 Applications

J'ai très largement utilisé les termes "applications" et "problèmes réels" dans le cadre de cette synthèse. Outre l'intérêt d'utiliser les instances issues d'applications réelles comme base de tests pour l'évaluation pratique des techniques de résolution mise en œuvre, il me paraît important de considérer les applications depuis la phase de modélisation jusqu'à la résolution en passant par l'étape de codage. En plus du lien étroit entre ces différentes phases, cette approche permet souvent d'exhiber de nouvelles directions de recherche. En effet, les applications permettent de montrer non seulement les limites des méthodes de résolution, mais également les difficultés de représentation.

De plus, de nombreuses applications (e.g. planification, ordonnancement) peuvent être résolues en utilisant différentes techniques issues de différents domaines (e.g. recherche opérationnelle, intelligence artificielle). Il est important de continuer dans ce cadre à étudier les liens entre ces différentes approches et les possibilités d'intégration des techniques issues de l'IA et de la RO. Les résultats obtenus ces dernières années sur ce sujet montrent l'importance de continuer ce type de rapprochement.

Les arguments développés ci-dessus montrent l'importance d'une telle perspective ; je reste cependant conscient des difficultés et du travail nécessaire pour faire un pas significatif dans cette direction. Néanmoins, j'envisage une telle démarche pour mes travaux futurs.

Bibliographie

[Achlioptas *et al.* 2000] Dimitris ACHLIOPTAS, Carla GOMES, Henry KAUTZ, et Bart SELMAN. « Generating Satisfiable Instances ». Dans *AAAI'00*, Austin, Texas, 2000. To appear.

[Aguirre 1992] A. San Miguel AGUIRRE. « Symmetries and the Cardinality Operator ». Dans Bernd NEUMANN, éditeur, *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 18–22, Vienna, Austria, 1992. John Wiley, Sons.

[AI 1996] *Special Volume on Frontier in Problem Solving: Phase Transitions and Complexity*, volume 81, numéro 1 de *Artificial Intelligence*, 1996.

[Akers 1978] B. AKERS. « Binary Decision Diagrams ». *IEEE Transactions on Computers*, 27(6):509–516, 1978.

[Amilhastre 1999] J. AMILHASTRE. « *Représentation par automate d'ensemble de solutions de problèmes de satisfaction de contraintes* ». PhD thesis, Université de Montpelier II, Janvier 1999.

[Anbulagan 1998] ANBULAGAN. « *Hypothèse de contrainte : une explication de la réussite de l' heuristique UP dans la résolution des problèmes de satisfiabilité des expressions booléennes* ». Thèse de doctorat, Université de Technologie de Compiègne, juin 1998.

[André 1993] Pascal ANDRÉ. « *Aspects probabilistes du problème de la satisfaction d'une formule booléenne. Étude des problèmes SAT, #SAT et max-SAT* ». Thèse de doctorat, Université Paris 6, 1993.

[Bahia 1992] Groupe BAHIA. « Etude Comparative de Trois Formalismes en Calcul Propositionnel ». Dans *4èmes journnées nationales du PRC-GDR Intelligence Artificielle*, pages 239–318, Marseille, 19-21 Octobre 1992. Teknea.

[Bahia 1995] Groupe BAHIA. « Etude comparatives de trois formalismes en calcul propositionnel ». Dans TEKNEA, éditeur, *5èmes Journées Nationales du PRC-GDR IA*, pages 125–157, Nancy, Février 1995.

[Barth 1995] P. BARTH. « A Davis-Putnam Based Enumeration Algorithm for Linear pseudo-Boolean Optimization ». Rapport Technique MPI-I-95-2-003, MPI Technical Report, 1995.

[Bayardo & Schrag 1996] R. J. BAYARDO et R. SCHRAG. « Using CSP Look-Back Techniques to Solve Exceptionally Hard SAT Instances ». Dans *Proceedings of the Second International Conference on Principles and Practice of Constraint Programming (CP'96)*, volume 1118, pages 46–60. LNCS, Springer, 1996.

[Bayardo Jr. & Schrag 1997] Roberto J. BAYARDO JR. et Robert C. SCHRAG. « Using CSP Look-Back Techniques to Solve Real-World SAT Instances ». Dans *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, pages 203–208, Providence (Rhode Island, USA), juillet 1997.

[Belleannée & Vorc'h 1994] C. BELLEANNÉE et R. VORC'H. « A linear tableau proof for the pigeon-hole formulae using symmetry ». Dans *Proceedings of the Workshop on Theorem*

Proving with Analytic Tableaux and Related Methods, Imperial College - London - UK Oxford, 1994. TR-94/5.

[Bellicha *et al.* 1994] Amit BELLICHA, Michel HABIB, Marie Catherine VILAREM, Christian CAPELLE, et Tibor KÖKÉNY. « CSP Techniques using Partial Orders on Domain Values ». Dans Thomas SCHIEX et Christian BESSIÉRE, éditeurs, *Workshop on Constraint Satisfaction Issues raised by Practical Applications (ECAI'94)*, Amsterdam, août 1994.

[Benhamou & Saïs 1991] B. BENHAMOU et L. SAÏS. « Study of symmetries in propositional calculus ». Dans *International Workshop on Computer Science*, pages 78–102, Annaba, 16-18, Décembre 1991.

[Benhamou & Saïs 1992] Belaïd BENHAMOU et Lakhdar SAÏS. « Theoretical Study of Symmetries in Propositional Calculus and Applications ». Dans *Proceedings of Eleventh International Conference on Automated Deduction (CADE-11)*, volume 607 de *Lecture Notes in Artificial Intelligence*, pages 281–294. Springer Verlag, 1992.

[Benhamou & Saïs 1994] Belaïd BENHAMOU et Lakhdar SAÏS. « Tractability Through Symmetries in Propositional Calculus ». *Journal of Automated Reasoning*, 12:89–102, 1994.

[Benhamou 1994] Belaïd BENHAMOU. « Study of Symmetry in constraint satisfaction problems ». Dans *Proceedings of PPCP'94*, Orcas Island Resario, Washington, 1994.

[Benhamou *et al.* 1992a] B. BENHAMOU, L. SAÏS, et P. SIEGEL. « Dealing with cardinality formuls in propositional calculus ». Dans *International Workshop on Tractable Reasoning (AAA'92)*, pages 6–12, San Jose, California, 12-17 July 1992.

[Benhamou *et al.* 1992b] B. BENHAMOU, L. SAÏS, et P. SIEGEL. « Dealing with symmetries in propositional calculus ». Dans *International Workshop on Tractable Reasoning (AAA'92)*, San Jose, California, 12-17 July 1992.

[Benhamou *et al.* 1994] Belaïd BENHAMOU, Lakhdar SAÏS, et Pierre SIEGEL. « Two Proof Procedures for a Cardinality Based Language in Propositional Calculus ». Dans P. ENJALBERT, E.W. MAYR, et K.W. WAGNER, éditeurs, *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 775 de *Lecture Notes in Computer Science*, pages 71–82, Caen (France), février 1994. Springer Verlag.

[Bennaceur & Plateau 1992] H. BENNACEUR et G. PLATEAU. « FAST: une méthode de résolution du problème linéaire de satisfaction de contraintes ». *Techniques et Sciences Informatique*, 11(3):33–57, 1992.

[Bennaceur & Plateau 1995] H. BENNACEUR et G. PLATEAU. « An exact algorithm for constraint satisfaction problem: Application to logical inference ». *Information Processing Letter*, 48:151–158, 1995.

[Benson & Freuder 1992] B.W. BENSON et E.C. FREUDER. « Interchangeability Preprocessing Can Improve Forward Checking Search ». Dans Bernd NEUMANN, éditeur, *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI'92)*, pages 28–30, Vienna, Austria, 1992. John Wiley, Sons.

[Bessant *et al.* 1998] B. BESSANT, E. GRÉGOIRE, P. MARQUIS, et L. SAÏS. « Combining nonmonotonic reasoning and belief revision: a practical approach ». Dans F. GIUNCHIGLIA, éditeur, *8th Int. Conf. on Artificial Intelligence - Methodology, Systems Applications (AIMSA'98)*, pages 115–128, Sozopol, Bulgaria, september 1998. LNCS 1480.

[Bessant *et al.* 2000] B. BESSANT, E. GRÉGOIRE, P. MARQUIS, et L. SAÏS. « *Frontiers in Belief Revision* », Chapitre Iterated Syntax-Based Revision in a Nonmonotonic Setting. Kluwer Academic Publishers, M.-A. Williams and H. Rott édition, 2000. sous presse.

[Bessière & Régin 1996] C. BESSIÈRE et J-C. RÉGIN. « MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems ». Dans *International conference on Constraint programming (CP'96), LNCS 1118*, pages 61–75, 1996.

[Bessière 1999] C. BESSIÈRE. « Non-binary constraints ». Dans *International conference on Constraints Programming (CP'99)*, pages 24–27, 1999. Invited lecture.

[Biere *et al.* 1999] A. BIERE, A. CIMATTI, E. M. CLARKE, M. FUJITA, et Y. ZHU. « Symbolic Model Checking using SAT procedures instead of BDDs ». Dans *Design Automaton Conference, DAC'99*, 1999.

[Billionnet & Sutter 1992] Alain BILLIONNET et Alain SUTTER. « An efficient algorithm fot the 3-satisfiability problem ». *Operations Research Letters*, 12:29–36, 1992.

[Blake 1937] BLAKE. « *Canonical Expressions in Boolean Algebra* ». PhD thesis, University of Chicago, Illinois, USA, 1937.

[Boole 1854] G. BOOLE. *Les lois de la pensée*. Mathesis, 1854.

[Boufkhad & Roussel 2000] Yacine BOUFKHAD et Olivier ROUSSEL. « Redundancy in Random SAT Formulas ». Dans *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'2000)*, Austin, Texsas, August 2000.

[Boufkhad 1996] Yacine BOUFKHAD. « *Aspects probabilistes et algorithmiques du problème de satisfiabilité* ». Thèse de doctorat, Université de Paris 6, Laboratoire d' Informatique de Paris 6 (LIP6), décembre 1996.

[Boufkhad *et al.* 1997] Yacine BOUFKHAD, Éric GRÉGOIRE, Pierre MARQUIS, Bertrand MAZURE, et Lakhdar SAÏS. « Tractable Cover Compilations ». Dans *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, volume 1, pages 122–127, Nagoya (Japan), août 1997.

[Bouquet & Jégou 1997] F. BOUQUET et P. JÉGOU. « Using obdds to handle dynamic constraints ». *Information Processing Letters*, 62:111–120, 1997.

[Brisoux *et al.* 1998] Laure BRISOUX, Lakhdar SAÏS, et Éric GRÉGOIRE. « Mieux Exploiter les échecs au sein des arbres de recherche à la Davis et Putnam ». Dans *Actes des Quatrièmes Journées Nationales sur la Résolution Pratique des Problèmes NP-Complets (JNPC'98)*, pages 31–39, Nantes (France), mai 1998.

[Brisoux *et al.* 1999] L. BRISOUX, É. GRÉGOIRE, et L. SAÏS. « Improving backtrack search for SAT by means of redundancy ». Dans Z.W. RAS et A. SKOWRON, éditeurs, *11th Int. Symposium on Methodologies for Intelligent Systems (ISMIS'99)*, pages 301–309, Varsovie, Pologne, June 1999.

[Brisoux *et al.* 2000] L. BRISOUX, L. SAIS, et E. GRÉGOIRE. « Recherche locale : vers une exploitation des propriétés structurelles ». Dans *Actes des Sixièmes Journées Nationales sur la Résolution Pratique des Problèmes NP-Complets(JNPC'00)*, pages 243–244, Marseille, 2000.

[Brisoux-Devendeville 1999] L. BRISOUX-DEVENDEVILLE. « *Satisfaisabilité propositionnelle en informatique: Aspects algorithmiques et extensions du formalisme* ». PhD thesis, Université d'Artois, Décembre 1999.

[Bruni & Sassano 2000] Renato BRUNI et Antonio SASSANO. « Detecting Minimaly Unsatisfiable Subformulaein unsatisfiable SAT instances by means of Adaptative Core Search ». Dans *Third Workshop on the Satisfiability Problem (SAT'2000)*, May 14-18 2000.

[Bruynooghe 1980] M. BRUYNOOGHE. « Analysis of Dependencies to Improve the Behaviour of Logic Programs ». Dans *5th Conference on Automated Deduction, CADE'5*, pages 293–305, 1980.

[Bryant 1992] R. BRYANT. « Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams ». *ACM Computing Surveys*, 1992.

[Buno & Buning 1993] M. BUNO et H.K. BUNING. « Report on SAT competition ». *Bulletin of the European Association for Theoritical Computer Science*, 49:143–151, Feb. 1993.

[Carrol 1966] Lewis CARROL. *La logique sans peine*. Herman Paris, 1966.

[Castell & Cayrol 1996] Thierry CASTELL et Michel CAYROL. « Computation of prime implicates and prime implicants by the Davis and Putnam Procedure ». Dans *Proceedings of ECAI'96 Workshop on Advances in Propositional Deduction*, pages 61–64, Budapest (Hungary), août 1996.

[Castell & Cayrol 1997] Thierry CASTELL et Michel CAYROL. « Hidden Gold in Random Generation of SAT Satisfiable Instances ». Dans *International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 372–377, 1997.

[Castell & Fargier 1998] Thierry CASTELL et Hélène FARGIER. « Between SAT and CSP: Propositional Satisfaction Problems and Clausal CSPs ». Dans Henri PRADE, éditeur, *13th European Conference on Artificial Intelligence (ECAI'98), Brighton*, pages 214–218. John Wiley & Sons, 1998.

[Castell 1996] Thierry CASTELL. « Résolution arrière dans la procédure de Davis et Putnam ». Dans *Rencontres Françaises en Intelligence Artificielle (RFIA'96)*, pages 109–117, Rennes (France), 1996.

[Castell 1997] Thierry CASTELL. « *Consistance et déduction en logique propositionnelle* ». Thèse de doctorat, Université Paul Sabatier, Toulouse, janvier 1997.

[C.E. Blair & Lowe 1986] R.G. Jeroslow C.E. BLAIR et J.K. LOWE. « Some results and experiments in programming techniques for propositional logic ». *Comput. Oper. Res.*, 13(5):633–645, 1986.

[Cha & Iwama 1995] Byungki CHA et Kazuo IWAMA. « Performance Test of Local Search Algorithms Using New Types of Random CNF Formulas ». Dans *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 304–310, 1995.

[Chabrier 1997] Jacqueline CHABRIER. « *Programmation Par Contraintes: langages, méthodes et applications sur les domaines entiers et booléens* ». Thèse d'habilitation à diriger des recherches, Université de Bourgogne, décembre 1997.

[Chabrier *et al.* 1991] Jacqueline CHABRIER, Jean-Jacques CHABRIER, et F. TROUSSET. « Résolution Efficace d'un Problème de Satisfaction de Contraintes : le million de reines ». Dans *Onzièmes Journées Internationales sur les Systèmes Experts et leurs Applications*, 1991.

[Chabrier *et al.* 1995] Jacqueline CHABRIER, Vincent JULIARD, et Jean-Jacques CHABRIER. « SCORE(FD/B) an efficient complete local-based serach method for satisfiability problems ». Dans *Proceedings of Constraint Programming Workshop on Solving Really Hard Problems*, pages 25–30, Cassis (France), septembre 1995.

[Chabrier *et al.* 1996] Jacqueline CHABRIER, Jean-Michel RICHER, et Chabrier JEAN-JACQUES. « Resolution of structred SAT problems with SCORE(FD/B) ». Dans *Proceedings of the CP'96 Workshop on Constraint Programming Applications*, Boston, (MA, USA), août 1996.

[Chandru & Hooker 1990] V. CHANDRU et J.N. HOOKER. « Extended Horn sets in Propositional Logic ». *Journal of the Association for Computing Machinery*, 38:205–221, 1990.

[Chandru & Hooker 1999] Vijay CHANDRU et John HOOKER. *Optimization Methods for Logical Inference*. Wiley, 1999.

[Chatalic & Simon 2000] Philippe CHATALIC et Laurent SIMON. « ZRes: The Old Davis-Putnam Procedure meets ZBDD, David ». Dans David MACALLESTER, éditeur, *CADE'17*, pages 449–454, 2000.

[Cheeseman *et al.* 1991] Peter CHEESEMAN, Bob KANEFSKY, et William M. TAYLOR. « Where the Really Hard Problems Are ? ». Dans *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 331–337, 1991.

[Chmeiss & Saïs 2000] A. CHMEISS et L. SAÏS. « About Local consistency in Solving CSPs ». Dans *Twelfth IEEE International Conference on Tools with Artificial Intelligence, ICTAI'00*, Vancouver, British Columbia, Canada, November,13-15 2000. IEEE Computer Society. à paraitre.

[Chmeiss 1996] A. CHMEISS. « *Reseaux de contraintes: Algorithmes de Propagation et de Décomposition* ». PhD thesis, Université d'Aix-Marseille I, 1996.

[Chu-Min 2000] Li CHU-MIN. « Integrating Equivalency reasoning into Davis-Putnam proce-

dure ». Dans *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'2000)*, Austin Texsas, 2000.

[Chvátal & Reed 1992] V. CHVÁTAL et B. REED. « Miks gets some (the odds are on this side) ». Dans *Proceedings of the* 33^{rd} *IEEE Symposium on Foundations of Computer Science*, pages 620–627, 1992.

[Chvátal & Szemerédi 1988] Vašek CHVÁTAL et Endre SZEMERÉDI. « Many hard examples for resolution ». *Journal of the Association for Computing Machinery*, 35(4):759–768, 1988.

[Conforti & Cornuéjols 1992] M. CONFORTI et G. CORNUÉJOLS. « A class of logical inference problems solvable by linear programming ». Dans *Proceedings of FOCS'92*, volume 33, pages 670–675, 1992.

[Cook 1971] S. A. COOK. « The complexity of theorem-proving procedures ». Dans *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, New York (USA), 1971.

[Cook 1976] S.A. COOK. « A short Proof of the Pigeon-Hole Principle Using Extended Resolution ». *SIGACT News*, 8:28–32, 1976.

[Cooper *et al.* 1994] M.C. COOPER, D. A. COHEN, et P. G. JEAVONS. « Characterizing tractable Constraints ». *Artificial Intelligence*, 65:347–361, 1994.

[Crawford & Auton 1993] James M. CRAWFORD et Larry D. AUTON. « Experimental Results on the Crossover Point in Satisfiablity Problems ». Dans *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93)*, pages 21–27, 1993.

[Crawford & Auton 1996] James M. CRAWFORD et Larry D. AUTON. « Experimental Results on the Crossover Point in Random 3SAT ». *Artificial Intelligence*, 81(1-2), 1996.

[Crawford & Baker 1994] J.M. CRAWFORD et A.B. BAKER. « Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems ». Dans *Twelfth National Conference on Artificial Intelligence, AAAI'94*, 1994.

[Crawford 1992] James M. CRAWFORD. « A theorical analysis of reasoning by symmetry in first-order logic ». Dans *Proceedings of the AAAI'92 Workshop on Tractable Reasoning*, San Jose (USA), juillet 1992.

[Crawford 1993] James M. CRAWFORD. « Solving Satisfiability Problems Using a Combination of Systematic and Local Search ». Dans *Working notes of the DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*, 1993.

[Crawford *et al.* 1996] James CRAWFORD, Matthew L. GINSBERG, Eugene LUCK, et Amitabha ROY. Symmetry-Breaking Predicates for Search Problems. Dans Luigia Carlucci AIELLO, Jon DOYLE, et Stuart SHAPIRO, éditeurs, *Principles of Knowledge Representation and Reasoning (KR'96)*, pages 148–159. Morgan Kaufmann, San Francisco, California, 1996.

[CSPFlex 1995] Projet CSPFLEX. « Autour du problème de satisfaction de contraintes ». Dans *Actes des 5ièmes journées nationales du PRC-GDR Intelligence Artificielle*, Nancy, France, Février 1995. Groupe PRCIA Représentation et traitement de la flexibilité dans les problèmes sous contraintes.

[Cubbada & Mouseigne 1988] C. CUBBADA et M.D. MOUSEIGNE. « *Variantes de l'algorithme de SL-résolution avec retenue d'information* ». Thèse de doctorat, Université d'Aix-Marseille II (GIA Luminy), Marseille (France), 1988.

[Dalal & Etherington 1992] Mukesh DALAL et David W. ETHERINGTON. « A hierarchy of tractable satisfiability problems ». *Information Processing Letters*, 44(4):173–180, 10 décembre 1992.

[Dantsin *et al.* 2000] E. DANTSIN, A. GOERDT, E. A. HIRSCH, et U. SCHÖNING. « Deterministic algorithms for k-SAT based on covering codes and local search ». Dans *ICALP'00*, 2000.

[Davis & Putnam 1960] Martin DAVIS et Hilary PUTNAM. « A Computing Procedure for Quantification Theory ». *Journal of the Association for Computing Machinery*, 7:201–215, 1960.

[Davis *et al.* 1962] Martin DAVIS, George LOGEMANN, et Donald LOVELAND. « A Machine Program for Theorem Proving ». *Journal of the Association for Computing Machinery*, 5:394–397, 1962.

[Debruyne & Bessière 1997] R. DEBRUYNE et C. BESSIÈRE. « Some Practicable Filtering Techniques for the Constraint Satisfaction Problems ». Dans *International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 412–417, 1997.

[Dechter & Pearl 1989] R. DECHTER et J. PEARL. « Tree Clustering for Constraint Networks ». *Artificial Intelligence*, 38:353–366, 1989.

[Dechter & Rish 1994] Rina DECHTER et Irina RISH. « Directional Resolution: the Davis-Putnam procedure revisited ». Dans J. DOYLE, E. SANDEWALL, et P. TORASSI, éditeurs, *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 134–145, 1994.

[Dechter 1989] R. DECHTER. « Enhancement Schemes for Constraint Processing: Backjumping, Learning, and cutset Decomposition ». *Artificial Intelligence*, 41:273–312, 1989.

[Dechter 1990] R. DECHTER. « Enhancement Schemes for Constraint-satisfaction problems: Backjumping, Learning and Cutset Decomposition ». *Artificial Intelligence*, 41:273–312, 1990.

[del Val 1994] Alvaro DEL VAL. « On the Relation between the Coherence and Foundations Theories of Belief Revision ». Dans *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, pages 909–914, Menlo Park, CA, USA, 1994. AAAI Press.

[Dimacs 1993] DIMACS. « Second Challenge on Satisfiability Testing organized by the Center for Discrete Mathematics and Computer Science of Rutgers University », 1993. `http://dimacs.rutgers.edu/Challenges/`.

[Dubois & Boufkhad 1996] Olivier DUBOIS et Yacine BOUFKHAD. « From very hard doubly balanced SAT formulae to easy unbalanced SAT formulae, variations of the satisfiability threshold ». Dans J.G. DING-ZHU DU et P. PARDALOS, éditeurs, *Proceedings of the DIMACS Workshop on the Satisfiability Problem: Theory and Applications*, mars 1996.

[Dubois & Boufkhad 1997] Olivier DUBOIS et Yacine BOUFKHAD. « A General Upper Bound for the Satisfiability Threshold of Random r-SAT Formulae ». *Journal of Algorithms*, 24(2):395–420, août 1997.

[Dubois & Carlier 1991] Olivier DUBOIS et Jacques CARLIER. « Probabilistic Approach to the Satisfiability Problem ». *Theoretical Computer Science*, 81:65–75, 1991.

[Dubois 1990] Olivier DUBOIS. « On the r,s-SAT satisfiability problem and a conjecture of Tovey ». *Discrete Applied Mathematics*, 26:51–60, 1990.

[Dubois 2000] O. DUBOIS. « Typical random 3-SAT formulae and the satisfiability Threshold ». Dans *Third international workshop on the satisfiability problem (SAT2000)*, 2000.

[Dubois *et al.* 1996] Olivier DUBOIS, Pascal ANDRÉ, Yacine BOUFKHAD, et Jacques CARLIER. « SAT versus UNSAT ». Dans D.S. JOHNSON et M.A. TRICK, éditeurs, *Second DIMACS Challenge, 1993*, DIMACS Series in Discrete Mathematics and Theorical Computer Science, American Mathematical Society, pages 415–436, 1996.

[Dunham & Wang 1976] Bradford DUNHAM et Hao WANG. « Feasible solutions of the tautology problem ». *Annals of Mathematical Logic*, 10:117–154, 1976.

[E. Boros & Kogan 1994] P.L. Hammer E. BOROS et A. KOGAN. « Computational experiments with an exact SAT solver ». Dans *3rd International Symposium On Artificial Intelligence and Mathematics*, Florida, 1994.

[Edwards 1996] Douglas D. EDWARDS. « Research Summary », December 1996. http://www.cs.cornell.edu/home/selman/.

[Ellman 1993] Thomas ELLMAN. « Abstraction via Approximate Symmetry ». Dans *IJCAI'93*, 1993.

[Franco & Paul 1983] J. FRANCO et M. PAUL. « Probabilistic analysis of the Davis-Putnam procedure for solving satisfiability ». *Discrete Applied Mathematics*, 5:77–87, 1983.

[Franco *et al.* 1997] J. FRANCO, J. GU, P.W. PURDOM, et B.W. WAH. *Algorithms for Satisfiability (SAT) Problem: A Survey*. American Mathematical Society, 1997.

[Freeman 1995] Jon William FREEMAN. « *Improvements to Propositional Satisfiability Search Algorithms* ». PhD thesis, University of Pennsylvania, Department of Computer and Information Science, 1995.

[Freuder 1982] E. C. FREUDER. « A Sufficient Condition for Backtrack-free Search ». *Journal of ACM*, 29(1):24–32, 1982.

[Frieze & Suen 1996] Alan FRIEZE et Stephen SUEN. « Analysis of Two Simple Heuristics on a Random Instance of k-SAT ». *Journal of Algorithms*, 20(2):312–355, mars 1996.

[Galil 1977] Z. GALIL. « On the complexity of regular resolution and the Davis-Putnam Procedure ». *Theoretical Computer Science*, 4:23–46, 1977.

[Gallo & Scutellà 1988] Giorgio GALLO et Maria Grazia SCUTELLÀ. « Polynomially solvable satisfiability problems ». *Information Processing Letters*, 29(5):221–227, 24 novembre 1988.

[Gallo & Urbani 1989] Giorgio Gallo GALLO et Giampaolo URBANI. « Algorithms for Testing the Satisfiability of Propositional Formulae. ». *Journal of Logic Programming*, 7(1):45–61, juillet 1989.

[Garey & Johnson 1979] Michael R. GAREY et David S. JOHNSON. *Computers and Intractability: A Guide to the Theory on NP-completness*. A series of books in the Mathematical Sciences. W. H. Freeman and Company, New-York (USA), 1979.

[Génisson & Jégou 1996] R. GÉNISSON et P. JÉGOU. « Davis and Putnam were Already Checking Forward ». Dans *12th European Conference on Artificial Intelligence (ECAI'96)*, pages 180–184, Chichester - New York - Brisbane, août 1996. Wiley.

[Génisson & Rauzy 1996] Richard GÉNISSON et Antoine RAUZY. « Sur les relations algorithmiques des classes polynomiales du problème SAT et des problèmes de satisfaction de contraintes ». Dans *Actes de RFIA'96*, pages 97–107, 1996.

[Génisson & Saïs 1994] Richard GÉNISSON et Lakhdar SAÏS. « Some ideas on random generation of K-SAT instances ». Dans *International Workshop on Experimental Evaluation of Reasoning and Search Methods (AAAI'94)*, pages 91–92, 1994.

[Génisson & Siegel 1994] Richard GÉNISSON et Pierre SIEGEL. « A polynomial method for sub-clauses production ». Dans *Proceedings of Sixth International Conference on Artificila Intelligence: Methodology, Systems, Applications (AIMSA'94)*, pages 25–34, North Holland, 1994.

[Gent & Walsh 1993] Ian P. GENT et Tobby WALSH. « Towards an Understanding of Hill-Climbing Procedures of SAT ». Dans *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93)*, pages 28–33, 1993.

[Gent & Walsh 1994] Ian P. GENT et Tobby WALSH. « The SAT phase transition ». Dans *Proceedings of the Eleventh European Conference on Artificial Intelligence(ECAI'94)*, pages 105–109, 1994.

[Ginsberg 1993] Matthew L. GINSBERG. « Dynamic Backtracking ». *Journal of Artificial Intelligence Research (JAIR)*, 1:25–46, 1993.

[Glover 1989] F. GLOVER. « Tabu search - Part I ». *ORSA Journal of Computing*, 1:190–206, 1989.

[Goerdt 1992] A. GOERDT. « A Treshold for Unsatisfiability ». *Lecture Notes in Computer Science*, 629:264–??, 1992.

[Goldberg 1979a] A. GOLDBERG. « Average case complexity of the satisfiability problem ». Dans *4th workshop on Automated Deduction*, pages 1–6, 1979.

[Goldberg 1979b] A. GOLDBERG. « On the complexity of satisfiability problems ». Rapport Technique 16, Courant Computer Science, New York University, 1979.

[Goldberg *et al.* 1983] A. GOLDBERG, P.W. PURDOM, et C.A. BROWN. « Average time analysis of simplified Davis and Putnam Procedures ». *Information Processing Letter*, 15:72–75, 1983.

[Gomes & Selman 1997] Carla P. GOMES et Bart SELMAN. « Algorithm Portfolio Design: Theory vs. Practice ». Dans Dan GEIGER et Prakash Pundalik SHENOY, éditeurs, *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI'97)*, pages 190–197, San Francisco, 1997. Morgan Kaufmann Publishers.

[Gomes *et al.* 1998] Carla P. GOMES, Bart SELMAN, et Henry KAUTZ. « Boosting Combinatorial Search through Randomization ». Dans *Proceedings of the 15th National Conference on Artificial Intelligence, (AAAI'98)*, pages 431–437, 1998.

[Graham *et al.* 1980] R. L. GRAHAM, B. L. ROTHSCHILD, et J. H. SPENCER. *Ramsey Theory*. John Wiley & Sons, 1980.

[Gre] « Greentech Computing Ltd. (UK) ». http://www.greentech-computing.co.uk/.

[Grégoire *et al.* 1998] Éric GRÉGOIRE, Bertrand MAZURE, et Lakhdar SAÏS. « Logically-complete local search for propositional nonmonotonic knowledge bases ». Dans Ilkka NIELELÄ et Torsten SCHAUB, éditeurs, *Proceedings of the KR'98 Workshop on Computational Aspects of Nonmonotonic Reasoning*, numéro 52 dans Series A: Research Reports, pages 37–45. Helsinky University of Technology, Digital Systems Laboratory, 1998. ISSN 0783 5396, ISBN 951 22 4072 6.

[Groote & Warners 2000] Jan Friso GROOTE et Joost P. WARNERS. « The Propositional Formula Checker HeerHugo ». *Journal of Automated Reasoning*, 24, February 2000.

[Gu 1997] J. GU. *Constraint-Based Search*. Combridge University Press, 1997.

[Haken 1985] Armin HAKEN. « The intractability of resolution ». *Theritical Computer Science*, 39:297–308, 1985.

[Hansen & Jaumard 1990] Pierre HANSEN et Brigitte JAUMARD. « Algorithms for the Maximum Satisfiability Problem ». *Journal of Computing*, 22:279–303, 1990.

[Hao & Tétard 1996] Jin-Kao HAO et Laurent TÉTARD. « CH-SAT: A complete Heuristic procedure for satisfiability problem ». Dans *Proceedings of ECAI'96 Workshop on Advances in Propositional Deduction*, pages 27–38, Budapest (Hungary), août 1996.

[Hao 1995] J.K. HAO. « A clausal genetic representation and its related evolutionary procedures for satisfiability problems ». Dans *Int. Conf. on Artificial Neural Nets and Genetic Algorithms*, pages 289–292. Springer-Verlag, 1995.

[Hentenryck & Deville 1990] P.V. HENTENRYCK et Y. DEVILLE. « The cardinality operator: A new logical connective for constraint programming ». Rapport Technique 24, Brown University, Computer Science Departement, 1990.

[Hirsch 2000] E. A. HIRSCH. « SAT Local Search Algorithms: Worst-Case Study ». *Journal of Automated Reasoning (special issue on SAT)*, 24(1/2):127–143, February 2000.

[Hooker & Yan 1999] J. N. HOOKER et H. YAN. « Tight representation of logical constraints as cardinality rules ». *Mathematical Programming*, 85:363–377, 1999.

[Hooker 1988] J.N. HOOKER. « A quantitative approach to logical inference ». *Decision Support Systems*, 7(1):1–7, 1988.

[Hooker 2000] John HOOKER. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, 2000. sous press.

[Hoos & Stützle 2000] Holger H HOOS et Thomas STÜTZLE. « SATLIB: An Online Ressource for Research on SAT ». SAT 2000, Ian Gent, Hans van Maaren, and Toby Walsh, editors, IOS Pres, 2000. http://aida.intellektik.informatik.tu-darmstadt.de/SATLIB/.

[Iwama 1987] Kazuo IWAMA. « Complementary Approaches to CNF Boolean Equations ». *PERSPEC: Perspectives in Computing*, 15, 1987.

[Iwama 1989] Kazuo IWAMA. « CNF-satisfiability test by counting and polynomial average time ». *SIAM Journal on Computing*, 18(2):385–391, avril 1989.

[Jeannicot *et al.* 1988] S. JEANNICOT, Laurent OXUSOFF, et Antoine RAUZY. « Évaluation sémantique en calcul propositionnel : une propriété de coupure pour rendre plus efficace la procédure de Davis et Putnam ». *Revue d'Intelligence Artificielle*, 2(1):41–60, 1988.

[Jeroslow & Wang 1990] Robert G. JEROSLOW et Jinchang WANG. « Solving Propositional Satisfiability Problems ». *Annals of Mathematics and Artificial Intelligence*, 1:167–187, 1990.

[Jégou 1990] Ph. JÉGOU. « Cyclic-Clustering: a Compromise Between Tree-Clustering and the Cycle-Cutset Method for Improving Search Efficiency ». Dans *ECAI'90*, pages 369–371, Stockholm, 1990.

[Jégou 1993] P. JÉGOU. « Decomposition of domains based on the structure of finite constraints satisfaction problems ». Dans *AAAI'93*, pages 731–736, 1993.

[Johnson 1999] Geroge JOHNSON. « Separating Insolvable and Difficult ». The New York Times on the Web, July 13 1999. http://www.nytimes.com/.

[Kamath *et al.* 1994] Anil KAMATH, Rajeev MOTWANI, Krishna PALEM, et Paul SPIRAKIS. « Tail Bounds for Occupancy and the Satisfiability Threshold Conjecture ». Dans *35th Annual Symposium on Foundations of Computer Science, IEEE*, pages 592–603, Santa Fe, New Mexico, 20–22 novembre 1994.

[Kautz & Selman 1992] H. KAUTZ et B. SELMAN. « Planning as satisfiability ». Dans *European Conference on Artificial Intelligence (ECAI'92)*, pages 359–363, Vienna, Austria, 1992.

[Kim & Zhang 1994] Sun KIM et Hantao ZHANG. « ModGen: Theorem Proving by Model Generation ». Dans *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94)*, pages 162–167, Menlo Park, CA, USA, 1994. AAAI Press.

[Kowalski & Kuehner 1971] R.A. KOWALSKI et D. KUEHNER. « Linear Resolution with Selection Function ». *Artificial Intelligence*, 2:227–260, 1971.

[Krishnamurthy 1982] B. KRISHNAMURTHY. « *Examples of hard tautologies and Word-Case complexity results in propositional calculus* ». PhD thesis, Dept. of Computer and Information Science, University of Massachussets, 1982.

[Krishnamurthy 1985] B. KRISHNAMURTHY. « Short Proofs for Tricky Formulas ». *Acta Informatica*, 22:253–275, 1985.

[Kullmann 1999a] Oliver KULLMANN. « New methods for 3-SAT decision and worst-case analysis ». *Theoretical Computer Science*, 223(1-2):1–72, July 1999.

[Kullmann 1999b] Oliver KULLMANN. « On a generalization of extended resolution ». *Discrete Applied Mathematics*, 96-97(1-3):149–176, 1999.

[Lang & Marquis 1998] J. LANG et P. MARQUIS. « Complexity Results for Independence and Definability in Propositional Logic ». Dans *6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 356–367, Trento, 1998.

[Larrabee & Tusji 1993] T. LARRABEE et Y. TUSJI. « Evidence for satisfiability threshold for rand 3CNF formalas ». Dans H.E. HIRSH, éditeur, *Proceedings of Spring Symposium on Artificial Intelligence and NP-Hard Problems*, pages 112–118, Stanford (CA, USA), 1993.

[Larrabee 1992] T. LARRABEE. « Efficient generation of test patterns unsing boolean satisfiability ». *IEEE Transaction on CAD*, 11:4–15, 1992.

[Leberre & Cayrol 1999] Daniel LEBERRE et Michel CAYROL. « Influence du codage en calcul propositionnel », 1999. Thème 1 du Groupe de Travail 1.2 "axe algorithme" (GdR I3).

[Li & Anbulagan 1997] Chu Min LI et ANBULAGAN. « Heuristics Based on Unit Propagation for Satisfiability Problems ». Dans *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 366–371, Nagoya (Japan), août 1997.

[Liberatore & Schaerf 1996] Paolo LIBERATORE et Marco SCHAERF. « The Complexity of Model Checking for Belief Revision and Update ». Dans *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 556–561, Menlo Park, 1996. AAAI Press / MIT Press.

[Lobjois & Lemaître 1997] Lionel LOBJOIS et Michel LEMAÎTRE. « Coopération entre méthodes complètes et incomplètes pour la résolution de (V)CSP : une tentative d'inventaire ». Dans *Actes des Troisièmes Journées Nationales sur la Résolution Pratique de Problèmes NPè-complets (JNPC'97)*, pages 67–73, Rennes (France), avril 1997. PRC-GDR IA – Universitè de Rennes 1.

[Loveland 1978] D. LOVELAND. *Automated Theorem Proving: A Logical Basis*. North Holland, 1978.

[Makinson & Gärdenfors 1991] David MAKINSON et Peter GÄRDENFORS. Relations between the Logic of Theory Change and the Nonmonotonic Logic. Dans André FUHRMANN et M. MORREAU, éditeurs, *The Logic of Theory Change*, pages 185–205. Springer-Verlag, Berlin, 1991.

[Marquis 1995] Pierre MARQUIS. « Knowledge compilation using theory prime implicates ». Dans *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 837–843, 1995.

[Massacci & Marraro 2000] Fabio MASSACCI et Laura MARRARO. « Logical cryptanalysis as a SAT-problem: Encoding and analysis of the u.s. Data Encryption Standard ». *Journal of Automated Reasoning*, 2000.

[Massacci 1999] F. MASSACCI. « Using Walk-SAT and Rel-SAT for Cryptographic Key Search ». Dans *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 290–295, 1999.

[Mazure 1995] Bertrand MAZURE. « Expérimentations, analyse et amélioration des méthodes de résolution du problème SAT ». Mémoire de dea, CRIL – Université d'Artois, Centre de Recherche en Informatique de Lens (Faculté Jean Perrin), juillet 1995.

[Mazure 1999] Bertrand MAZURE. « *De la Satisfaisabilité à la Compilation de Bases de Connaissances Propositionnelles* ». Thèse de doctorat, Université d'Artois, Centre de Recherche en Informatique de Lens (Faculté Jean Perrin), janvier 1999.

[Mazure *et al.* 1996] Bertrand MAZURE, Lakhdar SAÏS, et Éric GRÉGOIRE. « Detecting Logical Inconsistencies ». Dans *Proceedings of Mathematics and Artificial Intelligence Symposium*, pages 116–121, Fort Lauderdale (FL USA), janvier 1996.

[Mazure *et al.* 1997a] Bertrand MAZURE, Lakhdar SAÏS, et Éric GRÉGOIRE. « A comparison of Two Approaches to Inconsistency Detecting ». Dans *European Symposium on Intelligent Techniques*, Bari (Italy), mars 1997.

[Mazure *et al.* 1997b] Bertrand MAZURE, Lakhdar SAÏS, et Éric GRÉGOIRE. « An Efficient Technique to ensure the Logical Consistency of Interacting Knowledge Bases ». *International Journal of Cooperative Information Systems*, 6(1):27–36, 1997.

[Mazure *et al.* 1998] Bertrand MAZURE, Lakhdar SAÏS, et Éric GRÉGOIRE. « Boosting Complete Techniques thanks to local search methods ». *Annals of Mathematics and Artificial Intelligence*, 22:319–331, 1998.

[McAllester 1980] D.A. MCALLESTER. « An Outlook on Truth Maintenance ». AI Memo551, August 1980. MIT AI Laboratory.

[McAllester *et al.* 1997] David A. MCALLESTER, Bart SELMAN, et Henry A. KAUTZ. « Evidence for Invariants in Local Search ». Dans *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, pages 321–326, août 1997.

[Minton *et al.* 1990] Steven MINTON, Marc D. JOHNSTON, Andrew B. PHILIPS, et Philip LAIRD. « Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method ». Dans *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI'90)*, 1990.

[Mitchell *et al.* 1992] David MITCHELL, Bart SELMAN, et Hector J. LEVESQUE. « Hard and Easy Distributions of SAT Problems ». Dans *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 459–465, 1992.

[Moinard 1994] Y. MOINARD. « Revision and nonmonotonicity ». *Int. Journ. of Intelligent Systems*, 9, 1994.

[Monien & Speckenmeyer 1985] B. MONIEN et E. SPECKENMEYER. « Solving satisfiability in less than 2^n steps ». *Discrete Applied Math.*, 10:287–295, 1985.

[Morris 1993] P. MORRIS. « The Break Out Method For Escaping From Local Minima ». Dans *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93)*, pages 40–45, 1993.

[Nebel 1992] B. NEBEL. Syntax-Based Approaches to Belief Revision. Dans P. GÄRDENFORS, éditeur, *Belief Revision*, Cambridge Tracts in Theoretical Computer Science 29, pages 52–88. Cambridge University Press, Cambridge, UK, 1992.

[Newell 1982] A. NEWELL. « The knowledge level ». *Artificial Intelligence*, 18:87–127, 1982.

[Oxusoff & Rauzy 1989] Laurent OXUSOFF et Antoine RAUZY. « *L'évaluation sémantique en calcul propositionnel* ». Thèse de doctorat, Groupe d'Intelligence Artificielle Université d'Ais-Marseille II, Faculté de Luminy, Marseille (France), janvier 1989.

[Papini 1991] O. PAPINI. « Revision in Propositional Calculus ». Dans Rudolf KRAUSE et Pierre SIEGEL, éditeurs, *Proceedings of Symbolic and Quantitative Approaches to Uncertainty (ECSQAU'91)*, volume 548 de *LNCS*, pages 272–276, Berlin, Germany, octobre 1991. Springer.

[Polya & Read 1987] G. POLYA et R.C. READ. *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds*. Springer-Verlag, 1987.

[Pretolani 1993] D. PRETOLANI. « Solving satisfiability problems: An algorithm implementation challenge? ». Dans *Working notes, 2nd DIMACS Challenge*, 1993. Extended abstract.

[Boy de la Tour & Demri 1995] T. BOY DE LA TOUR et S. DEMRI. « On the complexity of Extending Ground Resolution with Symmetry Rules ». Dans *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 289–295, 1995.

[Prover] PROVER. « A commercially packaged proof engine ». Compagny founded in 1989. http://www.prover.com/.

[Puget 1993] J.-F. PUGET. « On the Satisfiability of Symmetrical Constrainted Satisfaction Problems ». Dans J. KOMOROWSKI et Z. W. RAŚ, éditeurs, *Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems (ISMIS'93)*, volume 689 de *LNAI*, pages 350–361, Trondheim, Norway, juin 1993.

[Purdom 1990] P. PURDOM. « A survey of average time analyses of satisfiability algorithms ». *Information Processing*, 13(4), 1990.

[Ramsey 1930] F.P. RAMSEY. « On a Problem of Formal Logic ». Dans *London Math. Soc.*, pages 264–286, 1930.

[Rauzy 1994] Antoine RAUZY. « On the Complexity of the Davis & Putnam's Procedure on Some Polynomial Sub-Classes of SAT ». Rapport Interne 806-94, LaBRI, Université de Bordeaux I, 1994.

[Rauzy 1995a] Antoine RAUZY. « On the Random Generation of 3-SAT Instances ». Rapport Interne 1060-95, LaBRI, Université de Bordeaux I, 1995.

[Rauzy 1995b] Antoine RAUZY. « Polynomial restrictions of SAT: What can be done with an efficient implementation of the Davis and Putnam's procedure. ». Dans

U. MONTANARI et F. ROSSI, éditeurs, *Proceedings of the International Conference of Principle of Constraint Programming, CP'95*, volume 976 de *Lecture Notes in Computer Science*, pages 515–532. Springer Verlag, 1995.

[Rauzy *et al.* 1999] A. RAUZY, L. SAÏS, et L. BRSOUX. « Calcul Propositionnel : Vers une extension du formalisme ». Dans *JNPC'99*, pages 189–198, Lyon, 1999.

[Reiter & Mackworth 1989] R. REITER et A. MACKWORTH. « A logical framework for depiction and image interpretation ». *Artificial Intelligence*, 43(2):125–155, 1989.

[Régin & eds 1998] J.C. RÉGIN et W. Nuijtens EDS, éditeurs. *International Workshop on Non-binary constraints (ECAI'98)*, Brighton, 1998.

[Robinson 1965] J.A. ROBINSON. « A machine-oriented logic based on the resolution principle ». *Journal of the Association for Computing Machinery*, 12:23–41, 1965.

[Sabin & Freuder 1994] D. SABIN et E. FREUDER. « Contradicting conventional wisdom in constraint satisfaction ». Dans *ECAI'94*, pages 125–129, 1994.

[Saïs 1993] Lakhdar SAÏS. « *Étude des Symétries et de la Cardinalité en Calcul Propositionnel : Application aux algorithmes sémantiques* ». Thèse de doctorat, Université de Provence, Marseille (France), février 1993.

[Saïs 1994a] L. SAÏS. « Characterization of the set of models by means of symmetries ». Dans *Proceedings of the second Workshop on the Principles and Practice of Contraint Programming (PPCP'94)*, Orcas Island, Washington USA, May 2-4 1994.

[Saïs 1994b] L. SAÏS. « A Computational Study of DP with Symmetry on Hard Satisfiability Problems ». Dans *International Workshop On Experimental Evaluation of Reasoning and Search Methods (AAAI'94)*, pages 52–56, 1994.

[Saïs 1994c] L. SAÏS. « Finding non isomorphic solutions ». Dans P. JORRAND et V. SGUREV, éditeurs, *6th International Conference on AI : Methodology, Systems and Application (AIMSA'94)*, pages 35–44, Sofia Bulgaria, Sept 21-24 1994.

[Schiermeyer 1983] I. SCHIERMEYER. « Solving 3-Satisfiability in less than $O(1.579^n)$ steps ». *Lecture Notes in Computer Science*, 702:379–394, 1983.

[Schiermeyer 1996] I. SCHIERMEYER. « Pure literal lookahead: an $O(1.497^n)$ 3-Satisfiability algorithm ». Dans *Workshop on Satisfiability*, pages 63–72, Universita delgi Studi, Siena, 1996.

[Schiex & Verfaillie 1993] T. SCHIEX et G. VERFAILLIE. « Nogood Recording for Static and Dynamic Constraint Satisfaction Problems ». Dans *International Conference on Tools with Artificial Intelligence (ICTAI'93)*, pages 48–55, 1993.

[Schlipf *et al.* 1995] John S. SCHLIPF, Fred S. ANNEXSTEIN, John V. FRANCO, et R. P. SWAMINATHAN. « On finding solutions for extended Horn formulas ». *Information Processing Letters*, 54(3):133–137, 12 mai 1995.

[Schrag 1996] Robert C. SCHRAG. « Compilation for critically constrained knowledge bases ». Dans *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, pages 510–515, juillet 1996.

[Selman & Kautz 1991] Bart SELMAN et Henry A. KAUTZ. « Knowledge compilation using Horn approximations ». Dans *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI'91)*, pages 904–909, 1991.

[Selman & Kautz 1993] Bart SELMAN et Henry A. KAUTZ. « Domain-Independent Extensions to GSAT: Solving Large Structured Satisfiability Problems ». Dans *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 290–295, 1993.

[Selman 1995] Bart SELMAN. « Stochastic Search and Phase Transitions: AI Meets Physics ». Dans Chris S. MELLISH, éditeur, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 998–1002, San Mateo, 1995.

[Selman *et al.* 1992] Bart SELMAN, Hector J. LEVESQUE, et David MITCHELL. « GSAT: A New Method for Solving Hard Satisfiability Problems ». Dans *Proceedings of the*

Tenth National Conference on Artificial Intelligence (AAAI'92), pages 440–446, 1992.

[Selman *et al.* 1994] Bart SELMAN, Henry A. KAUTZ, et B. COHEN. « Noise strategies for improving local search ». Dans *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94)*, pages 337–343, 1994.

[Selman *et al.* 1997] Bart SELMAN, Henry A. KAUTZ, et David A. MCALLESTER. « Computational Challenges in Propositional Reasoning and Search ». Dans *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, volume 1, pages 50–54, Nagoya (Japan), août 1997.

[Shannon 1938] C.E. SHANNON. « A symbolic analysis of relay and switching circuits ». *Transactions AIEE*, 57:713–719, 1938.

[Sheeran & Stalmarck 2000] Mary SHEERAN et Gunnar STALMARCK. « A Tutorial on Stalmarck's Proof Procedure for Propositional Logic ». *Formal Methods in System Design, Kluwer Academic Pu*, 16(1):23–58, January 2000.

[Siegel 1987] Pierre SIEGEL. « *Représentation et Utilisation de la connaissance en calcul propositionnel* ». Thèse d'état, Université de Provence, GIA–Luminy, Marseille (France), 1987.

[Siegel 1990] Pierre SIEGEL. « Communication personnelle », 1990.

[Silva & Sakallah 1996] João P. Marques SILVA et Karem A. SAKALLAH. « GRASP – A New Search Algorithm for Satisfiability ». Dans *IEEE/ACM International Conference on Computer-Aided Design*, November 1996.

[Silva & Sakallah 2000] João P. Marques SILVA et Karem A. SAKALLAH. « Boolean Satisfiability in Electronic Design Automation ». Dans *in Proceedings of the IEEE/ACM Design Automation Conference, DAC'00*, June 2000.

[Silva & e Silva 1999] João P. Marques SILVA et Luís Guerra e SILVA. « Algorithms for Satisfiability in Combinational Circuits Based on Backtrack Search and Recursive Learning ». Dans *Proceedings of the XII Symposium Integrated Circuits and Systems Design (SBCCI'99)*, 1999.

[Slaney 1994] John SLANEY. « The crisis in finite mathematics: Automated reasoning as cause and cure ». Dans Alan BUNDY, éditeur, *Proceedings of the 12th International Conference on Automated Deduction*, volume 814 de *LNAI*, pages 1–13, Berlin, 1994.

[Slaney *et al.* 1993] J. SLANEY, M. FUJITA, et M. STICKEL. « Automated Reasoning and Exhaustive Search: Quasigroup Existence Problems ». *Computers and Mathematics with Applications*, 1993.

[Smith *et al.* 2000] Barbara SMITH, Kostas STERGIOU, et Toby WALSH. « Using Auxiliary Variables and Implied Constraints to Model Non-binary Problems ». Dans *AAAI'2000*, 2000.

[Sosic & Gu 1991] R. SOSIC et J. GU. « 3 000 000 queens in less than one minute ». *Sigart Bulletin (AI), ACM press*, 2(2):22–24, 1991.

[Stallman & Sussman 1977] R.M. STALLMAN et G.J. SUSSMAN. « Forward Reasoning and Dependency-Directed backtraking in a System for Computer AIded Circuit Analysis ». *Artificial Intelligence*, 9:135–196, October 1977.

[Tovey 1984] Craig A. TOVEY. « A Simplified NP-complete Satisfiability Problem ». *Discrete Applied Mathematics*, 8:85–89, 1984.

[Tseitin 1968] G.S. TSEITIN. « On the complexity of derivations in the propositional calculus ». Dans H.A.O. SLESENKO, éditeur, *Structures in Constructives Mathematics and Mathematical Logic, Part II*, pages 115–125, 1968.

[Uribe & Stickel 1994] T. E. URIBE et M. E. STICKEL. « Ordered Binary Decision Diagrams and the Davis-Putnam Procedure ». *Lecture Notes in Computer Science*, 845:34–??, 1994.

[Urquhart 1987] A. URQUHART. « Hard Examples for Resolution ». *ACM*, 34:209–219, 1987.

[Van Gelder & Tsuji 1996] Allen VAN GELDER et Yumi K. TSUJI. « Satisfiability Testing with More Reasoning and Less Guessing ». Dans D.S. JOHNSON et M.A. TRICK, éditeurs, *Second DIMACS Challenge*, DIMACS Series in Discrete Mathematics and Theorical Computer Science, American Mathematical Society, pages 559–866, 1996.

[van Gelder 1999] van GELDER. « Complexity Analysis of Propositional Resolution with Autarky Pruning ». *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 96, 1999.

[Vempaty 1992] Nageshwara Rao VEMPATY. « Solving Constraint Satisfaction Problems Using Finite State Automata ». Dans William SWARTOUT, éditeur, *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI'92)*, pages 453–458, San Jose, CA, juillet 1992. MIT Press.

[Vidal 2000] V. VIDAL. « Contribution à la planification par compilation de plans ». Rapport Technique 00/03-R, IRIT, Université Paul Sabatier, 2000.

[Vlach 1993] F. VLACH. « Simplification in a Satisfiability Checker for VLSI Applications ». *Journal of Automated Reasoning*, 10:115–136, 1993.

[Walser 1997] Joachim P. WALSER. « Solving Linear pseudo-Boolean Constraint Problems with Local Search ». Dans *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI'97)*, pages 269–274, Menlo Park, 1997.

[Walser 1999] Joachim Paul WALSER. *Integer optimization by local search: a domain-independent approach*, volume 1637 de *Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence*. Springer-Verlag Inc., New York, NY, USA, 1999.

[Zhang & Hsiang 1994] H. ZHANG et J. HSIANG. « Solving open quasigroup problems by propositional reasoning ». Dans *International Computer Symposium*, Hsinchu, Taiwan, 1994.

[Zhang & Zhang 1995] J. ZHANG et H. ZHANG. « SEM: a System for Enumerating Models ». Dans *International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 11–18, Montreal, August 11-18 1995.

[Zhang 1997] H. ZHANG. « SATO: An Efficient Propositional Prover ». Dans *International Conference on Automated Deduction, CADE'97*, 1997.

[Zhang *et al.* 1997] Dongmo ZHANG, Shifu CHEN, Wujia ZHU, et Hongbing LI. « Nonmonotonic Reasoning and Multiple Belief Revision ». Dans *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, volume 1, pages 95–101, 1997.

Deuxième partie

Curriculum Vitae

Chapitre 5

État civil et cursus

Sommaire

5.1 Notice individuelle 59
5.2 Diplômes 59
5.3 Activités professionnelles 60

5.1 Notice individuelle

Lakhdar SAÏS
Né le 17 octobre 1966 à Ighil-Boukiassa (Algérie)
Nationalité algérienne, marié, 2 enfants.

Fonction actuelle : Maître de conférences (section 027) depuis le 01/09/1994
- titulaire d'une prime d'encadrement doctorale depuis 01/10/1999
- 1er échelon de la 1ère classe depuis 01/09/1998
- 2ème échelon de la 2ème classe depuis 01/10/1994

Adresse professionnelle :
Centre de Recherche en Informatique de Lens
IUT de LENS, Université d'Artois
Rue de l'Université, SP-16
62307 Lens Cedex 07

Tél. : (+33) 3 21 79 32 76 78 *Fax* : (+33) 3 21 79 32 72 *Mél.* : sais@cril.univ-artois.fr

5.2 Diplômes

1990 – 1993 Doctorat d'Université (Spécialité informatique)
Université de Provence, Aix-Marseille I,
Laboratoire d'Informatique de l'Université de Provence,

Titre : Étude des symétries et de la cardinalité en calcul propositionnel :
Application aux algorithmes sémantiques
Directeur de thèse : Pierre SIEGEL
Soutenu le 5 février 1993 ; mention très honorable et félicitations du jury

1989 – 1990 Diplôme d'Étude Approfondie d'Informatique et Mathématique
(option Intelligence Artificielle)
Groupe Intelligence Artificielle
Université Aix-Marseille II (Luminy)
Mémoire : Étude des symétries en calcul propositionnel
Responsable : Professeur Pierre SIEGEL
Mention : Bien

1983 – 1984 Tronc commun sciences exactes

1984 – 1988 Diplôme d'ingénieur d'état en informatique
Institut National d'Enseignement Supérieur en Informatique
Université de Tizi-Ouzou, Algérie
Mémoire : Etude des problèmes de programmation linéaire à grand nombre de paramètres
Responsable : Professeur Mohand Said AIDENE
Mention : Très Bien

1982 – 1983 Baccalauréat scientifique (spécialité Mathématique)
Lycée Chihani Bachir Azazga, Tizi-Ouzou
Mention : Passable

5.3 Activités professionnelles

1994 – . . . Maître de Conférences
IUT de Lens, Université d'Artois

1992 – 1994 Attaché Temporaire d'Enseignement et de Recherche
Université de Provence, Marseille

1990 – 1992 Vacataire en informatique
Université de Provence, Marseille

1988 – 1989 Chargé d'enseignement en mathématiques, classes terminales scientifique
Lycée technique d'Azazga, Tizi-ouzou, Algérie
Moniteur d'informatique à la maison de jeunes d'Azazga

Chapitre 6

Activités de recherche

Sommaire

6.1 Encadrement de la recherche et animation scientifique 62
6.1.1 Activités d'encadrement 62
6.1.2 Participation à des jurys de thèse 62
6.1.3 Participation à des comités de programme et d'organisation 63
6.1.4 Participation à des projets de recherche 63
6.1.5 Relecture d'articles 63
6.2 Publications 63
6.2.1 Thèse 63
6.2.2 Édition d'actes de conférences 64
6.2.3 Ouvrages et chapitres d'ouvrages 64
6.2.4 Articles publiés dans des revues d'audience internationale avec comité de rédaction 64
6.2.5 Communications à des manifestations internationales avec comité de sélection 64
6.2.6 Conférences d'audience nationale avec comité de sélection 66
6.2.7 Tutoriels 67
6.2.8 Posters 67
6.2.9 Articles collectifs 67
6.2.10 Colloques et journées sur invitation 67
6.2.11 Rapports internes 68
6.2.12 Articles soumis 68
6.2.13 Travaux en cours 68
6.2.14 Rapport d'activités 68
6.2.15 Séminaires 68

Mes travaux de recherche s'articulent autour des thèmes suivants :

- représentation des connaissances et démonstration automatique
- satisfiabilité d'une formule booléenne mise sous forme normale conjonctive (SAT)
- problèmes de satisfaction de contraintes (CSP)
- compilation des bases de connaissances
- logiques non monotones
- révision des bases de connaissances propositionnelles

– démonstration en logique du premier ordre
– travail coopératif et fusion de connaissances
– validation des systèmes à base de connaissances

6.1 Encadrement de la recherche et animation scientifique

6.1.1 Activités d'encadrement

Thèses (co-encadrées et sous la direction du Pr. Éric Grégoire)

– Bertrand MAZURE, septembre 95 à janvier 99
Titre : De la Satisfaisabilté à la compilation de Bases de Connaissances Propositionnelles
Mention : Très Honorable
Fonction actuelle : Maître de conférences à l'Université d'Artois (Faculté des Sciences Jean Perrin) depuis le 01/09/99
– Laure BRISOUX, septembre 96 à décembre 99
Titre : Satisfaisabilité Propositionnelle en Informatique : Aspects Algorithmiques et Extensions du Formalisme
Mention : Très Honorable
Fonction actuelle : Maître de conférences à l'Université de Picardie Jules Verne depuis le 01/09/00

Mémoires de DEA (co-encadrés avec Éric GRÉGOIRE)

– Bertrand MAZURE, mémoire de DEA, année 94-95, « Expérimentations, analyse et amélioration des méthodes de résolution du problème SAT »
– Laure BRISOUX, mémoire de DEA, année 95-96, « De la logique des propositions au premier ordre fini »

6.1.2 Participation à des jurys de thèse

– ANBULAGAN, Thèse de l'Université de Technologie de Compiègne
Titre : Hypothèse de contrainte : une explication de la réussite de l'heuristique UP dans la résolution des problèmes de satisfiabilité des expressions booléennes
Date de soutenance : juin 1998
– David MARTINEZ, Thèse de l'École Nationale Supérieure de l'Aéronautique de Toulouse,
Titre : Résolution interactive de problèmes de satisfaction de contraintes
Date de soutenance : décembre 98
– Bertrand MAZURE, Thèse de l'Université d'Artois,
Titre : De la satisfaisabilté à la compilation de bases de connaissances propositionnelles
Date de soutenance : janvier 99
– Laure BRISOUX, Thèse de l'Université d'Artois,
Titre : Satisfaisabilité propositionnelle en informatique : aspects algorithmiques et extensions du Formalisme
Date de soutenance : décembre 99

6.1.3 Participation à des comités de programme et d'organisation

- Membre du comité de programme des « 5èmes Journées Nationales sur la résolution Pratique des problèmes NP-Complets », JNPC'99, Lyon, 1999
- Président du comité de programme et membre du comité d'organisation des « 4ème Journées nationales sur la résolution pratique des problèmes NP-Complets », JNPC'98, Ecole des Mines de Nantes, 27-29 Mai 1998.
- Président du comité de programme et membre du comité d'organisation du « Workshop on advances in propositional deduction », ECAI'96, Budapest, 1996
- Membre du comité de programme des « 3èmes Journées Nationales sur la résolution Pratique des problèmes NP-Complets », JNPC'97, Rennes, 1997
- Membre du comité de programme « 2ème Rencontre des Jeunes Chercheurs en IA », RJCIA'94, Marseille, 1994

6.1.4 Participation à des projets de recherche

- Projet du PRC IA « Booléens : Algorithmes et Heuristiques pour L'Intelligence Artificielle» (BAHIA), Période 92-94.
- Projet du PRC IA « Classes polynomiales », Période 92-94.
- Groupe de Travail du PRC IA « aspects algorithmiques de la résolution de problèmes exprimés à l'aide de contraintes », période 95-96.
- projets de recherche GANYMÈDE et GANYMÈDE II, Contrat de plan État/Région Nord–Pas-de-Calais sur le thème : «Communication Avancée et Activités Coopératives ».

6.1.5 Relecture d'articles

- International Joint Conference on Artificial Intelligence (IJCAI), 1997 et 1999.
- International Conference on Constraint Programming (CP), 1997.
- European Conference on Artificial Intelligence (ECAI), 1998.
- International Symposium on Theoritical Aspects of Computer Science (STACS), 1999 et 2000.
- Congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA), 1996.
- Journées Nationales sur la résolution Pratique de problèmes NP-Complets (JNPC), 1997 à 1999.
- Conference Nationale des Jeunes Chercheurs en Intelligence Artificielle (RJCIA), 1994.
- Revue d'Intelligence Artificielle (RIA), 1998.

6.2 Publications

6.2.1 Thèse

[1] Saïs L., "Etude des symétries et de la cardinalité en calcul propositionnel : applications aux algorithmes sémantiques" , Thèse de doctorat de l'université de Provence, Marseille, 5 février 1993.

6.2.2 Édition d'actes de conférences

[2] Saïs L. (ed.), Actes des 4ème journées nationales sur la résolution pratique des problèmes NP-Complets, Ecole des mines de Nantes, Mai 1998.

6.2.3 Ouvrages et chapitres d'ouvrages

[3] Bessant B., Grégoire E., Marquis P., Saïs L., "Iterated Syntax-Based Revision in a Nonmonotonic Setting" , in: M.-A. Williams and H. Rott (eds.), Frontiers in Belief Revision, Kluwer Academic Publishers, 2000 (sous presse).

[4] Grégoire É., Saïs L., "Practical Inconsistency Management for Critical-Tasks Decision Support Systems", in: D. Ruan (ed), Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry, World Scientific, Singapour, pp. 384-391, 1998.

6.2.4 Articles publiés dans des revues d'audience internationale avec comité de rédaction

[5] Brisoux L., Grégoire É., Saïs L., "Checking depth-limited consistency and inconsistency in knowledge-based systems", Int. Journ. of Intelligent Systems, 2000 (sous presse).

[6] Grégoire É., Saïs L., "Practical inconsistency management for critical-tasks decision-support systems ", Int. Journ. of General Systems, vol. 29, n 1, pp. 123-140, 2000.

[7] Mazure B., Saïs L., Grégoire É., "Boosting complete techniques thanks to local search", Annals of Mathematics and Artificial Intelligence, vol.22, pp. 319-322, 1998.

[8] Mazure B., Saïs L., Grégoire É., "An efficient technique to ensure the logical consistency of cooperative agents", Int. Journ. of Cooperative Information Systems, vol. 6, no 1, pp. 27-36, 1997.

[9] Benhamou B., Saïs L. "Tractability through symmetries in propositional calculus", Journal of Automated Reasoning, 12 : 89-102, 1994.

6.2.5 Communications à des manifestations internationales avec comité de sélection

- Congrès internationaux, avec comité de sélection et actes :

[10] Chmeiss A., Saïs L., "About Local consistency in Solving CSPs", Int. Conference on Tools and Artificial Intelligence ICTAI'00, Vancouver, Canada, 2000.

[11] Grégoire É., Saïs L., "Modelling and checking complex discrete critical system", Proc. of the Int.

Conf. on Modelling and Simulation (MS'2000), R. Berriel, V. Hernandez, R. Montenegro and J. Rocha (eds.), pages 233-240, Las Palmas de Gran Canaria, Spain, 25-27 septembre 2000.

[12] Brisoux L., Grégoire É., Saïs L., "Improving backtrack search for SAT by means of redundancy", Proc. of the 11th Int. Symposium on Methodologies for Intelligent Systems (ISMIS'99), Z.W. Ras and A. Skowron (eds.), Varsovie, Pologne, LNCS 1609, Springer, pp. 301-309, juin 1999.

[13] Mazure B., Saïs L., Grégoire É., " System Description : CRIL Platform for SAT ", Proc. of the 15th Int. Conf. on Automated Deduction (CADE-15), C. Kirchner et H. Kirchner (eds.), Lindau, Allemagne, LNCS 1421,Springer, pp. 124-128, juillet 1998.

[14] Bessant B., Grégoire É., Marquis P., Saïs L., " Combining nonmonotonic reasoning and belief revision: a practical approach ", Proc. 8th Int. Conf. on Artificial Intelligence - Methodology, Systems, Applications (AIMSA'98), F. Giunchiglia (ed.), Sozopol, Bulgarie, LNCS 1480, Springer, pp. 115- 128, septembre 1998 (**Best Paper Award**).

[15] Boufkhad Y., Grégoire É., Marquis P., Mazure B., Saïs L. " Tractable Cover Compilations " Proc. of the Int. Joint Conf. on Artificial Intelligence(IJCAI'97), Nagoya, Japon, pp. 1201-1206, août 1997.

[16] Grégoire É., Mazure B., Saïs L., "A comparison of two approaches to inconsistency detecting" Proc. European Symp. on Intelligent Techniques, 20- 21 March, Bari, Italy, 1997.

[17] Mazure B., Saïs L., Grégoire É. " Local search for common-sense reasoning " In D. Gabbay (ed.), Proc. of the Int. Joint Conference on qualitative and quantitative practical reasoning, LNCS 1244,Springer, Bad Honnef, Allemagne, pp. 122-130, juin 1997.

[18] Mazure B., Saïs L., Grégoire É. " Tabu Search for SAT ", Proc. of the 14th Nat. Conf. on Artificial Intelligence, AAAI'97, pp. 281-285, Providence, Rhode Island, USA, July 27-31, 1997.

[19] Mazure B., Saïs L., Grégoire É., "Detecting logical inconsistencies", Proceedings of the Int. Symposium on Artificial Intelligence and Mathematics, pp. 116-121, Fort Lauderdale, Florida U.S.A., January 1996.

[20] Mazure B., Saïs L., Grégoire É., "A powerful heuristic to locate inconsistent kernels in Knowledge-Based Systems",Proceedings of the 6th Int. Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU'96, Vol. 3, pp. 1265-1269, Granada Spain, July 1996.

[21] Mazure B., Saïs L., Grégoire É., "SUN: A multistrategy platform for SAT", First Int. Competition and Symp. on Satisfiability Testing , Beijing China, March 1996.

[22] Saïs L., "Finding non isomorphic solutions", In P. Jorrand and V. Sgurev, editors, Proceedings of the 6th Inter. Conference on AI : Methodology, Systems and Application AIMSA'94, pp. 35-44, Sofia Bulgaria, Sept 21-24, 1994.

[23] Benhamou B., Saïs L., "Two proof procedures for cardinality based language in propositional calculus", In P. Enjalbert, E. W. Mayr, K. W. Wagner (eds.), in Proc. of the 11th Int. Symposium on Theoritical Aspects of Computer Science STACS'94, LNCS 775, pp. 71-82, Caen France, Feb 24-26, 1994.

[24] Benhamou B., Saïs L., "Theoretical study of symmetries in propositional calculus and applications". In D.Kapur (ed.), in Proc. of the 11th International Conference on Automated Deduction (CADE'11), LNCS 607, pp. 281-294, New York, June 1992.

- Workshop à des congrès internationaux, avec comité de sélection et actes :

[25] Bessant B., Grégoire É., Marquis P., Saïs L., “Syntax-based belief revision through local search “, Proc. International Belief Revision Workshop, Trente, juin 1998.

[26] Grégoire E., Mazure B., Saïs L., “ Logically-complete local search for propositional nonmonotonic knowledge bases “, Proc. of the 7th Int. Workshop on Nonmonotonic Reasoning, I. Niemela et T. Schaub (eds.), pp.37-45, Trente, juin 1998.

[27] Brisoux L., Saïs L., Grégoire E., “Validation of knowledge-based systems by means of stochastic search”, Proc. DEXA Workshop on Verification, Validation and Integrity Issues in Expert and Database Systems, R.R. Wagner (ed.), Vienne, IEEE Computer Press, pp. 41-46, septembre 1998.

[28] Saïs L., “SAT : Experiments meet theory”, In T. Walsh (ed.), In Proceedings of the ECAI’96, Workshop on Empirical AI, Budapest Hungary, August 1996.

[29] Grégoire É., Saïs L., “Inductive reasoning is sometimes deductive”, In P. Falch (ed.), in Proc. of the ECAI’96 Workshop on Abductive and Inductive Reasoning, pp. 36-39, Budapest Hungary, August 1996.

[30] Mazure B., Saïs L., Grégoire É, “Twsat: a new local search algorithm for SAT. performance and analysis”, Proceedings of the CP’95 Workshop On Solving Really Hard Problems, pp. 127-130, Cassis France, September 1995.

[31] Saïs L., “Characterization of the set of models by means of symmetries”, Proceedings of the second Workshop on the Principles and Practice of Contraint Programming(PPCP’94), Orcas Island, Washington USA, May 2-4 1994.

[32] Saïs L., Génisson R., “Some Ideas on Random Generation of K-Sat Instances“, J.M. Crawford and B. Selman (ed.), in Proc. of the AAAI’94 Workshop On Experimental Evaluation of Reasoning and Search Methods, pp. 91-93, July 31-August 1, Seattle, USA, 1994.

[33] Saïs L. “A Computational Study of DP with Symmetry on Hard Satisfiability Problems”, in Proc. of the AAAI’94 Workshop On Experimental Evaluation of Reasoning and Search Methods(AAAI- 94), pp. 52-56, July 31-August 1, Seattle, USA, 1994.

[34] Saïs L., Génisson G., “Towards an understanding of hard satisfiability problems”, In Geoff Suttcliffe (ed.), in Proc. of the Workshop On Experimental Evaluation of Automated Theorem Proving Systems(CADE’12), pp. 24-29, June 28-July 1, Nancy, France, 1994.

[35] Benhamou B., Saïs L., Siegel P. “Dealing with symmetries in propositional calculus”, Workshop on Tractable Reasoning (AAAI’92), pp. 1-5, San Jose, California, 12-17 July 1992.

[36] Benhamou B., Saïs L., Siegel P. “Dealing with cardinality formulas in propositional calculus” Workshop on Tractable Reasoning (AAAI’92), pp. 6-12, San Jose, California, 12-17 July 1992.

[37] Benhamou B., Saïs L., “Study of symmetries in propositional calculus”, Intern. Workshop on Computer Science, Annaba, pp. 78-102, 16-18 Décembre 1991.

6.2.6 Conférences d’audience nationale avec comité de sélection

[38] Brisoux L., Saïs L. et Grégoire É., “Recherche locale : vers une exploitation des propriétés structurelles”, Actes des 6ièmes Journées Nationales sur la Résolution Pratique des Problèmes NP-complets (JNPC-00), Marseille, pp. 243-244, 2000.

[39] Rauzy A., Saïs L. et Brsoux L., “Calcul Propositionnel : Vers une extension du formalisme”, Actes des 5ièmes Journées Nationales sur la Résolution Pratique des Problèmes NP-complets (JNPC- 99), Lyon , pp. 189-198, 1999.

[40] Brisoux L., Saïs L. et Grégoire E., “Mieux exploiter les échecs au sein des arbres de recherche à la Davis et Putnam”, Actes des 4ièmes Journées Nationales sur la Résolution Pratique des Problèmes NP-complets (JNPC-98), Nantes, pp. 31-39, 1998.

[41] Mazure B., Saïs L., Grégoire É. “Deux approches pour la résolution du probleme SAT”, 2ème conférence nationale sur résolution pratique des problèmes NP-Complets (CNPC’96), pp. 103-114, Dijon France, Mars 1996.

[42] Benhamou B., Saïs L., “Formules de cardinalité et symétries en calcul propositionnel”, Rencontres nationales des jeunes chercheurs en Intelligence Artificielle (RJCIA’92), pp. 242-256, 7-9 Septembre, Rennes 1992.

6.2.7 Tutoriels

[43] Crawford J., Saïs L., “Symmetries”, First Int. Competition and Symposium on Satisfiability Testing , Beijing China, March 1996.

6.2.8 Posters

[44] Mazure B., Saïs L., Grégoire É., “Local search for computing normal circumstances models”, B. Reush (ed.), in Proc. of the computational Intelligence Conf. (theory and application), LNCS 1226, Springer, Dortmund, pp. 565-570, avril 1997.

6.2.9 Articles collectifs

[45] Groupe Bahia, “Etude Comparative de Trois Formalismes en Calcul Propositionnel”, Actes des 5èmes journées nationales du PRC-GDR Intelligence Artificielle. Teknea, p. 125-157, 1-3 Février, Nancy 1995.

[46] Projet Inter-PRC : “Classes Polynomiales : Travaux et Résultats”, Actes des 5èmes journées nationales du PRC-GDR Intelligence Artificielle. Teknea, pp. 3-28, 1-3 Février, Nancy 1995.

[47] Groupe Bahia,. “Etude Comparative de Trois Formalismes en Calcul Propositionnel”, Actes des 4èmes journées nationales du PRC-GDR Intelligence Artificielle. Teknea, p. 239-318, 19-21 Octobre, Marseille 1992.

6.2.10 Colloques et journées sur invitation

[48] Benhamou B., Saïs L., Siegel P., “Détection et utilisation des symétries en calcul propositionnel”. Journée meta-inference Pôle "D" du PRC-IA, Paris (Laforia) 3 Décembre 1990.

[49] Benhamou B., Saïs L., Siegel P., “Some results on symmetries in propositional calculus”, A. Colmerauer (ed.) In Proc. Workshop on Constraint and Logic Programming (WCLP’92), Luminy, Marseille 1992.

6.2.11 Rapports internes

[50] Benhamou B., Saïs L., “Etude des symétries en calcul propositionnel”, Rapport de Recherche de l’Université de Provence, Marseille, MAIUP 91 - 01 Janvier 1991.

[51] Benhamou B., Saïs L., “Cardinality formulas in propositional calculus”, Rapport de recherche de l’Université de Provence, Marseille, LIUP 92 - 01 Janvier 1992.

6.2.12 Articles soumis

[soumis] Grégoire É., Mazure B., Saïs L., “Using failed local search as an oracle for tackling harder optimization problems more efficiently” (soumis au Journal of Heuristics).

6.2.13 Travaux en cours

[en cours] Grégoire É., Marquis P., Mazure B., Saïs L., “Sur la généralisation du théorème de partition du modèle”

[en cours] Saïs L., “Extension de la substituabilité de voisinage”.

[en cours] Bessiere C., Chmeiss A., Saïs L., “Autour de l’utilisation de la singleton arc consistance (SAC)”.

[en cours] Bessiere C., Chmeiss A., Saïs L., ”CSP: heuristiques de choix de variables et de valeurs”.

6.2.14 Rapport d’activités

– rapport de synthèse sur les activités de recherche du GT12 Axe algorithme, Mai 2000.

6.2.15 Séminaires

– “Problème SAT: aspects algorithmiques et extension du formalisme”, LRI, Université d’Orsay,3 décembre 1999.
– “Résolution du problème SAT”, groupe NOn Monotone, LIFL, 1998.

- "Problèmes de Satisfaction de Contraintes(CSP)/ Satisfaibilité (SAT) : Survol et liens", séminaire de DEA LIFL, décembre 1999.
- "Autour de l'influence du codage en calcul propositionnel", exposé à la réunion du thème 1 du GT12, IRIT, Toulouse, 26 janvier 2000.
- "Problèmes de satisfactions de contraintes", séminaire de DEA LIFL, novembre 2000.

Chapitre 7

Activités administratives

Sommaire

7.1 Mandats électifs 70
7.2 Responsabilités diverses 70

7.1 Mandats électifs

- Membre élu du conseil scientifique de l'Université d'Artois (1997 - . . .)
- Membre extérieur de la commission de spécialistes, 27ème section Université d'Amiens(1998 - . . .)
- Membre extérieur de la commission de spécialistes, 27ème section Université de Valenciennes(1999 - . . .)
- Membre de la commission de documentation de l'Université d'Artois (1997 - 1999).

7.2 Responsabilités diverses

- *Enseignement* :
  - Responsable des forums carrières (depuis septembre 1999).
  - Responsable du suivi des anciens étudiants du département informatique (depuis septembre 1999).
  - Gestion et administration du réseau d'enseignement sous Windows-NT(année 1997).
  - Responsable de la gestion d'une salle TP sous Unix (année 1995).
- *Recherche* :
  - Responsable des séminaires au Centre de Recherche en Informatique de Lens (1995-2000).
  - Co-responsable avec Philippe JÉGOU (Aix-Marseille III) du groupe de travail GT1.2 « axe algorithme » du GdR-I3, depuis mars 1999.
  - Responsable de la maintenance du site web du GT1.2 du GdR I3.

Chapitre 8

Activités d'enseignement

Sommaire

8.1 Synopsis 71
8.2 Contenu détaillé 72
8.3 Autres activités liées à l'enseignement 74

8.1 Synopsis

- *Public* :
  - 2ème cycle:
    - Maîtrise d'Informatique.
    - Maîtrise d'Ingénierie Mathématique.
    - Licence d'Informatique.
    - Licence de Mathématique.
  - 1er cycle:
    - DUT Informatique.
    - DUT Gestion des Entreprises et des Administrations.
    - DUT Techniques de Commercialisation.
    - DEUG MIAS, SM, SV.
  - Formation continue.
  - Secondaire: Classes terminale scientifique.
- *Matières enseignées* :
  - Structures de données avancées (responsable du cours).
  - Algorithmique et programmation.
  - Infographie.
  - Architecture des ordinateurs.
  - Réseaux.
  - Génie logiciel et programmation orientée objet.
  - Programmation Logique (avec contraintes) (responsable du cours).
  - Bureautique (traitement de texte, tableur, bases de données).

  - Outils logiques pour l'informatique
- *Langages de programmation utilisés* : Pascal, C, C++, Java, Scheme, Prolog, Sicstus (Prolog avec contraintes).

8.2 Contenu détaillé

2000 – 2001 Maître de conférences
IUT de Lens, Département Informatique
1ère Année (DUT Info) :
Structures de données avancées et langage C++ (38h COURS, 51h TD et 17h TP)
Architecture des ordinateurs (26h TD, 16h TP Orcad et assembleur)
Reseaux (TD et TP)
Année spéciale :
Structures de données avancées et langage C++ (23h COURS, 34h TD)

1999 – 2000 Maître de conférences
IUT de Lens, Département Informatique
1ère Année (DUT Info) :
Structures de données avancées et langage C (34h COURS, 26h TD)
Algorithmique et programmation Java (51h TD, 34h TP JAVA)
2ème Année (DUT Info) :
Génie logiciel et programmation orientée objet (30h TD, 18h TP C++)

1998 – 1999 Maître de conférences
IUT de Lens, Département Informatique
Maîtrise d'Informatique (Faculté Jean-Perrin):
Programmation logique (10h COURS, 12h TP Prolog)
1ère Année (DUT Info) :
Structures de données avancées (34h COURS, 26h TD)
Algorithmique et programmation (51h TD, 34h TP Pascal et C)
2ème Année (DUT Info) :
Génie logiciel et programmation orientée objet (30h TD, 18h TP C++)
1ère et 2ème Année (DEUG SM, MIAS, Faculté Jean Perrin) :
Algorithmique et programmation (48h TD Scheme)

1997 – 1998 Maître de conférences
IUT de Lens, Département Informatique
Maîtrise d'Informatique (Faculté Jean-Perrin):
Programmation logique avec contraintes (10h COURS, 12h TP Sicstus Prolog)
1ère Année (DUT Info) :
Structures de données avancées (34h COURS, 26h TD)
Algorithmique et programmation (51h TD, 34h TP Pascal et C)
2ème Année (DUT Info) :
Génie logiciel et programmation orientée objet (30h TD, 18h TP C++)
1ère et 2ème Année (DEUG SM, MIAS, Faculté Jean Perrin) :
Algorithmique et programmation Pascal (50h TD)

1996 – 1997 Maître de conférences
IUT de Lens, Département Informatique
1èreAnnée (DUT Info) :

Structures de données avancées (34h COURS, 34h TD)
Algorithmique et programmation (136h TD, 102h TP Pascal et C)
2ème Année (DUT Info) :
Génie logiciel et programmation orientée objet (30h TD, 18h TP C++)
1ère et 2ème Année (DEUG SM, MIAS, Faculté Jean Perrin, Univ. Artois) :
Algorithmique et programmation Pascal (70h TD)

1995 – 1996 Maître de conférences
IUT de Lens, Département Informatique
1èreAnnée (DUT Info) :
Structures de données avancées (51h COURS)
Algorithmique et programmation (51h TD, 85h TP Pascal et C)
2ème Année (DUT Info) :
Génie logiciel et programmation orientée objet (30h TD, 18h TP C++)
1ère et 2ème année (DEUG SM, MIAS, Faculté J. Perrin) :
Algorithmique et programmation Pascal (80h TD)
2ème Année (DUT TC) : Bureautique (40h TD sur machines)
Formation continue : Système d'Information, Bureautique (30h Cours, 12h TP)

1994 – 1995 Maître de conférences
IUT de Lens, Département Informatique
1ère Année (DUT Info) :
Algorithmique et programmation (136h TD, 136h TP)
Structures de données avancées (34h TD)
2ème Année (DUT Info) :
Génie logiciel et programmation orientée objet (30h TD, 18h TP)
2ème année (DEUG SV, Fac J. Perrin) :
Algorithmique et programmation Pascal (50h TP)
2ème Année (DUT TC) : Bureautique(20h TD sur machines)
1ère et 2ème année (DUT GEA) : Bureautique (90h TD sur machines)

1993 – 1994 Attaché temporaire d'Enseignement et de Recherche
Laboratoire d'Informatique, Université de Provence, UFR-MIM
DEUG SSM 2ème année : informatique fondamentale(48h TD, 20h TP)
Licence d'Informatique:
outils logiques pour l'informatique (20h TD, 25h TP)
algorithmique 2(39h TD, 26h TP)
Formation continue (Informatique Scientifique et Technique) :
programmation en C (10h COURS, 8h TD, 18h TP)

1992 – 1993 Attaché temporaire d'Enseignement et de Recherche
Laboratoire d'Informatique, Université de Provence, UFR-MIM
DEUG SSM 1ère et 2ème année : informatique fondamentale (80h TD, 40h TP)
Licence de Mathématiques : algorithmique 1 (30h TD, 8h TP)
Licence d'Informatique : algorithmique 2 (39h TD, 26h TP)

1991 – 1992 Vacataire en informatique
Université de Provence
DEUG SSM 1ème année : informatique fondamentale (34h TD, 62h TP)
Maîtrise d'ingénierie mathématique : infographie (45h TP)
Licence d'informatique et Formation Continue : (125h TP)
Assistance Publique de Marseille : initiation à l'informatique (39h Cours, 26h TP)

1990 – 1991 Vacatire en informatique

	Université de Provence
	DEUG SSM 1ème année : informatique fondamentale(96h TD, 160h TP)
	Assistance Publique de Marseille : initiation à l'informatique (39h Cours, 26h TP)
1988 – 1989	Chargé d'enseignement en mathématiques, classes terminales
	Lycée technique d'Azazga, Tizi-OuZou, Algérie

8.3 Autres activités liées à l'enseignement

- Tuteur de Bertrand Mazure dans le cadre du monitorat (CIES),
- Suivis de stages d'étudiants de 2ème année DUT informatique (depuis 1994),
- Participation à l'organisation des journées portes ouvertes à l'IUT de Lens (1997),
- Participation aux jurys d'admission au DUT informatique (depuis 1994).
- Rédaction d'un polycopié, "Dossier d'analyse et de programmation : Étude d'un cas", année 2000,
- Encadrement de projets de synthèse en 2ème année:
  - Boite à outils sur les graphes,
  - Logique propositionnelle et applications,
  - Lens Internet : Mise en place de pages WEB pour la ville de Lens,
  - Projet Multimedia : Création d'une borne interactive présentant l'IUT de Lens,
  - Implémentation et évaluation de différentes méthodes de stockage d'informations,
  - Création d'un site web Axe algorithme du GdR I3,
  - Base de données des anciens étudiants.

Troisième partie

Publications jointes

Dans cette partie, je joins les publications (qui me semblent) les plus significatives. Le choix a été effectué de manière à représenter les différents aspects abordés dans la synthèse (cf. chapitre 2.2.1).

- **Exploitation des propriétés structurelles**
  - [1] Benhamou B., Saïs L.
    "Tractability through symmetries in propositional calculus",
    Journal of Automated Reasoning, 12 : 89-102, 1994.

- **Aspect codage et extension du formalisme**
  - [2] Benhamou B. et Saïs L.,
    "Two proof procedures for cardinality based language in propositional calculus",
    In P. Enjalbert, E. W. Mayr, K. W. Wagner (eds.), in proceedings. of the *11th International Symposium on Theoritical Aspects of Computer Science (STACS'94)*, LNCS 775, pages 71-82, Caen France, Feb 24-26, 1994.

  - [3] Rauzy A., Saïs L. et Brsoux L.,
    "Calcul Propositionnel : Vers une extension du formalisme",
    Actes des *5ièmes Journées Nationales sur la Résolution Pratique des Problèmes NP-complets (JNPC'99)*, Lyon , pages 189-198, 1999.

- **Aspect algorithmique** :
  - [4] Mazure B., Saïs L., Grégoire É.
    " Tabu Search for SAT ",
    in proceedings of the *14th National Conference on Artificial Intelligence (AAAI'97)*, pages 281-285, Providence, Rhode Island, USA, July 27-31, 1997.

  - [5] Mazure B., Saïs L. et Grégoire É.,
    "Boosting complete techniques thanks to local search",
    Annals of Mathematics and Artificial Intelligence, vol.22, pages 319-322, 1998.

- **Autour de SAT** :
  - [6] Chmeiss A. et Saïs L.,
    "About Local consistency in Solving CSPs",
    in proceedings of the *Twelfth International Conference on Tools and Artificial Intelligence (ICTAI'2000)*, Vancouver, Canada, 2000.

  - [7] Boufkhad Y., Grégoire É., Marquis P., Mazure B. et Saïs L.
    " Tractable Cover Compilations ",
    in proceedings of the *International Joint Conference on Artificial Intelligence(IJCAI'97)*, Nagoya, Japon, pp. 1201-1206, août 1997.

  - [8] Bessant B., Grégoire E., Marquis P. et Saïs L.
    "Iterated Syntax-Based Revision in a Nonmonotonic Setting"
    in : M.-A. Williams and H. Rott (eds.), *Frontiers in Belief Revision, Kluwer Academic Publishers*, 2000.

Tractability Through Symmetries in Propositional Calculus

Parue dans « *Journal of Automated Reasoning* », volume 12, pages 89–102, 1994.

Tractability Through Symmetries in Propositional Calculus

BELAÏD BENHAMOU and LAKHDAR SAÏS
L.I.U.P. Case G, Université de Provence,
3 Place Victor Hugo, F13331 Marseille cedex 3, France.
Tel: 91.10.61.08. e-mail: ({benhamou, sais}@gyptis.univ-mrs.fr*)*

(Received: 12 March 1992; accepted: 15 June 1993)

Abstract. Many propositional calculus problems - for example the Ramsey or the pigeonhole problems - can quite naturally be represented by a small set of first-order logical clauses which becomes a very large set of propositional clauses when we substitute the variables by the constants of the domain *D*. In many cases the set of clauses contains several symmetries, i.e. the set of clauses remains invariant under certain permutations of variable names. We show how we can shorten the proof of such problems. We first present an algorithm which detects the symmetries and then we explain how the symmetries are introduced and used in the following methods: SLRI, Davis and Putnam and semantic evaluation. Symmetries have been used to obtain results on many known problems, such as the pigeonhole, Schur's lemma, Ramsey's, the eight queen, etc. The most interesting one is that we have been able to prove for the first time the unsatisfiability of Ramsey's problem (17 vertices and three colors) which bas been the subject of much research.

Keywords: Theorem proving, propositional calculus, symmetries.

1. Introduction

The subject of this paper is the study of symmetries in propositional calculus in order to make the automated deduction algorithms more efficient. Using resolution as a base proof system for the propositional calculus, Tseitin [12] has shown how certain tricky mathematical arguments can be considered as short proofs for some propositional tautologies representing mathematical statements. He suggested increasing resolution by the principle of extension, which consists of the introduction of auxiliary variables to represent intermediate formulas so that the length of proof can be significantly reduced by manipulating these variables, rather than the formulas that they represent.

On the other hand, Krishnamurty [9] proposed the principle of symmetries, which allows us to recognize that a tautology remains invariant under certain permutation of variables names, and uses this information to avoid repeated independent derivations of intermediate formulas that are permutations of others. Consider the logical formulas associated with a concrete problem. In most cases, we use first-order logic to express this formula, using variables, constants, predicates. In other words, we use the usual mathematical language. From this set of first-order logic clauses we obtain the

propositional clauses by substituting the variables by constants of the domain D. Thus the previous set of clauses becomes a large set for many problems and contains several symmetries. The property of symmetry can lead to a shorter proof for the problem. However, the current methods of theorem proving do not take advantage of symmetry properties, and in ref. [9] no algorithm is presented neither for searching for symmetries nor for using them in a formal way.

The purpose of this study is the detection and the use of symmetries. First we describe the method which computes the symmetry on a given set of clauses. The algorithm is given in greater detail in refs [3] and [2]. Second, we explain how symmetries are introduced and combined with different automated deduction algorithms like Si-resolution [5], the Davis and Putnam procedure [6], and semantic evaluation [11]. Finally we apply SLRI [5] and the semantic evaluation methods with and without symmetry to some classical problems, such as the pigeonhole [4] and [15], Schur's lemma and Ramsey's problems [11] and the comparison shows that it is possible to obtain shorter proofs for hard propositional totaulogies and to significantly reduce the complexity of resolution in many cases. One of the most interesting results is that we have been able to prove the unsatisfiability of Ramsey's problem with 17 vertices and three colors, which had not been shown by a theorem proving program before.

2. Definitions and Notations

We shall assume that the reader is familiar with the propositional calculus. For a formal description see e.g. ref. [10]. Let V be the set of propositional variables, which are simply called variables. Variables will be distinguished from literals, which are variables with an assigned parity 1 or 0 (which means TRUE or FALSE, respectively). This distinction will be ignored whenever it is convenient but not confusing. For a propositional variable, p, there are two literals: p, the positive literal and $\neg p$, the negative one.

A clause is a disjunction of literals $(p_1, p_2, \ldots, p_n)$ such that no literal appears more than once and is denoted by $p_1 \vee p_2 \vee \ldots \vee p_n$. A system, S, of clauses is a conjunction of clauses. In other words we say that S is in the conjunctive normal form (CNF).

A truth assignment to a system of clauses S is a map I from the set of variable in S to the set {TRUE, FALSE}. Sometimes I can be considered as a set of literals which are true. If $I[p]$ is the value for the positive literal p then $I[\neg p] = 1 - I[p]$. The value of a clause $p_1 \vee p_2 \vee \ldots p_n$ in I is the maximum value of its literals in I. By convention we define the value of the empty clause ($n = 0$) to be FALSE. The value $I[S]$ of the system of clauses is TRUE if the value of each clause of S is TRUE; FALSE otherwise. We say that a

system of clauses, S, is satisfiable if there exists some truth assignment in which S takes the value TRUE; it is unsatisfiable otherwise. In the first case I is called a model of S. Note that a system which contain the empty clause is unsatisfiable.

It is well-known that for every propositional formula F there exists a formula F' in conjunctive normal form which is longer than F (see ref. [1]) and which is satisfiable iff F is satisfiable. More precisely, for every polynomial $p(n)$, there is a formula F such that any equivalent formula in CNF is of length at least $p(|F|)$, where $|F|$ is the length of F. If F is a clause c then $|c|$ is the number of literals in the clause c. In the following we will assume that the formulas are given in a conjunctive normal form.

3. Symmetries

First of all, let us define the concepts of permutations and symmetry, and prove significant properties that will enable us to improve the proof algorithms. For more detail we refer the reader to ref. [3].

A bijection map $\sigma : V \to V$ is called a permutation of variables. If S is a set of clauses, c a clause of S and a σ permutation of variables occuring in S, then $\sigma(c)$ is the clause obtained by applying σ to each variable of c and $\sigma(S) = \{\sigma(c)/c \in S\}$.

DEFINITION 1. A set, P, of literals is called *complete* if $\forall \ell \in P$; then $\neg\ell \in P$.

DEFINITION 2. Let P be a complete set of literals and S a set of clauses of which all literals are in P. Then a permutation σ defined on P ($\sigma : P \to P$) is called a *symmetry* of S if it satisfies the following conditions:

$$\forall \ell \in P, \sigma(\neg\ell) = \neg\sigma(\ell) \tag{1}$$

$$\sigma(S) = S \tag{2}$$

DEFINITION 3. Two literals (variables) ℓ and ℓ' are symmetrical in S notation ($\ell \sim \ell'$) if there exists a symmetry σ of S such that $\sigma(\ell) = \ell'$. A set $\{\ell_1, \ell_2, \ldots, \ell_n\}$ of literals is called a *cycle of symmetry* in S if there exist a symmetry σ defined on S, such that $\sigma(\ell_1) = \ell_2, \ldots, \sigma(\ell_{n-1}) = \ell_n, \sigma(\ell_n) = \ell_1$.

REMARK 4. *All the literals in a cycle of symmetry are symmetrical two by two.*

EXAMPLE 5. Let S be the following set of clauses :

$S = \{c_1 : a \vee \neg b, c_2 : c\}$

and σ the map defined on the complete set P of literals occurring in S:

$$\begin{aligned}\sigma: \quad & P \longrightarrow P \\ & a \longrightarrow \sigma(a) = \neg b \\ & \neg a \longrightarrow \sigma(\neg a) = b \\ & b \longrightarrow \sigma(b) = \neg a \\ & \neg b \longrightarrow \sigma(\neg b) = a \\ & c \longrightarrow \sigma(c) = c \\ & \neg c \longrightarrow \sigma(\neg c) = \neg c\end{aligned}$$

σ is a symmetry of S , a and $\neg b$ are symmetrical in S ($a \sim \neg b$).
$\sigma(S) = \{c_1 : \neg b \vee a, c_2 : c\} = S$.

REMARK 6. *$\sigma(S)$ is the set of clauses obtained from S by exchanging the literal positions in the same clause or exchanging the order of the clauses in S. Generally, the two previous operations can be applied simultaneously.*

- *the identity map is a symmetry;*
- *the inverse map of a symmetry is also a symmetry;*
- *the composition of symmetries is a symmetry.*

DEFINITION 7. Let P be a complete set of literals, σ a symmetry, I a truth assignment of P and S a set of clauses then

- I/σ is the *truth assignment* obtained by substituting every literal ℓ in I by $\sigma(\ell)$.
- S/σ is the *set of clauses* obtained with substituting every literal ℓ in S by $\sigma(\ell)$.

4. Symmetry Properties

If I is a model of S and σ a symmetry, then we can get another model of S by applying σ on the variables which appear in I. In the following propositions we assume that σ is a symmetry of the system of clauses S.

PROPOSITION 8. *I is a model of $S \Leftrightarrow I/\sigma$ is a model of S.*

Proof. Cf. ref. [3].

PROPOSITION 9. *Let ℓ be a literal, such that $\ell' = \sigma(\ell)$ and $I' = I/\sigma$. If I is such that $I[\ell] = 1$, then I' is such that $I'[\ell'] = 1$*

Proof. Direct consequence of Proposition 8: if $I[\ell] = 1$ then I is a model of ℓ or, I/σ is a model of $\sigma(\ell)$ (Proposition 8). Then I' is a model of ℓ', thus $I'[\ell'] = 1$.

PROPOSITION 10. *If ℓ has the value true in a model of S, then ℓ', such that $\ell' = \sigma(\ell)$, will have the value true in a model of S.*

Proof. Direct consequence of Proposition 9: if $\ell =$ true in a model I of S, then $I[\ell] = 1$, applying Proposition 9 we get $I'[\ell'] = 1$ such that $I' = I/\sigma$, then $\ell' =$ true in I' which is a model of S.

THEOREM 11. *Let ℓ and ℓ' be two literals of S. If $\ell \sim \ell'$ in S, then [ℓ has a model in S $\Leftrightarrow$ ℓ' has a model in S] (ℓ has a model in S if and only if there exists a model I of S such that $I[\ell] = 1$).*

Proof. ($\Rightarrow$) Suppose that ℓ has a model in $S \Leftrightarrow \exists I$ a model of S such that $I[\ell] = 1$. $\ell \sim \ell'$ in $S \Rightarrow \exists\sigma$ a symmetry of S such that $\sigma(\ell) = \ell'$. I is a model of $S \Leftrightarrow I/\sigma$ is a model of $S/\sigma = S$ (Proposition 8). We also have $\{\sigma(\ell) \in S$ and $I[\ell] = 1\} \Rightarrow \sigma(l) \in I/\sigma$ (Proposition 9). Then $\ell' \in I/\sigma \Rightarrow I/\sigma$ is a model of ℓ'.

($\Leftarrow$) Let $\ell' = \sigma^{-1}(\ell)$. The proof is then identical.

It is very important to draw a distinction between symmetric literals and equivalent literals. Equivalent literals have exactly the same models. But if ℓ and ℓ' are symmetrical then: [ℓ has a model in $S \Leftrightarrow \ell'$ has a model in S], and the two models are generally different, so it is easier to find symmetric literals than equivalent ones. Note that Theorem 11 expresses an important property that we shall use in many cases to make cuts in the resolution trees. Indeed, if ℓ has no model in S and $\ell \sim \ell'$ then ℓ' will have no model in S, thus we cut the branch which corresponds to the assignment of ℓ' in the resolution tree. Therefore, if there are n symmetrical literals we can cut $n-1$ branches in the resolution tree.

PROPOSITION 12. *(Necessary conditions for symmetry). If two literals (variables) ℓ and ℓ' are symmetric in a set of clauses S then the number of occurrences of the variable ℓ in S is the same as the number of occurrences of ℓ' and there must be a correspondence between the length of clauses in which ℓ occurs and the length of clauses in which ℓ' occurs; i.e.*

$$\ell \sim \ell' \Rightarrow \begin{cases} \mathit{occurrence_number}(\ell) = \mathit{occurrence_number}(\ell') \\ \\ \mathit{occurrence_number}(\neg\ell) = \mathit{occurrence_number}(\neg\ell') \\ \\ \mathit{if}\ \exists c \in S \text{ s.t. } |c| = n \text{ and } \ell \in c \Leftrightarrow \exists c' \in S \text{ s.t. } |c'| = n \text{ and } \ell' \in c' \end{cases}$$

Proof.

(1) The first condition is not satisfied: Let ℓ and ℓ' be two symmetric literals in S. Suppose that occurrence_number$(\ell) = n$ and occurrence_number$(\ell') = m$ with $(n \neq m)$. Then there are m clauses which contain ℓ' in the set S and n clauses which contain ℓ. After all permutations of ℓ and ℓ' the new set S' will have m clauses with the literal ℓ and n clauses with the literal ℓ' but $n \neq m$.

Then $S \neq S' \Leftarrow$ there does not exist a symmetry σ of S such that $\sigma(\ell) = \ell'$ or $\sigma(\ell) = \ell$. Then ℓ and ℓ' are not symmetric. This gives a contradiction with the previous hypothesis. □

The same proof can be performed if we suppose that :

$$\text{occurrence_number}(\neg\ell) \neq \text{occurrence_number}(\neg\ell').$$

(2) If the second condition is not satisfied: then $\exists c \in S$ such that $|c| = n$ and $\exists \ell \in c$ and $\nexists c' \in S$ such that $|c'| = n$ and $\ell' \in c'$. After all permutation of ℓ and ℓ' the new system S' will have a clause of length n in which ℓ' occurs. Hence, $S \neq S' \Rightarrow$ there does not exist a symmetry σ of S, such that $\sigma(\ell) = \ell'$, or $\sigma(\ell) = \ell$, then ℓ and ℓ' are not symmetric. This gives a contradiction with the previous hypothesis. □

5. Symmetry Detection Method

Let S be a set of clauses. Finding a symmetry on S is equivalent to finding a permutation σ which keeps S invariant. Many symmetries may exist, but at each resolution step we are interested in only one of them. For example, in the SLRI method [5] for a refuting literal ℓ, it is more interesting to get the symmetry which has more literals occurring in the same clause as ℓ. But in the Davis and Putnan or in semantic evaluation procedures it is better to compute the wider cycle of given literal ℓ - i.e. we try to find a set of literals $\{\ell, \ell_1, \ell_2, \ldots, \ell_n\}$ so that $\sigma(\ell) = \ell_1, \sigma(\ell_1) = \ell_2, \ldots, \sigma(\ell_n) = \ell$ with σ a symmetry defined on S.

It is known that each permutation can be expressed as a product of elementary permutations called transpositions. Thus we write all permutations, β, as a product in the form $\beta = (x_1, y_1)(x_2, y_2)\ldots(x_n, y_n)$. In order to check whether two literals x_0 and y_0 are symmetric in S we shall compute a symmetry, σ, of S, such that $\sigma(x_0) = y_0$. To that end, we try to express σ as a product of the form $\sigma = (x_0, y_0)\sigma'$, with σ' a sub-permutation of σ that we shall define as we do the other transpositions. Therefore, to substitute x_0 into y_0, we replace it by y_0 in each clause in which it appears. Thus we insert relationship clauses in which x_0 appears into the clauses in which y_0 does.

To do this, clauses and other variables will be related (taking into account the previously necessary conditions of symmetry: Proposition 12). For instance, let us consider the two following clauses: $c = x_0 \vee x_1 \vee x_2 \ldots \vee x_n$, $c' = y_0 \vee y_1 \vee y_2 \ldots \vee y_n$ and try to relate them. After replacing x_0 by y_0 we process the other variables in c. Thus each variable, x_i, $i \in \{1 \ldots n\}$, must be

linked to a variable y_j, $j \in \{1 \ldots n\}$ and at each step we add the transposition (x_i, y_j) to σ.

If it is possible to substitute all the variables, we say that c is transformed into c' with σ i.e. $\sigma(c) = c'$. If all clauses containing x_0 have been related, then we consider the next transposition and we apply the same process. When there is no more transposition, the σ found is a symmetry of S.

The other possibility is that when we try the transposition (x_i, y_j) we cannot put into the relationship a clause in which x_i occurs. Thus, if $(x_i, y_j) = (x_0, y_0)$ then the symmetry σ does not exist. Otherwise we backtrack and try to link x_i, to another variable (either in the same clause in which y_j occurs, or else we try to relate c to another clause).

Note that the symmetry between x_{i-1} and y_{j-1} depends on the one between x_i, and x_j: i.e. if $\sigma = (x_0, y_0) \ldots (x_{i-1}, y_{j-1})(x_i, y_j) \ldots (x_k, y_k)$, then $(x_0 \sim y_0) \Rightarrow (x_{i-1} \sim y_{i-1}) \Rightarrow (x_i \sim y_i) \Rightarrow (x_k \sim y_k)$. In actual implementation we fix a level of backtracking in order not to spend too much time on failure cases. Thus we take another variable to substitute x_i iff the current backtracking number is not greater than the fixed level. Otherwise we reject the symmetry, since we are interested only in the ones which do not take more time than resolution. In practice our algorithm computes all symmetry in less than twenty backtrackings; when we go over this number, generally the symmetry does not exist. In many cases we find the symmetry with no backtracking. Our method therefore bas a linear complexity aspect.

The problem is now to find a clever strategy (or a choice function) for substituting the variables in order to build σ in a satisfactory time.

STRATEGY

It is clear that the efficiency of the algorithm depends on the ordering of the variable to be permuted. We propose the following strategy:

(1) first consider the variables which can only be substituted by themself;

(2) begin the processing with the variable of which we compute the cycle of symmetry;

(3) each clause we begin to relate must be achieved as soon as possible.

In other words, variables in this clause are in priority in order to backtrack quickly in case of wrong choice.

(4) Identity is applied for the main variables, when there is no new transpositions to try.

For more details about this method and its implementation we refer the reader to ref. [3].

6. Advantage of Symmetries

The SLRI method is the algorithm of SL resolution [8], augmented by the variant of Cubbada [5]. This method is generally much more efficient than the others.

6.1. Introducing symmetries ine the SLRI method

Using the result of Theorem 4.4, we can improve the method of semantic evaluation as well as the SLRI method. Instead of refuting each literal separately, it is possible to refute symmetric literals simultaneously. Thus, let $C = \{\ell_1, \ell_2, \ldots, \ell_{i-1}, \ell_i, \ldots, \ell_n\}$ be a clause in a system of clauses S. Suppose that the literals $\ell_1, \ell_2, \ldots, \ell_{i-1}$ have been refuted and that ℓ_i is the literal we try to refute. If we prove that ℓ_i is symmetric to one of the previous literals, then it will be directly refuted by the symmetry property.

Indeed, if $\ell_i \sim \ell_j$ in S with $1 \leq j \leq i-1 \Rightarrow$ [ℓ_i has a model in $S \Leftrightarrow \ell_j$ has a model in S], as ℓ_j bas no model in $S \Rightarrow \ell_i$ has no model in S, then ℓ_i is refuted and we can cut the branch which corresponds to ℓ_i assignment in the resolution tree. For optimizing the use of symmetries, we try to get more symmetric literals in the same clause, in order to cut a large part of the resolution tree.

EXAMPLE 13. Let us consider the pigeonhole problem (Put n pigeons in $n-1$ holes such that one pigeon at most can be in a hole at each moment). For instance, let $n = 3$(3 pigeons and 2 holes), this problem is described by the following set of clauses:

$$
\begin{array}{ll}
C_1: & p_1(1) \vee p_1(2) \\
C_2: & p_2(1) \vee p_2(2) \\
C_3: & p_3(1) \vee p_3(2) \\
C_4: & \neg p_1(1) \vee \neg p_2(1) \\
C_5: & \neg p_1(1) \vee \neg p_3(1) \\
C_6: & \neg p_2(1) \vee \neg p_3(1) \\
C_7: & \neg p_1(2) \vee \neg p_2(2) \\
C_8: & \neg p_1(2) \vee \neg p_3(2) \\
C_9: & \neg p_2(2) \vee \neg p_3(2)
\end{array}
$$

The resolution tree of the SLRI method is shown in Figure 1.

The resolution tree of the SLRI method associated with symmetry is shown in Figure 2.

It is clear that the subtree built from l $p_1(1)$ in the resolution tree of SLRI (Figure 1) is symmetric with the one built from $p_1(2)$. This is due to the fact that the two literals $p_1(1)$ and $p_1(2)$ are symmetric in S. Therefore, in the second resolution tree (Figure 2) the sub-tree of $p_1(2)$ disappears, because $p_1(2)$

is refuted directly according to its symmetry with $p_1(1)$. When the number of pigeons grows, symmetries are used at different levels of resolution. Thus we cut more than one branch in the resolution tree; the complexity of the problem is linear in number of steps. Let us remark that, when the number of pigeons is greater than 8, the resolution of this problem with the SLRI method without symmetry becomes impossible. The pigeonhole problem is studied in greater detail in the next section in order to show the influence of the symmetry property on the SLRI method.

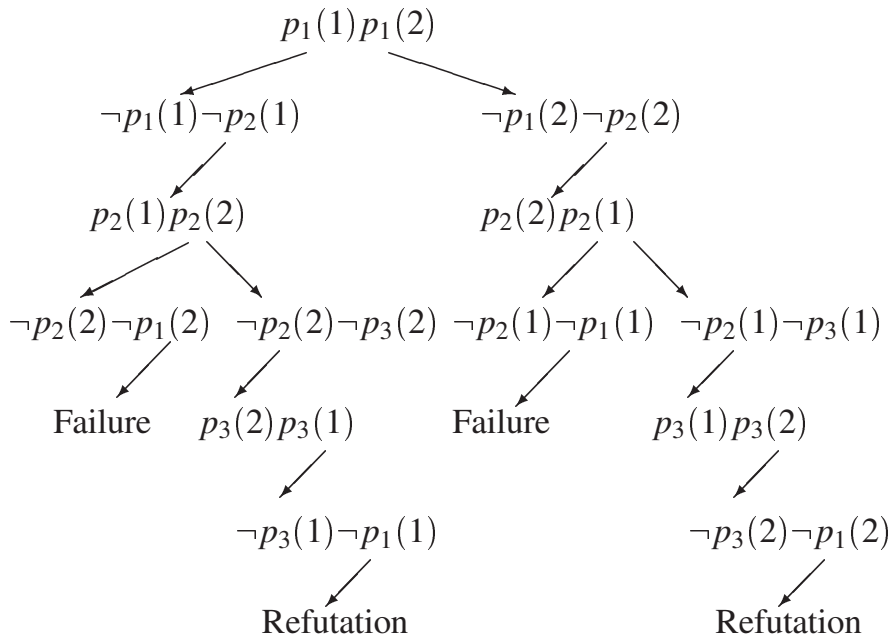

Figure 1. Resolution tree of the SLRI method. (pigeonhole problem, $n = 3$)

6.2. SYMMETRIES IN THE SEMANTIC EVALUATION METHOD

DEFINITION 14. Let S be a system of clauses and $p_1, p_2, \ldots, p_n$ distinct literals which occur in S such that the set $\{p_1, p_2, \ldots, p_n\}$ does not contain both a literal and its opposite.

$T_{p_1,p_2,\ldots,p_n}(S)$ is the set of clauses obtained from S by removing all the clauses containing one of the following literals $p_1, p_2, \ldots, p_n$ and all the occurrences of $\neg p_1, \neg xcp_2, \ldots, \neg p_n$ the other clauses.

$T''_{p_1,p_2,\ldots,p_n}(S)$ is the set $T_{p_1,p_2,\ldots,p_n}(S)$ simplified by subsumption: i.e. if $T_{p_1,p_2,\ldots,p_n}(S)$ contains both a clause c and a sub-clause of c then the clause c does not stand in $T''_{p_1,p_2,\ldots,p_n}(S)$.

The QUINE algorithm bas a semantic aspect, and recursively checks the satisfiability of a system of clauses.

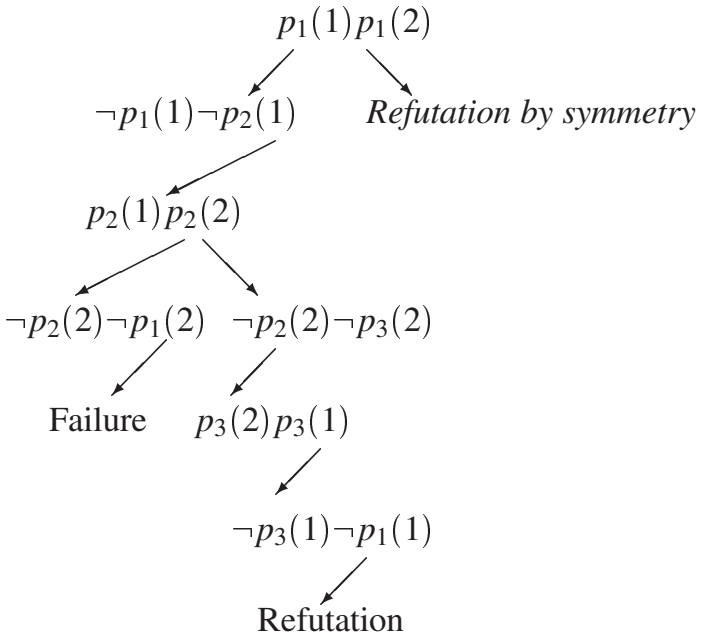

Figure 2. The $n = 3$ pigeonhole problem resolution tree, SLRI method with symmetry

ALGORITHM 1. Let S be a system of clauses.

```
Satisfaisable(S):
if S = ∅
then S is satisfiable.
else if S contains an empty clause
     then S is not satisfiable
     else begin
       choose arbitrarily one literal p witch occurs in S
       if T''_p(S) is satisfiable
       then S is satisfiable
       else if T''_¬p(S) is satisfiable
            then S is satisfiable
            else S not satisfiable
     end.
```

Many improvements have been added to the basic algorithm (QUINE). The Davis and Putnam procedure is obtained by introducing the following two rules.

The first one consists to satisfy the unit clauses in priority. The second one consists in assigning the monotone literals in priority. Indeed, Davis and Putnam uses the following property. if p is a monotone literal in S, then S is satisfiable iff $T'_p(S)$ is satisfiable.

Assigning a pure literal allows to make a cut in the proof tree. Generalizing this property leads to the theorem of model partitioning [11] which allows

us to cut more branches in the proof tree when a monotone literal has been assigned a value.

The method of semantic evaluation has its origin in the previous property. It is more efficient than the two previous algorithms.

6.3. INTRODUCING SYMMETRIES

It is also possible to introduce symmetries into the semantic evaluation method. We will take account of the same property as above (if the literals ℓ and ℓ' are symmetric in S, then [ℓ has a model in $S \leftrightarrow \ell'$ has a model in S]).

At each step of resolution we assign to a literal the value TRUE or FALSE. If it generates the empty clause, then we insert in the model which is being built the opposite of that literal, together with all the other opposites of its symmetric literals.

Indeed: if ℓ is a literal in a System S such that $T"_\ell(S) \models \square$ ($\square$ is the notation of the empty clause and $\models$ the logical implication symbol), then ℓ has no model in $S \Rightarrow \forall I$ a model of S' $I[\neg\ell] = 1$, if $\ell \sim \ell'$ has no model in S (Theorem 4.4), thus $\forall I$ a model of S' $I[\neg\ell'] = 1$. Now we can cut in the resolution tree the branch which corresponds to the assignment of ℓ' and insert $\neg\ell$ and $\neg\ell'$ in the model which is being built.

As a consequence, the system of clauses is significantly simplified, and many branches of the resolution tree are cut simultaneously, because in the backtracking case we do not take care of negation assignments of symmetric literals. Thus, for each symmetric literal, we cut a branch in the resolution tree. Then with n symmetric literals we can make $n-1$ cuts.

To implement this new algorithm we need a procedure which, for a given literal, first computes a cycle of its symmetric literals which generates the empty clause. Such a cycle is obtained by putting literals in the same clause into correspondence. If this cycle exists, then we backtrack immediately, because we know that there exists no model at the current level; otherwise we compute the larger cycle (which contains more symmetric literals) in order to quickly obtain the empty clause or a model of the system of clauses.

7. Application

7.1. SLRI APPLICATIONS

We have seen that the symmetry property improves the efficiency of the resolution of classical problems, such as the pigeonhole problem, Shur's lemma, the eight queens, etc.

The two graphs below (Figures 3 and 4) illustrate the results of the pigeonhole problem solved by the SLRI method, with and without advantage being taken of symmetries.

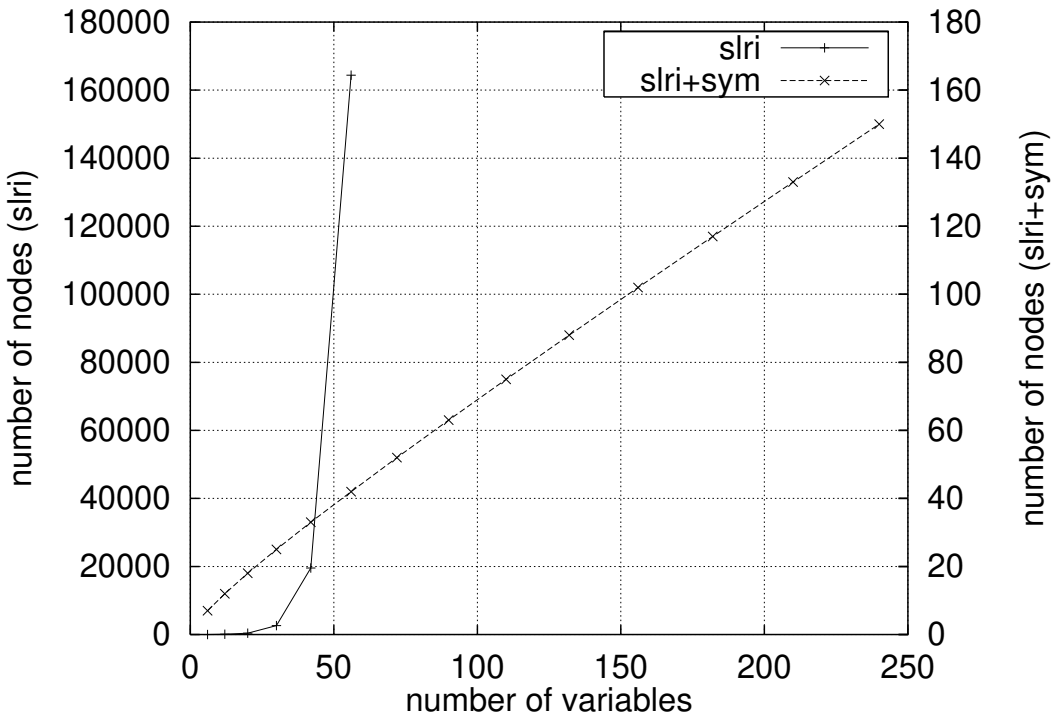

Figure 3. The pigeonhole problem: comparison of the number of nodes.

Interpretation

Figure 3 shows that the complexity of resolution of the SLRI method associated with symmetry is linear in the number of elementary steps; whereas SLRI without symmetry cannot solve the problem when the number of pigeons is greater than eight. The complexity of resolution is exponential. Figure 4 shows that the CPU time is no longer exponential, but becomes proportional to n^2.

The statement of the Ramsey problem with 17 vertices and 3 colors is: 'Use three different colors to color a complete graph with 17 vertices, such that no monochrome triangles will appear'. In the SLRI method we use symmetry on literals which occur in the same clause. Unfortunately such symmetries do not exist at different levels of resolution in the Ramsey problem. Moreover, there are symmetric literals in different clauses at all levels of resolution; therefore, the method of semantic evaluation can give better results. In that case, we need a good heuristic to decide which is the literal to assign next, in order to retain the symmetries in the next resolution levels.

7.2. Semantic evaluation applications

We now present some results on the semantic evaluation method with and without taking advantage of the symmetry property. For comparison we present in program Eval_Sem some results obtained with the semantic evaluation

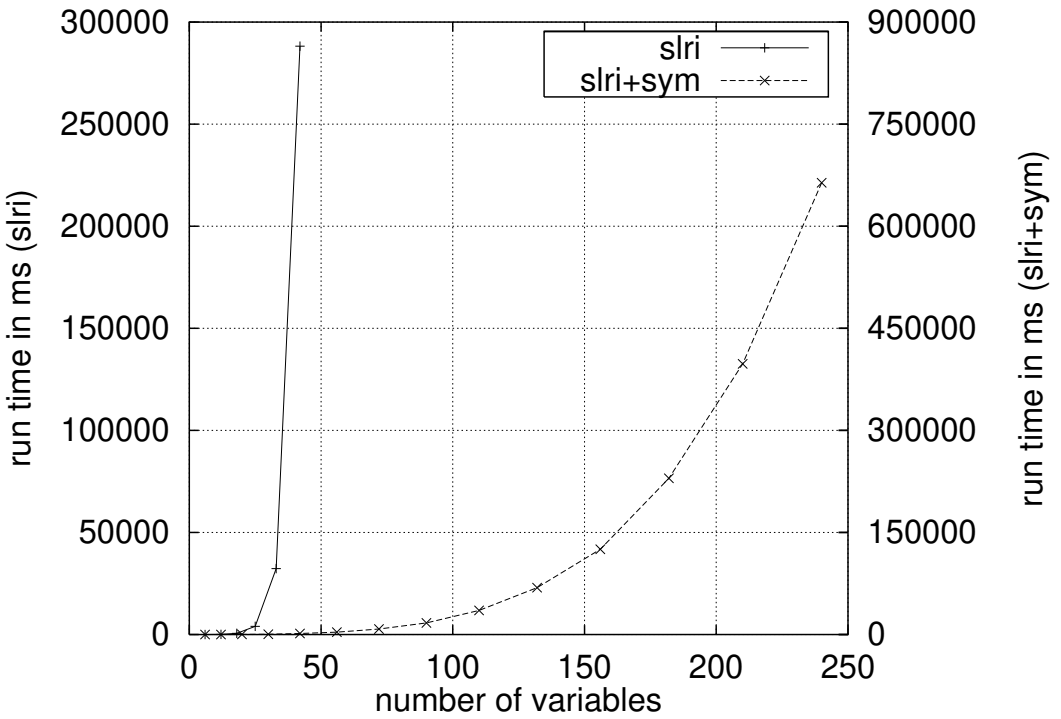

Figure 4. The pigeonhole problem: comparison of run times

method and in program Eval_Sem+Sym we give the corresponding results of the method, taking advantage of symmetries.

Table I shows some results on Schur's Lemma and on the Ramsey problems. Schur's Lemma consists of coloring the first n integers with three colors such that if i and j are colored with the same color then the integer k, such that $k = i + j$ and $k \leqslant n$, must be colored with a different one. For all integer i the color of every integer k, such that $k = 2 \times i (k \leqslant n)$ must be different. For the Ramsey problem, the edges of a complete graph on n vertices are colored with three different colors such that no monochromatic triangle appears.

The most satisfying result is that we have proved for the first time the unsatisfiability of the Ramsey problem with 17 vertices and three colors. This result is given by our algorithm implemented in Pascal, in 30 min CPU time on a SUN4/110. The semantic evaluation algorithm without the symmetry property, bas run for 15 h of CPU time and more than 1400000 steps on an HP 9000/350 without success.

Table II contains some results on the pigeonhole problem described above. In this problem there are symmetries at each level. Thus SLRI and the semantic evaluation method with symmetry detection have a linear resolution complexity. Without the symmetry property both methods fail to find the proof when the number of pigeons is greater than 10.

Table I. Schur's lemma and Ramsey's problem

Problems	Clauses	Variables	Eval_Sem		Eval_Sem+Sym	
			Steps	Times	Steps	Times
Schur13	178	39	684	3.3"	42	0.05"
Schur14	203	42	2061	11.2"	249	1.33"
Ramsey14	1456	273	273	5.7"	273	1.33"
Ramsey15	1785	315	4078	1'.30"	723	50.73"
Ramsey16	2160	360	4094	1'.29"	2957	1'.25"
Ramsey17	2584	408	–	–	27000	30'

Table II. Pigeonhole problems

Number of pigeons	Clauses	Variables	Eval_Sem+Sym	
			Steps	Times
14	1197	182	193	4.8"
16	1816	240	253	7.73"
18	2619	306	321	13.50"
20	3630	380	397	22.36"
22	4873	462	481	37.11"
24	6372	552	573	55.58"
26	8150	650	673	1'36"
28	10234	756	781	2'2"
30	12645	870	897	3'4"

8. Conclusion

Many problems cannot be solved with the classical resolution methods when they involve more than a certain number of variables. However, most of these problems include symmetries; the efficiency of the resolution methods can be significantly increased by taking advantage of this property.

To that end, we have proved several propositions, and we have implemented the corresponding results in two resolution methods. Thus we have obtained very satisfactory CPU limes for the pigeonhole problem; we have also been able to solve the Ramsey problem with 17 vertices and 3 colors, which bas always been impossible before.

We intend to use the symmetries in order to compute only the models which are not symmetric with a model which has already been found during resolution. Thus, a problem will be solved when all the basic models are

found; all the other ones can be deduced from these basic models by applying the symmetries. In particular, this method can be applied when considering how to obtain the two solutions which are not isomorphic for the Ramsey problem with 16 vertices and 3 colors [7]. Promising results are expected in the near future.

We are also working on what we call "strongly symmetrical cycles", i.e: for any order of the literals, a cycle is still a cycle. This will be useful when dealing with the statement. 'N literals are true among the literals of the clause c'.

Acknowledgement

This research was supported by the PRC-GDR Intelligence Artificielle, the BAHIA and the MRE-NTER-PRC 'Classes Polynomiales' projects. Also, our thanks to Pierre Siegel and Jean-Louis Lassez for their helpful comments.

References

1. Bauer, M., D. Brand, M. Fischer, A. Meyer, and M. Paterson: 1973, 'A note on disjonctive form tautologies'. *SIGACT News* **5**, 17–20.
2. Benhamou, B. and L. Saïs: 1990, 'Étude des symétries en calcul propositionnel'. Technical report, Mémoire de DEA, GIA-Marseille Luminy (France).
3. Benhamou, B. and L. Saïs: 1991, 'Study of symmetries in propositional calculus'. In: *Inter. Workshop on Computer Science*. Annaba, pp. 78–102.
4. Bibel, W.: 1990, 'Short proofs of the pigeonhole formulas based on the connection method'. *Journal of Automated Reasoning* **6**, 287–297.
5. Cubbada, C. and M. Mouseigne: 1988, 'Variantes de l'algorithme de SL-résolution avec retenue d'information'. Thèse de doctorat, Université d'Aix-Marseille II (GIA Luminy), Marseille (France).
6. Davis, M. and H. Putnam: 1960, 'A Computing Procedure for Quantification Theory'. *Journal of the Association for Computing Machinery* **7**, 201–215.
7. Kalbfleisch, J. and R. Stanton: 1969, 'On the maximal triangle-free edge-chromatic graphs in three colors'. *Journal Combinatorial Theory* **5**, 9–20.
8. Kowalski, R. and D. Kuehner: 1971, 'Linear Resolution with Selection Function'. *Artificial Intelligence* **2**, 227–260.
9. Krishnamurthy, B.: 1985, 'Short Proofs for Tricky Formulas'. *Acta Informatica* **22**, 253–275.
10. Lyndon, R. C.: 1964, *Notes on Logic*. Van Nostrand Mathematical Studies.
11. Oxusoff, L. and A. Rauzy: 1989, 'L'évaluation sémantique en calcul propositionnel'. Thèse de doctorat, Groupe d'Intelligence Artificielle Université d'Ais-Marseille II, Faculté de Luminy, Marseille (France).
12. Tseitin, G.: 1968, 'On the complexity of derivations in the propositional calculus'. In: H. Slesenko (ed.): *Structures in Constructives Mathematics and Mathematical Logic, Part II*. pp. 115–125.

Two proof procedures for cardinality based language in propositional calculus

Parue dans les actes de « *The eleventh annual Symposium on Theoretical Aspects of Computer Science (STACS'94)* », Caen (France), P. ENJALBERT, E.W. MAYR et K.W. WAGNER éditeurs, volume 775 de *Lecture Notes in Computer Science*, Springer Verlag, pages 71–82, février 1994.

Two proof procedures for cardinality based language in propositional calculus[1]

Belaid Benhamou, Lakhdar Sais and Pierre Siegel

L.I.U.P. Case G - Université de Provence,
3,Place Victor Hugo - F13331 Marseille cedex 3, France
phone number : 91.10.61.08
e-mail : [Benhamou, Sais, Siegel]@gyptis.univ-mrs.fr

Abstract. In this paper we use the cardinality to increase the expressiveness efficiency of propositional calculu and improve the efficiency of resolution methods. Hence to express propositional problems and logical constraints we introduce the pair formulas $(\rho, \mathcal{L})$ which mean that "at least ρ literals among those of a list $\mathcal{L}$ are true". This makes a generalization of propositional clauses which express only "One literal is true among those of the clause". We propose a cardinality resolution proof system for which we prove both completeness and decidability. A linear proof for Pigeon-Hole problem is given in this system showing the advantage of cadinality. On other hand we provid an enumerative method (DPC) which is Davis and Putnam procedure adapted with Cadinality. Good results are obtained on many known problems such as Pigeon-hole problem, Quueenes and some other instances derived from mathematical theorems (Ramsey, Schur's lemma) when this method is augmented with the principle of symmetry.

Key words: theorem proving, propositional calculus, symmetry and cardinality.

1 Introduction

Determining an appropriate language for the representation of knowledge, requires a good compromise between expressiveness and efficiency of the resolution methods. With increased interest in theorem proving procedures, a collection of proof systems and constraint logic programming languages CLP [4, 7, 11] have been proposed to handle a variety of logical theories. The cardinality operator [8] is a new logical connective for constraint logic programming. It provides CLP users with a new way of combining (primitive) constraints to build non-primitive ones. It also implements the principle "infer simple constraints from difficult ones" which is actually the basic principle behind the design of CLP over Finite Domains, and would be appropriate

[1] This work is supported by the PRC-GDR Intelligence Artificielle, the project BAHIA and the MRE-INTER-PRC project *CLASSES POLYNOMIALES*

to build non-primitive constraints as well.

In this paper we introduce the pair formulas which are cardinality formulas, to express logic problems. In Section 2 we introduce the necessary terminology and notations. A proof system is given in Section 3 followed by a method of derivation in Section 4 for which both completeness and decidability are proved. In section 5, we apply this to obtain a linear proof for the Pigeon-Hole problem. In Sections 6 we describe the method DPC which is Davis and Putnam procedure adapted with cardinality. The property of symmetry (section 7) [2] is combined with the method DPC to provide short proofs for some problems showing the efficiency of our method.Finally we give computation times for the method DPC with the advantage of symmetry on some known benchmarks.

2 Definition and notations

We shall assume some familiarities with the propositional calculus. A propositional variable is called a variable, and we distinguish it from a literal, which is a variable together with a parity-positive or negative. A formula means a well formed propositional formula using one of the binary connectives. The connectives of primary interest are: AND ($\wedge$), OR ($\vee$), NOT ($\neg$). The constants TRUE and FALSE will be represented by 1 and 0, respectively.

2.1 Syntax

A pair formula is a statement $(\rho, \mathcal{L})$ in which ρ is a positive integer and $\mathcal{L}$ a list of literals with eventual repetitions. The pair formula $(\rho, \mathcal{L})$ expresses the constraint "At least ρ literals among those of the list $\mathcal{L}$ are true". In the following we use pair instead of pair formula. [2]

Exemple 2.1

- $(1, abc)$ is logically equivalent with the clause $a \vee b \vee c$
- $(2, abc)$ is logically equivalent to the set of clauses $\{a \vee b, a \vee c, b \vee c\}$
- $(2, aabc)$ is logically equivalent to the set of clauses $\{a \vee a \vee b, a \vee a \vee c, a \vee b \vee c\}$ which is equivalent to $\{a \vee b, a \vee c, a \vee b \vee c\}$

It is interesting to note at this point that pairs are quite expressive. A conjunction of literals $\psi_1 \wedge \psi_2 \ldots \wedge \psi_n$ can be expressed by $(n, \psi_1\psi_2 \ldots \psi_n)$, a disjunction $\psi_1 \vee \psi_2 \ldots \vee \psi_n$ by $(1, \psi_1\psi_2 \ldots \psi_n)$ and a negation $\neg\psi$ by $(1, \neg\psi)$. In the sequel we prove

[2] Two additional cardinality formulas can be considered : "At most ρ literals among those of $\mathcal{L}$ are true", encoded by $(\rho, +, \mathcal{L})$ and " Exactly ρ literals among those of $\mathcal{L}$ are true", (notation is $(\rho, e, \mathcal{L})$). These are interesting formulas, but in practice they can be expressed, using pairs. Indeed, if $\mid \mathcal{L} \mid$ denotes the number of literals in $\mathcal{L}$ and $\neg\mathcal{L}$ the list of the negation literals of those of $\mathcal{L}$ then $(\rho, +, \mathcal{L}) \Leftrightarrow (\mid \mathcal{L} \mid -\rho, \neg\mathcal{L})$ and $(\rho, e, \mathcal{L}) \Leftrightarrow (\rho, \mathcal{L}) \wedge (\mid \mathcal{L} \mid -\rho, \neg\mathcal{L})$ are tautologies.

that a pair $(\rho, \mathcal{L})$ represents $C_{|\mathcal{L}|}^{|\mathcal{L}|-\rho+1}$ of propositional clauses. For example, a pair $(2, abc)$ is logically equivalent to the set $\{a \vee b, a \vee c, b \vee c\}$.

2.2 Semantics

A classical truth assignment I satisfies a pair $(\rho, \mathcal{L})$ if and only if at least ρ literals among those of $\mathcal{L}$ are assigned to the value true in I. It is obvious that a pair $(\rho, \mathcal{L})$ such that $\rho >\mid \mathcal{L} \mid$ is unsatisfiable (contradictory). On the other hand, each pair $(0, \mathcal{L})$ is a tautology. The assignment I satisfies a set S of pairs if it satisfies all of its pairs. Henceforth we will assume that the formulas are given in pair formula representation. We observe that in the case of sets containing only pairs of kind $(1, \mathcal{L})$ the definition of a truth assignment is the same as in propositional calculus.

Exemple 2.2 The truth assignment $\{a, \neg b, \neg c\}$ satisfies the paires $(1, abc)$ and $(2, ab\neg c)$, but did not satisfy the pair $(2, abc)$.

3 Proof system

Let S be a set of pairs. We give two basic inference rules to define a derivation method. We also prove both completeness and decidability for this method.

3.1 Resolution rule

Let $(\alpha, \ell_1\ell_2 \ldots \ell_n.\mathcal{L})$ and $(\beta, \neg\ell_1\neg\ell_2 \ldots \neg\ell_n.\acute{\mathcal{L}})$ be two pairs of the set S such that $n \geq 1$, then we can deduce the pair $(\alpha + \beta - n, \mathcal{L}.\acute{\mathcal{L}})$. This is,

$$\frac{(\alpha, \ell_1\ell_2 \ldots \ell_n.\mathcal{L}),\ (\beta, \neg\ell_1\neg\ell_2 \ldots \neg\ell_n.\acute{\mathcal{L}})}{(\alpha + \beta - n, \mathcal{L}.\acute{\mathcal{L}})}$$

Exemple 3.1 If $S = \{(2, abc), (1, \neg a\neg b)\}$, then a single application of the previous rule is enough to obtain the pair $(3 + 1 - 2, c) = (1, c)$.

However the previous rule is not sufficient to prove the unsatisfiability of the set $S = \{(2, aay), (2, \neg a\neg ax)\}$. To get a complete system, we need to introduce the following merging rule as it is donne in classical resolution.

3.2 Merging rule

If $c = (\alpha, \ell^k\mathcal{L})$[3] is a consistent pair of S such that $\mid \ell^k\mathcal{L} \mid= \theta$ and k a positive integer, then for every sublist φ of $\mathcal{L}$ such that $\mid \varphi \mid= max\{0, \theta - \alpha + 1 - k\}$ we

[3] ℓ^k means k times the literal ℓ

infer the pair $(1, \ell\varphi)$ and the pair $(max\{0, \alpha - k\}, \mathcal{L})$. This is :

$$\frac{(\alpha, \ell^k \mathcal{L}) \forall \varphi \subseteq \mathcal{L} :\mid \varphi \mid = max\{0, \theta - \alpha + 1 - k\}}{(1, \ell\varphi), (max\{0, \alpha - k\}, \mathcal{L})}$$

Note that when $k \geq \theta - \alpha + 1$ the pair $(1, \ell)$ is deduced. This means that when the number of redundancy k of the literal ℓ is greater than or equal to $\theta - \alpha + 1$ we prove the literal ℓ. The number of pairs $(1, \ell\varphi)$ to be generated becomes smaller when $\theta - \alpha + 1$ gets close to zero. In general the number of pairs we generate is not important.

The proof of the previous example $S = \{(2, aay), (2, \neg a \neg ax)\}$ becomes now obvious. Indeed the pairs $(1, a)$ and $(1, \neg a)$ are infered by the merging rule. Applying the resolution rule on them we deduce the unsatisfiable pair $(1, \emptyset)$.
In the case of a pair $(1, \ell^k \mathcal{L})$, the rule has the same behavior as the merging rule in propositional calculus.

Proposition 3.2 *Both previous rules are sound.*

Proof Due to the semantics defined before, the soundness of the previous rules becomes evident.

3.3 Subsumption rules

To make the system decidable and more efficient, we have to add subsumption rules.

1. Let $(\alpha, \mathcal{L})$ and $(\beta, \acute{\mathcal{L}})$ be two satisfiable pairs of S such that $\beta \leq \alpha$ and $\mathcal{L} \subseteq \acute{\mathcal{L}}$, then the pair $(\alpha, \mathcal{L})$ subsumes the pair formula $(\beta, \acute{\mathcal{L}})$ in S.

   Example. $(3, a\ b\ c\ d)$ subsumes $(2, a\ b\ c\ d\ e)$

2. Let $(\alpha, \mathcal{L})$ and $(\beta, \acute{\mathcal{L}})$ be two satisfiable pairs of S such that $\beta \leq \alpha$, $\acute{\mathcal{L}} \subseteq \mathcal{L}$ and $\beta \leq \alpha - (\mid \mathcal{L} \mid - \mid \acute{\mathcal{L}} \mid)$, then the pair formula $(\alpha, \mathcal{L})$ subsumes the pair $(\beta, \acute{\mathcal{L}})$ in S.

   Example. $(2, a\ b\ c\ d)$ subsumes $(1, a\ b\ c)$

4 Method of derivation

In the following we shall give some analogs of the classical resolution properties. A derived pair (derivation) is :

- Either the pair r obtained by applying the resolution rule on pairs c_1 and c_2, which contains at least two opposite literals.
  If $c_1 = (\alpha, \ell_1 \ell_2 \ldots \ell_n . \mathcal{L})$ and $c_2 = (\beta, \neg\ell_1 \neg\ell_2 \ldots \neg\ell_n . \acute{\mathcal{L}})$, then
  $r = (\alpha + \beta - n, \mathcal{L}.\acute{\mathcal{L}})$
- Or the pair r obtained by applying the merging rule on a pair $c = (\alpha, \ell^k \mathcal{L})$, then $r = (1, \ell\varphi)$ with $\mid \varphi \mid = max\{0, \theta - \alpha + 1 - k\}$ or $r = (\alpha - k, \mathcal{L})$

The soundness of the proof system is straightforward, for each of the inference rules soundness can be eazily checked. Thus, if cr is a pair derived from a set of pairs S, then $S \models cr$.

Proposition 4.1 *If S is a set of pairs and Cr a set of pairs derived from S, then $S \equiv^4 S \cup Cr$.*

Proof This is a direct consequence of the Proposition 3.2.

Definition 4.2 *A method of derivation defined on a sytem S of pairs is a map R which associates with each subset C of S a subset $R(C)$ such that:*

1. $C \subset R(C)$
2. *Each pair in $R(C)$ which does not appear in C is logically equivalent to a derived pair of C.*

By vertue of proposition 4.1 we get,

Proposition 4.3 $R(C) \equiv C$

If $R(C)$ contains the unsatisfiable pair, then C is unsatisfiable. Let C be a subset of S, we denote $R^0(C) = C, R^{n+1}(C) = R^n(R(C))$, and $R^*(C) = \cup_{n \in N} R^n(C)$. By construction of $R^*(C)$ we see that if c is a pair of C, then c belongs to $R^*(C)$ iff there exists $n \in N$ such that $c \in R^n(C)$.
Applying Proposition 4.3 we obtain the following property.

Proposition 4.4 $C \equiv R^*(C)$

It is obvious that if $R^*(C)$ includes the unsatisfiable pair, then C will be contradictory. A derivation method R defined on a set S of pairs is complete iff for all contradictory subset C of S, the unsatisfiable pair is in $R^*(C)$. This means that there exists an integer n such that the unsatisfiable pair belongs to $R^n(C)$. We say that the method R is decidable if it is complete and for each subset C of S there exists an integer n such that $R^n(C) = R^{n+1}(C)$. Thus $R^n(C) = R^*(C)$.

[4] $\equiv$ denote the semantic equivalence

Definition 4.5 *A set of pairs S is saturated if all of its derived pairs that are not tautologies are subsumed by other pairs of S.*

Definition 4.6 *If $\mathcal{L}$ is a list of literals such that $\mid \mathcal{L} \mid = n$, and m an integer such that $m \leq n$ then $\mathcal{C}_n^m\{\mathcal{L}\}$ denotes the set of propositional clauses obtained by considering all the disjunctions formed by the combinations of m literals among those of the list $\mathcal{L}$.*

Proposition 4.7 *Let $c = (\alpha, \mathcal{L})$ be a pair such that $\mid \mathcal{L} \mid = \theta$ then we have the following logical equivalence :*
$c \equiv \mathcal{C}_\theta^{\theta-\alpha+1}\{\mathcal{L}\}$

Proof Let I be a model of c. Thus at least α literals among those of $\mathcal{L}$ are satisfied by I. In other words, I satisfies at most $\theta - \alpha$ literals among the negation literals of $\mathcal{L}$. This implies that I satisfies each clause of the set $\mathcal{C}_\theta^{\theta-\alpha+1}\{\mathcal{L}\}$. This is so, because each clause of the set $\mathcal{C}_\theta^{\theta-\alpha+1}\{\mathcal{L}\}$ contains exactly $\theta - \alpha + 1$ literals. To prove the converse, we show that if $(\alpha, \mathcal{L})$ has no model then $\mathcal{C}_\theta^{\theta-\alpha+1}\{\mathcal{L}\}$ has no model too. Suppose that $(\alpha, \mathcal{L})$ has no model, then any truth assignment I, satisfies less than α literals among those of $\mathcal{L}$ (at most $\alpha - 1$); That is, I satisfies more than $\theta - \alpha$ literals among the negation literals of $\mathcal{L}$ (at least $\theta - \alpha + 1$). It is not hard to see that the clause of $\mathcal{C}_\theta^{\theta-\alpha+1}\{\mathcal{L}\}$ which is formed with the negations of the $\theta - \alpha + 1$ literals with the value true in I, is not satisfied by I. We conlude that I is not a model of $\mathcal{C}_\theta^{\theta-\alpha+1}\{\mathcal{L}\}$.

As a consequence of proposition 4.7, we obtain the following lemma.

Lemma 4.8 *If $S = \cup_{i=1,k}\{(\alpha_i, \mathcal{L}_i)/ \mid \mathcal{L}_i \mid = \theta_i\}$ is a set of k pairs, and S_p a set of propositional clauses, such that $S_p = \cup_{i=1,k}\mathcal{C}_{\theta_i}^{\theta_i-\alpha_i+1}\{\mathcal{L}_i\}$, then $S \equiv S_p$.*

The lemma is a direct consequence of the proposition 4.6. It is intersting to see that for each set S of pairs there exists an equivalent set S_p of propositional clauses and vice versa. It is obvious that in general the number of clauses in S_p is greater than the number of pairs in S.

Lemma 4.9 *Let S be a set of pairs, and S_p the corresponding set of propositional clauses, then S is saturated iff S_p is saturated.*

The proof is given in [3]. Now we can conclude :

Theorem 4.10 (completeness) A saturated set S of pairs is contradictory iff it contains the unsatisfiable pair.

Note that if R is a derivation method defined on a set S of pairs, such that for all subset C of S, $R^*(C)$ is saturated, then R is complete. The proof of the theorem

is based on the two previous lemmas. Indeed completeness property is a known result for saturated propositional clauses systems. On the other hand we deduce from lemma 4.7 and lemma 4.8 that all saturated set S of pairs can be transformed into an equivalent system S_p of propositional clauses which is also saturated. It is obvious that the property of completeness is verified for the set S.
Many improvement can be done for this method as it has been done in resolution.

5 A proof for Pigeon-Hole problem

The problem consists in putting n pigeons in $n-1$ holes such that each hole holds at most one pigeon. We show here how cardinality leads to find a linear proof for Pigeon-Hole problem. This problem is described with the following set of pairs.

1 $(1, p_{1(1)}p_{1(2)} \dots p_{1(n-1)})$
2 $(1, p_{2(1)}p_{2(2)} \dots p_{2(n-1)})$
$\vdots$
n $(1, p_{n(1)}p_{n(2)} \dots p_{n(n-1)})$

n+1 $(n-1, \neg p_{1(1)} \neg p_{2(1)} \dots \neg p_{n(1)})$
n+2 $(n-1, \neg p_{1(2)} \neg p_{2(2)} \dots \neg p_{n(2)})$
$\vdots$
2n-1 $(n-1, \neg p_{1(n-1)} \neg p_{2(n-1)} \dots p_{n(n-1)})$

Note that $p_i(j)$ means that the the pigeon-hole j holds the pigeon i. The problem is expressed with only $2n-1$ pairs, however the same problem requires $n+n(n-1)^2/2$ propositional clauses.
One way to prove the unsatisfiability of the problem is to use a linear resolution method. It consist to apply at each step the resolution rule on the current resolvant and another pair of the remaining pair formulas. The proof is given by the following steps:

Proof By application of the resolution rule on the formulas (1) and $(n+1)$ we deduce the pair $(n-1, p_{1(2)} \dots p_{1(n-1)} \neg p_{2(1)} \dots \neg p_{n(1)})$. We continue the resolution on the $n-2$ first positive literals of the previous pair. Thus by $n-2$ applications of the resolution rule we get the pair:
$((n-1)+(n-2)(n-1)-(n-2), \neg p_{2(1)} \dots \neg p_{n(1)} \neg p_{2(2)} \dots \neg p_{n(2)} \dots \neg p_{2(n-1)} \dots \neg p_{n(n-1)})$
in which only negative literals appear. It is obvious now that with $n-1$ applications of the resolution rule on the literals $\neg p_{2(1)} \neg p_{2(2)} \dots \neg p_{2(n-1)}$ we obtain the pair $((n-1)+(n-2)(n-1)-(n-2)+(n-1)(n-2), \emptyset)$ which is identical to the unsatisfiable pair $(1, \emptyset)$.Therefore the unsatisfiability of the problem is shown using

only $2n-2$ applications of the resolution rule. Thus we find a linear proof for the pigeon-hole problem which is not possible in classical resolution.

6 The method DPC

Lemma 4.7 permits the translation of resolution tools from propositional logic to pair formula representation. On the other hand, by few elementary transformations we obtain the corresponding method of Davis and Putnam in the pair formulas representation. We describe it below and give some results and applications for some interesting problems.
The Davis and Putnam procedure[6] is based on the semantical approach which deal with the interpretation of the formulas. We give below the corresponding procedure on pair formulas.

Given a set S of pair formulas and a literal p of S, we will denote by $S_{p\leftarrow 1}(S_{p\leftarrow 0})$ the set obtained by the following two rules :

1. - Striking out all the pairs of cardinality one, that contain $p(\neg p)$ respectively.
   - Deleting all occurences of $p(\neg p)$ and decreasing the cardinality by one from the remaining pairs.
2. deleting all occurences of $\neg p(p)$ from the remaining pair formulas without decreasing the cardinality.

Proposition 6.1 *Let S be a set of pair formulas and let p be a literal of S, then S is satisfiable if and only if $S_{p\leftarrow 1}$ or $S_{p\leftarrow 0}$ is satisfiable.*

The Davis and Putnam is based on the previous proposition. To define the procedure we use two other rules:

1. *Unit pair rule.* If the set S of pairs contains a unit pair, *i.e.* a pair $(n, \mathcal{L})$, such that $\mid \mathcal{L} \mid = n$, then assign all the literals in $\mathcal{L}$ the value TRUE.
2. *Pure literal rule.* If there exists a literal p in S such that $\neg p$ does not appear in S, then strick it in all pairs and decrease the cardinality. Thus we obtain the system $S_{p\leftarrow 1}$ which is satisfiable if and only if S is satisfiable.

We present in Figure 1 the Davis and Putnam procedure with cardinality (DPC) using the two previous rules :

DPC Procedure

Given a set S of pairs defined over a set of variables V.

- If S is empty, return "satisfiable".
- If S contains an empty pair, return "unsatisfiable".
- (Unit pair rule) If S contains a unit pair c, assign all the variables mentioned the value TRUE which satisfies c, and return the result of calling DPC on the simplified set of pairs.
- (Pure literal rule) If there exists a pure literal in S, assign it a value TRUE, and return the result of calling DPC on the simplified set of pairs.
- (Splitting rule) Select from V a variable v which has not been assigned a truth value. Assign it a value, and call DPC on the simplified set of pairs. If this call returns "satisfiable", then return "satisfiable". Otherwise, set v to the opposite value, and return the result of calling DPC on the re-simplified set of pairs.

Figure 1: The Davis and Putnam procedure with cardinality

7 Symmetries in cardinality formulas

The principle of symmetry is originaly introduced by Krishnamurty [13], he showed that it is a powerful augmentation to resolution. On the other hand symmetries have been studied in [2] and have been used to increase the efficiency of both Sl-resolution [12] and Semantic Evaluation [14] methods.

First of all, let us define the concepts of permutations and symmetry, and prove significant properties that will enable us to improve the proof algorithms, for more details we refer the reader to [2, 3].
If V is a set of propositional variables then a bijection map $\sigma : V \to V$ is called a permutation of variables. If S is a set of pairs, c a pair of S and σ a permutation of variables occuring in S, then $\sigma(c)$ is the pair obtained by applying σ to each variable of c and $\sigma(S) = \{\sigma(c)/c \in S\}$

Definition 7.1 *A set P of literals is called complete if $\forall \ell \in P$; then $\neg\ell \in P$*

Definition 7.2 *Let P be a complete set of literals and S a set of pairs of which all literals are in P. Then a permutation σ defined on P ($\sigma : P \to P$) is called a symmetry of S if it satisfies the following conditions:*

1. $\forall \ell \in P, \sigma(\neg \ell) = \neg \sigma(\ell)$
2. $\sigma(S) = S$

Two literals (variables)ℓ and $\acute{\ell}$ are symmetrical in S (notation $\ell \sim \acute{\ell}$) if there exists a symmetry σ of S such that $\sigma(\ell) = \acute{\ell}$. A set $\{\ell_1, \ell_2, \ldots, \ell_n\}$ of literals is called a cycle of symmetry in S if there exists a symmetry σ defined on S, such that $\sigma(\ell_1) = \ell_2, \ldots \sigma(\ell_{n-1}) = \ell_n, \sigma(\ell_n) = \ell_1$.

Remark 7.3 All the literals in a cycle of symmetry are symmetrical two by two. The symmetry detection method is not discribed here, it is very closed to the one given in [2].

Theorem 7.4 *Let ℓ and ℓ' be two literals of S. If $\ell \sim \ell'$ in S, then ℓ has a model in S iff ℓ' has a model in S .*

This theorem expresses an important property that we use in [2] to make prune the proof tree. Indeed if ℓ has no model in S and $\ell \sim \ell'$, then ℓ' will have no model in S, thus we prune the branch which corresponds to the assignment of ℓ' in the resolution tree. Therefore, if there are n symmetrical literals we can cut $n-1$ branches in the proof tree.

The DPC method is increased with the previous property and applied for the following problems.

7.1 Description of the benchmarks

- Queens. Placing N queens in $N \times N$ chessboard such that there is no couple of queens attacking each other.

- Erdös's theorem. Find the permutation σ of N first numbers such that for each 4-tuple $1 \leq i < j < k < l \leq$ N none of the two relations $\sigma(i) < \sigma(j) < \sigma(k) < \sigma(l)$ and $\sigma(l) < \sigma(k) < \sigma(j) < \sigma(i)$ is verified.
  This problem is modeled by creating for each couple (i,j) a variable $f_{i,j}$ which means $\sigma(i) < \sigma(j)$. The rules express the associativity of the relation $<$, and prohibit the misplaced 4-tuples.
  For $N \leq 9$ the problem admits solutions, beyond it doesn't.

- Schur's lemma: How to distribute N counters numbred from 1 to N into 3 boxes A, B, C in accordance with the following rules:
  1) A box can't contain both the counters numbered i and $2 * i$
  2) A box can't contain the counters numbered i, j and $i + j$
  This problem is modeled simply by creating one variable by counter and by box.For $N \leq 13$ the problem admits solutions, beyond it doesn't.

- Ramsey problem's: Color the edges of a complete graph on N vertices with three different colors such that no monochromatic triangle appears.
  For $N \leq 16$ the problem admits solutions, beyond it doesn't.

Below we present in Tables 1 and 2 some results for different problems obtained by DPC with the advantage of symmetry. The most satisfying results is that we

Problems	Size formula		DPC+Sym	
	pairs	Variables	Steps	Times
Queens 8	50	64	47	0.06"
Queens 10	64	100	85	2.04"
Erdös 9	420	36	35	0.36"
Erdös 10	660	45	914	8.21"
Schur13	152	39	42	0.05"
Schur14	175	42	249	1.33"
Ramsey14	1274	273	273	2.82"
Ramsey15	1575	315	723	50.73"
Ramsey16	1920	360	2957	1'.25"
Ramsey17	2312	408	27000	30'

Table 1: Schur's Lemma, Ramsey's problem, Erdös and Queens

Number of pigeons	Size formula		DPC+Sym	
	Pairs	Variables	Steps	Times
10	19	90	61	0.183"
15	29	210	131	0.950"
20	39	380	226	3.350"
25	49	600	346	9.950"
30	59	870	491	25.133"
35	69	1190	661	57.333"
40	79	1560	673	1'57"
45	89	1980	781	3'35"
50	99	2450	897	4'57"

Table 2: Pigeon hole problems

have proved for the first time the unsatisfiability of Ramsey problem $R(3,3,3)$ with 17 vertices. This result is given by our algorithm, implemented in Pascal, 30 min CPU on a SUN4/110. The Semantic Evaluation algorithm has run 15 h CPU and

more than 1400000 steps on a HP 9000/350 without success. Note that the complexity of pigeon-hole problem, in number of steps becomes linear.

8 Related works

- Hooker [9] has established the connection between resolution method for logic and cutting plane methods for integer programming[5, 10]. In his work he gives a generalized resolution procedure for clauses that asserts that at least a certain number of their literals are true.

- The cardinality operator [8] implements the principle "infer simple constraints from difficult ones" which is actually the basic principle behind the design of CLP over Finite Domains, and would be appropriate to build non-primitive constraints as well
- Four types of constrains are defined in [1] to express naturally propositional problems and a generalization for model partition theorem is given.

9 Conclusion

In this paper, we have introduced the pair formulas to express problems in a more natural way and inprove the efficiency of resolution methods in propositional calculus. The pair formulas have a simple semantic and let to get a generalization of the conjunctive normal form (CNF). A proof system which is at once complete and decidable is given. To reason about the statement "at least ρ literals are true among the literals of a list $\mathcal{L}$", the detection of symmetries[2] seem to be less expensive in this formalism. Symmetries are applied to Davis and Putnam procedure with cadinality and satisfactory CPU times are obtained on different problems. We intend to generalize Sl-resolution method to pair formulas, some properties of cardinality seems to be particularly useful when dealing with this method.

References

[1] A. Aguirre. *How to use symmetries in boolean constraints solving.* PhD thesis, GIA- Luminy, Marseille, 1992.

[2] B. Benhamou and L. Sais. Theoretical study of symmetries in propositional calculus and application. *Eleventh International Conference on Automated Deduction, Saratoga Springs,NY, USA*, 1992.

[3] B. Benhamou and L. Sais. Theoritical study of symmetries in propositional calculus. Technical Report 1, Universit de Provence, 1992.

[4] A. Colmerauer. An introduction to prolog iii. *CACM*, 4(28):412–418, 1990.

[5] C. Cook and G. Turan. on the complexity of cutting-planes proofs. In *working paper, cornell university, ithaca, ny.* 1985.

[6] M. Davis and H. Putnam. A computing procedure for quantification theory. *JACM*, 7:201–215, 1960.

[7] M. Dincbas, P. hentenryck, H. Simonis, A. Aggoun, T. Grof, and F. Berthier. The constraint logic programming language chip. In *International Conference on Fifth Generation Computer Systems*, Tokyo, japan, 1988.

[8] P. Hentenryck and Y. Deville. The cardinality operator: A new logical connective for constraint logic programming. Technical report, CS Departement, Brown University, October 1990.

[9] J. Hooker. Generalised resolution and cutting planes. In *Approches to Intelligent Decision Support, a volume in Annals of Operations Researchs series.*

[10] J. Hooker. A quantitative approach to logical inference. *Decision Support Systems*, 4:45–69, 1988.

[11] J. Jaffar and J. Lassez. Constraint logic programming. In *POPL-87*, Munich, FRG, January 1988.

[12] R. Kowalski and D. Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2:227–260, 1971.

[13] B. Krishnamurty. Short proofs for tricky formulas. *Acta informatica*, (22):253–275, 1985.

[14] L. Oxusoff and A. Rauzy. *L'valutaion smantique en calcul propositionnel.* PhD thesis, GIA-Luminy, Marseille, 1989.

Calcul propositionnel : Vers une extension du formalisme

Parue dans les actes des « *Cinquièmes Journées Nationales sur la Résolution Pratique de Problèmes NP-Complets (JNPC'99)* », Lyon (France), juin 1999, pages 31 à 39.

Calcul propositionnel : Vers une extension du formalisme

A. Rauzy
LaBRI – CNRS – Université Bordeaux I
351, cours de la Libération
33405 Talence Cedex – FRANCE
rauzy@labri.u-bordeaux.fr

L. Saïs et L. Brisoux
CRIL – Université d'Artois
rue de l'Université – S.P. 16
62307 Lens Cedex – FRANCE
{sais,brisoux}@cril.univ-artois.fr

Résumé

Les nombreux travaux autour du problème SAT qui ont été effectués ces dernières années ont permis de se faire une idée assez précise des classes d'algorithmes à mettre en œuvre pour résoudre des requêtes exprimées en calcul propositionnel. La plupart de ces algorithmes demandent un codage de l'ensemble de clauses étudié par une matrice creuse. Dans cet article, nous étudions dans quelle mesure le formalisme d'expression des problèmes peut être étendu sans changer significativement ni les algorithmes de résolution, ni surtout les structures de données. L'extension que nous proposons est double : des formules à deux niveaux aux circuits à un nombre arbitraire de niveaux, des clauses aux fonctions de seuils. Nous montrons que tous les algorithmes classiques (filtrages, algorithmes énumératifs, méthodes stochastiques et diagramme binaire de décision) s'appliquent quasiment à l'identique sur cette extension.

1 Introduction

Depuis quelques années, des efforts importants ont été faits pour mettre au point un corpus d'algorithmes efficaces pour résoudre des requêtes posées sur des formules du calcul propositionnel. Bien entendu, le terme efficace doit être compris relativement à la difficulté intrinsèque des questions posées. Tester la satisfiabilité d'un ensemble de clauses est et reste un problème difficile. Cependant, on commence à avoir une idée assez précise du type de méthodes à mettre en œuvre pour réduire, autant que faire se peut, cette difficulté. On peut diviser ces algorithmes en quatre catégories (par ordre croissant de complexité[1]) :

–Les filtrages. Il s'agit principalement de la propagation des littéraux purs et unitaires en temps linéaire [34, 16, 42]. Ils sont destinés à résoudre des problèmes dans P : 2SAT, Horn-SAT, Horn-renomabilité, ... [17, 18, 24, 25].

–Les algorithmes de recherche stochastique, au premier rang desquels GSAT [44] et ses multiples variantes . Ils sont destinés à chercher une solution au problème de satisfiabilité. Ils traitent donc un problème dans NP.

–La procédure de Davis et Putnam [19] et ses variantes [4, 20] qui trouvent une solution au problème de satisfiabilité quand il y en a une et montrent qu'il n'y en a pas dans le cas contraire. Ces algorithmes résolvent donc un problème dans coNP[2].

–Les diagrammes binaires de décision [7, 6]. Ils sont utilisés pour le calcul d'impliquants premiers [14, 40], la minimisation à deux niveaux [12], le calcul des états accessibles d'un automate d'états fini [13, 8], l'évaluation de la fiabilité [40]. Ces problèmes sont en général dans PSPACE ou #P.

Les améliorations apportées à/par ces algorithmes sont dues principalement à la mise au point de structures de données efficaces. C'est clair pour les filtrages et pour les diagrammes binaires de décision. Qui a programmé une procédure de Davis et Putnam ou une méthode de réparation locale sait que l'efficacité pratique de ce type d'algorithmes est pour beaucoup une question de codage intelligent des clauses. Il n'est pas dans notre propos de minimiser l'importance des heuristiques — l'autre grand sujet de discussion — mais le lecteur conviendra que la pertinence de celles-ci est très dépendante du type de formules étudiées[3]. De plus, si l'on s'intéresse non plus à des heuristiques individuelles mais à des classes d'heuristiques on retombe, mutatis mutandis, sur une discussion sur les structures de données permettant de les mettre en œuvre.

À l'exception des diagrammes binaires de décision, les algorithmes cités plus haut travaillent tous sur des ensembles de clauses (formules CNF ou DNF). Quelques articles ont proposé des extensions vers des formules non clausales (par exemple [47, 16, 11]), mais cette voie n'a pas été systématiquement explorée. Or tous les problèmes que l'on peut modéliser en calcul propositionnel ne se codent pas naturellement par des formules CNF ; loin s'en faut. Il existe bien sûr des algorithmes efficaces permettant de passer d'une formule générale à une formule CNF équivalente pour la satisfiabilité

[1] Le lecteur pourra se reporter à [23] ou [38] pour une définition exacte des classes de complexité évoquées ci-après.

[2] Ils sont peuvent aussi être utilisés pour produire des impliquants et des impliqués premiers [36, 10, 43].

[3] C'est ce que montrent en particulier les travaux sur les variations des modèles de générations aléatoires [41].

[39, 49], mais d'une part cela se fait au détriment de la structure de la formule (qui peut contenir des informations intéressantes que les résolveurs peuvent exploiter, e.g. des symétries), d'autre part cette équivalence ne tient que pour la satisfiabilité.

De plus, certains problèmes se codent mal avec des ensembles de clauses. Par exemple, la satisfiabilité des formules de Tseitin [46] peut être prouvée efficacement si ces dernières sont représentées par une conjonction de chaînes d'equivalences (Biconditional Normal Form) alors que leur résolution est exponentielle dans le cas d'un codage par une CNF [21]. Il faut aussi noter que lors de la modélisation d'un problème réel, de nombreuses informations ne sont que des fonctions booléennes exprimant des dépendances entre variables. Cette notion de « dépendance » [31] explicite dans le cas du formalisme que nous proposons, est difficile à obtenir dans le cas CNF[29]. La reconnaissance et l'utilisation des dépendances entre variables peut amener une réduction sensible du temps de recherche. En effet, l'affectation d'une variable indépendante détermine les valeurs des variables dépendantes, ce qui réduit le nombre d'interprétations possibles à 2^n où n est le nombre de variables indépendantes.

Dans cet article, nous étudions comment étendre au maximum le formalisme utilisé tout en :

- Restant dans le cadre du calcul propositionnel qui nous semble adapté à la résolution de nombreux problèmes pratiques.
- Gardant les mêmes structures de données, ou en tout cas les mêmes principes de mise en œuvre des algorithmes cités plus haut.
- Gardant la même complexité algorithmique sur les formules étendues que sur les formules CNF et surtout la même efficacité pratique.

Plus précisément, nous montrons que cela est possible sur des circuits combinatoires à plusieurs niveaux et construits avec des fonctions de seuils.

Le reste de cet article est organisé comme suit : la prochaine section décrit formellement les circuits combinatoires considérés. Dans les sections suivantes, nous étudions l'extension des algorithmes de chacune des quatre catégories au nouveau formalisme (respectivement dans les sections 3 pour les filtrages, 4 pour les algorithmes énumératifs, 5 pour les méthodes de recherche locale et 6 pour les diagrammes binaires de décision). Finalement, nous discutons les perspectives dans la section 7.

2 L'extension proposée

Le formalisme que nous étudions dans cet article étend les ensembles de clauses dans deux directions :

–Des formules à deux niveaux (conjonction de disjonctions) aux formules à un nombre arbitraire de niveaux.

–Des clauses aux fonctions de seuils.

Syntaxe Le langage propositionnel $\mathcal{L}$ que nous considérons est construit sur un ensemble infini dénombrable de variables propositionnelles $\mathcal{V}$, l'ensemble des entiers naturels $\mathbb{N}$, et les symboles auxiliaires "=", "#", "(", ")", "$\in$", "," "[" et "]". $\mathcal{L}$ utilise deux constructions syntaxiques : les portes et les circuits.

L'ensemble des portes de $\mathcal{L}$ est le plus petit ensemble tel que : si s, e_1, ... , e_k sont des variables de $\mathcal{V}$, α_1, ... , α_k, β_{min} et β_{max} sont des entiers positifs ou nuls, alors

$$s \;=\; \#(\alpha_1.e_1, \ldots, \alpha_k.e_k) \in [\beta_{min}, \beta_{max}] \qquad (1)$$

est une porte. On note $Var(P)$, l'ensemble des variables apparaissant dans la porte P. s est la sortie (le fanout) de la porte, les e_i sont ses entrées (fanins). On étend ces notations aux ensembles de portes : soit C un ensemble de portes, l'ensemble des variables apparaissant dans C est noté $Var(C)$, le sous-ensemble des variables n'apparaissant pas en membre droit d'une porte est noté $Sorties(C)$ et le sous-ensemble des variables n'apparaissant pas en membre gauche d'une porte est noté $Entrées(C)$. Notons que, dans un ensemble de portes certaines variables ne sont ni des entrées, ni des sorties.

L'ensemble des circuits de $\mathcal{L}$ est le plus petit ensemble d'ensembles de portes tel que :

–L'ensemble vide est un circuit (noté $\top$).

–Si C_1 et C_2 sont des circuits et

$Var(C_1) \cap Var(C_2) = \emptyset$, alors $C_1 \cup C_2$ est un circuit.

–Si C est un circuit et P est une porte dont la sortie s n'appartient pas à $Var(C)$, alors $C \cup \{P\}$ est un circuit.

On remarque que cette définition assure que, dans un circuit, toute variable est la sortie d'au plus une porte. Un circuit C peut être vu comme un (hyper)graphe orienté acyclique dont les sommets sont les variables et dont les (hyper)arcs relient les entrées de chaque porte avec sa sortie. Dans le cadre de cet article, nous nous restreindrons au cas où ce graphe n'a qu'une racine, c'est-à-dire où le circuit n'a qu'une sortie. De plus, nous dirons indifféremment équation ou porte et formule ou circuit. Les seules "vraies" variables d'un circuit sont ses entrées. Les autres sont des variables intermédiaires qui peuvent être vues comme des noms pour les sous-formules (ou la formule elle-même).

Il est important d'étendre la définition des circuits donnée ci-dessus de façon à permettre la substitution de valeurs aux variables. Soient C un circuit, x_1, ... , x_k des variables et v_1, ... , v_k des valeurs booléennes (0 ou 1). On note $C_{x_1 \leftarrow v_1, \ldots, x_k \leftarrow v_k}$ le circuit C dans lequel les valeurs v_1, ... ,v_k sont substituées respectivement aux occurrences des variables x_1, ... ,x_k.

Sémantique Une affectation σ est une application de $\mathcal{V}$ dans $\{0,1\}$. On étend par induction structurelle, les affectations en des applications de $\mathcal{L}$ dans $\{0,1\}$ de la façon suivante :

Soient σ une affectation et P une porte de la forme (1) alors $\sigma(P) = 1$ si et seulement si l'égalité $\sigma(s) = 1$ et la double inégalité

$\beta_{min} \leq \sum_{i=1,..,k} \alpha_i.\sigma(e_i) \leq \beta_{max}$ sont vérifiées ou infirmées simultanément (si x est une constante $\sigma(x) = x$).

Soient σ une affectation et C un circuit alors $\sigma(C) = 1$ si et seulement si σ donne la valeur 1 à toutes les portes de C.

Un circuit est satisfiable s'il existe une affectation le satisfaisant, c'est une tautologie si toute affectation le satisfait et c'est une antilogie si aucune affectation le satisfait.

Autres notions classiques Les notions de littéral, de produit (ensemble fini de littéraux interprétés conjonctivement), de clauses (ensemble fini de littéraux interprétés disjonctivement), d'impliquants premiers et d'impliqués premiers ont leur signification usuelle. Il y a une correspondance bi-univoque entre les substitutions et les produits : à la substitution $x_1 \leftarrow 1, \ldots, x_m \leftarrow 1, y_1 \leftarrow 0, \ldots, y_n \leftarrow 0$ correspond le produit $\{x_1, \ldots, x_m, \neg y_1, \ldots, \neg y_n\}$ et réciproquement. Les produits et les substitutions peuvent aussi être vus comme des affectations partielles.

Puissance d'expression Il est clair que $\mathcal{L}$ est une extension des formules CNF. En particulier, si a et b sont des variables, les formules $(a \vee b)$, $(a \wedge b)$ et $\neg a$ seront codées par les équations suivantes :

$$a.ou.b = \#(a,b) \in [1,2] \quad (2)$$
$$a.et.b = \#(a,b) \in [2,2] \quad (3)$$
$$non.a = \#(a) \in [0,0] \quad (4)$$

De la même façon, $\mathcal{L}$ est une extension des systèmes d'inéquations linéaires en variables 0/1. Soit l'inéquation linéaire E suivante :

$$\alpha_1.x_1 + \ldots + \alpha_m.x_m - \beta_1.y_1 - \ldots - \beta_n.y_n \leq \gamma$$

Pour la coder dans $\mathcal{L}$, on commence par coder les littéraux négatifs :

$$\Sigma_{i=1,m}\alpha_i.x_i + \Sigma_{j=1,n}\beta_j.(1-y_j) \leq \gamma + \Sigma_{i=1,n}\beta_i$$

Puis, on écrit directement les portes correspondantes :

$$\begin{aligned} E &= \#(\alpha_1.x_1, \ldots, \alpha_m.x_m, \beta_1.z_1, \ldots, \\ &\qquad \ldots \beta_n.z_n) \in [0, \gamma + \Sigma_{i=1,n}\beta_i] \\ z_1 &= \#(y_1) \in [0,0] \\ &\vdots \\ z_n &= \#(y_n) \in [0,0] \end{aligned}$$

Notons qu'en pratique la négation peut être codée directement (sans introduire de nouvelles variables), en utilisant des littéraux plutôt que des variables lors du codage d'une porte (1).

Les liens entre la programmation linéaire en $\{0,1\}$ et le calcul propositionnel ont déjà été étudiés, en particulier par Hooker (voir par exemple [27, 28]). Cependant, Hooker a surtout examiné comment interpréter les algorithmes de programmation linéaire dans le cadre de la logique booléenne. Nous proposons au contraire d'adapter les méthodes développées pour le calcul propositionnel à la programmation linéaire en $\{0,1\}$ (de plus nous considérons des formules à un nombre arbitraire de niveaux, et non des systèmes d'inéquations).

Signalons tout de suite quelques avantages de l'extension proposée sur les formules CNF.

–Elle est beaucoup plus concise. Par exemple, le fait que k formules parmi n doivent être vraies se code en une seule équation.

–Elle permet facilement l'introduction de lemmes, c'est à dire de variables codant des formules. Les lemmes permettent d'abord de construire des preuves courtes pour des formules intraitables autrement [46, 9], mais surtout de compiler facilement des descriptions de plus haut niveau (par exemple des diagrammes de fiabilité [37]).

–Elle permet de tirer parti facilement des symétries ou de toutes autres régularités apparaissant dans les problèmes, en les détectant automatiquement comme dans [4] ou en écrivant directement les formules de façon à en éliminer les redondances[4].

–Elle permet de traiter au même plan une formule et sa négation, ce qui est intéressant dans le cadre de la compilation compositionnelle de formalismes de haut niveau.

–Elle permet de mettre en œuvre des procédures de réécriture des formules dans l'esprit du vérificateur de modèle INSTEP [48]. INSTEP implémente deux types de règles : les filtrages (ou simplifications immédiates) des formules — nous reviendrons sur ce sujet dans la prochaine section — et la décomposition de Shannon qui consiste à remplacer une formule f par la formule $(x \wedge f_{x \leftarrow 1}) \vee (\neg x \wedge f_{x \leftarrow 0})$. Ce type de réécriture peut être utilisé en tant que tel pour résoudre le problème posé, ou comme pré-traitement des formules. Cette dernière voie semble particulièrement prometteuse pour les algorithmes utilisant les diagrammes binaires de décision.

Il reste bien entendu à montrer que cette extension ne se fait pas au prix d'une complexité accrue des algorithmes. C'est ce que nous allons faire dans les pro-

[4] Par exemple en imposant, dans un problème de Ramsey comme ceux traités dans [36], des inéquations sur le nombre d'arêtes de chaque couleur.

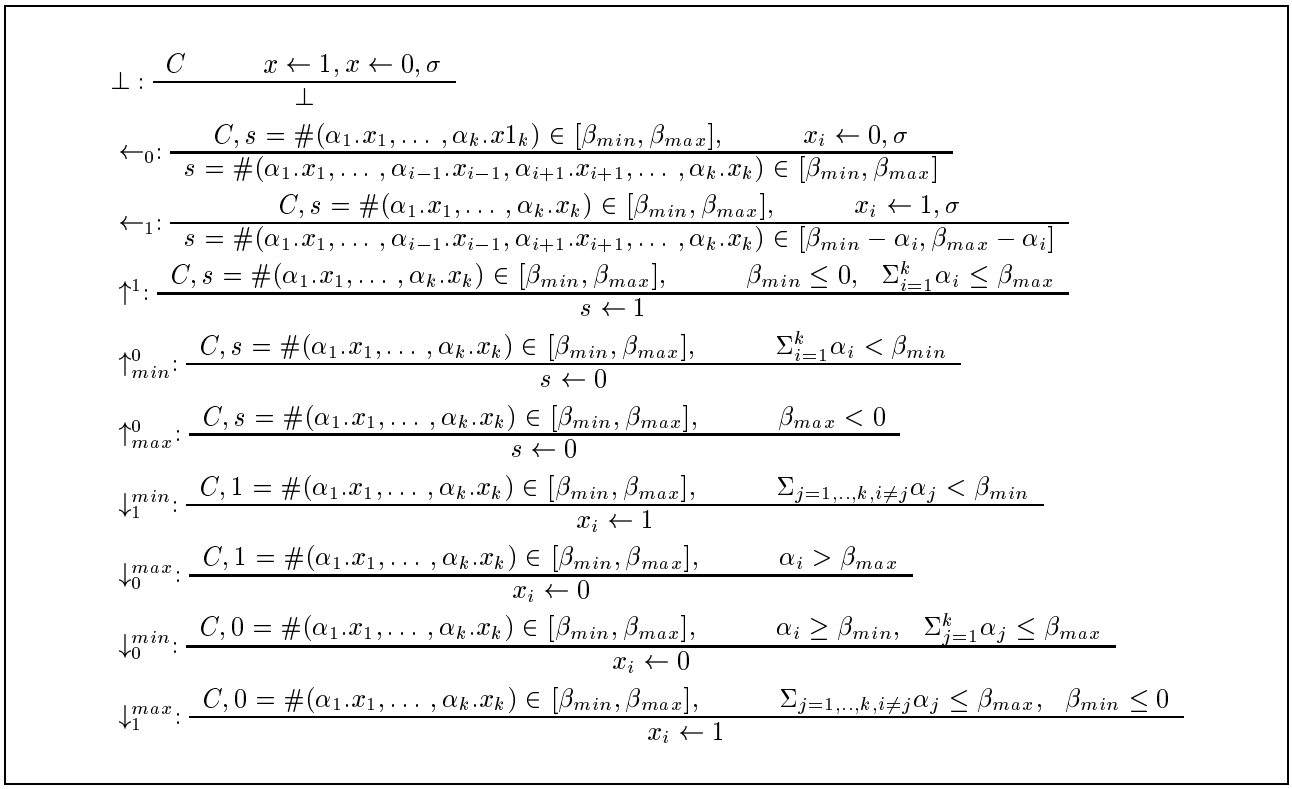

FIG. 1: Les règles définissant l'opérateur de conséquence immédiate $\vdash$.

chaines sections.

3 Filtrages

La première chose à vérifier est que le formalisme proposé peut être codé dans des structures de données permettant de propager efficacement les affectations de valeurs aux variables et de détecter les variables astreintes à prendre une valeur unique. Ces deux opérations sont la clé de voûte de toute procédure énumérative efficace, et par là, de la mise au point d'algorithmes optimums sur les classes polynômiales principales du problème SAT et des problèmes de satisfaction de contraintes [24]. La première propage des valeurs des entrées vers la sortie du circuit, la seconde procède en sens contraire. La structure de données doit donc permettre ce double mouvement, de haut en bas et de bas en haut.

Opérateur de conséquence immédiate Nous allons dans un premier temps définir un opérateur de conséquence immédiate $\vdash$ (qui décrit formellement les opérations mentionnées ci-dessus), puis nous proposerons des structures de données pour le mettre en œuvre efficacement.

La figure 1 présente $\vdash$ à l'aide de séquents. La signification de chacune de ces règles est suffisamment explicite pour que nous ne la décrivions pas en détails ici. Les séquents de $\vdash$ transforment un couple (circuit, substitution) en un autre couple (circuit, substitution) équivalent :

$$C_\sigma \vdash C'_{\sigma'} \quad \Rightarrow \quad C_\sigma \equiv C'_{\sigma'} \tag{5}$$

Nous noterons $\vdash^*$ l'opération consistant à appliquer un nombre quelconque de fois l'une des règles décrites figure 1. Il est bien clair que l'opérateur $\vdash$ admet un point fixe unique (quel que soit l'ordre d'application des règles). Nous noterons $\downarrow^\vdash C_\sigma$ ce point fixe.

Structure de données La structure de données que nous proposons (pour mettre en œuvre $\vdash$) est la suivante. On construit :

- La liste des portes du circuit étudié.
- La liste des entrées de chaque porte.
- Pour chaque variable, la liste des portes dont elle est une entrée.

Les deux dernières listes sont en fait des listes de pointeurs vers des monômes $\alpha_i.x_i$ qui comportent chacun un pointeur vers la variable x_i et un pointeur vers la porte dans laquelle ils apparaissent. La structure codant une variable doit aussi contenir un pointeur vers la porte dont elle est éventuellement la sortie. Cette structure est une simple généralisation des matrices creuses utilisées pour coder les ensembles de clauses [42].

Afin de coder l'affectation partielle courante, on maintient la valeur de chaque variable (et de chaque porte).

valeur de s	condition	affectation	caractère
$s = 1$	$\gamma_{min} + \alpha_i > \beta_{max}$	$x_i \leftarrow 0$	imposée
$s = 1$	$\gamma_{max} - \alpha_i < \beta_{min}$	$x_i \leftarrow 1$	imposée
$s = 0$	$(\gamma_{min} + \alpha_i \geq \beta_{min}) \wedge (\gamma_{max} \leq \beta_{max})$	$x_i \leftarrow 0$	imposée
$s = 0$	$(\gamma_{max} - \alpha_i \leq \beta_{max}) \wedge (\gamma_{min} \geq \beta_{min})$	$x_i \leftarrow 1$	imposée
$s = 1$	$\gamma_{max} \leq \beta_{max}$	$x_i \leftarrow 1$	libre
$s = 1$	$\gamma_{min} \geq \beta_{min}$	$x_i \leftarrow 0$	libre
$s = 0$	$\gamma_{max} \leq \beta_{max}$	$x_i \leftarrow 0$	libre
$s = 0$	$\gamma_{min} \geq \beta_{min}$	$x_i \leftarrow 1$	libre

TAB. 1: Table récapitulative des affectations imposées et libres

On maintient aussi, pour chaque porte de la forme (1), la valeur minimum γ_{min} et la valeur maximum γ_{max} que peut prendre la somme des $\alpha_i.x_i$ dans l'affectation partielle courante. Ces valeurs correspondent respectivement à l'affectation à 0 et à 1 des variables non encore affectées par σ. Quand une entrée reçoit une valeur, γ_{min} et γ_{max} sont mis à jour (en temps constant dans la mesure où il suffit d'ajouter ou de soustraire un α aux valeurs précédentes des γ). Si $\gamma_{min} > \beta_{max}$ ou si $\gamma_{max} < \beta_{min}$, la sortie s prend la valeur 0 et la propage. Si $\gamma_{min} \geq \beta_{min}$ et $\gamma_{max} \leq \beta_{max}$, la sortie s prend la valeur 1 et la propage.

Les affectations de valeurs aux variables se font donc sans modifier la structure de données. On utilise une pile pour mémoriser les affectations à effectuer.

Dans le cas où c'est la variable s qui est affectée, γ_{min} et γ_{max} sont utilisés pour effectuer les tests correspondants aux séquents $\downarrow_1^{min}$, $\downarrow_0^{min}$, $\downarrow_1^{max}$ et $\downarrow_0^{max}$ de la figure 1. On note que si la valeur de x_i est fixée par propagation alors toute variable x_j telle que $\alpha_j \geq \alpha_i$ est astreinte à prendre la même valeur. L'idée est donc de maintenir une liste des x_i triée par ordre décroissant des α_i ainsi qu'un pointeur vers la première variable non affectée de la liste. Ce tri initial de la liste se fait en $\mathcal{O}(k.log(k))$ (c'est un pré-traitement).

Le calcul du point fixe de l'opérateur de conséquence immédiate revient à étendre la substitution courante. Il se fait en temps linéaire amorti [45].

Lemme 1 (Complexité de la propagation) *Soient C un circuit et σ une substitution codés dans les structures de données décrites ci-dessus. Le calcul de $\downarrow^{\vdash} C_\sigma$ est en $\mathcal{O}(|C|)$, où $|C|$ dénote le nombre de monômes $\alpha_i.x_i$ présents dans le circuit C.*

Pour vérifier le lemme précédent, il suffit de remarquer que chaque monôme est parcouru au plus deux fois : une fois « verticalement » lors de l'affectation de la variable correspondante et une fois « horizontalement » si la variable est contrainte à prendre une valeur donnée.

Affectations libres Soient C un circuit, x une variable et v une valeur (0 ou 1). On dit que l'affectation de v à x est libre si pour toute affectation σ satisfaisant C et telle que $\sigma(x) = 1 - v$, l'affectation σ' égale partout à σ sauf en x satisfait C. Cette notion est sémantique. Il convient donc d'en donner une version affaiblie, syntaxique, qui pourra être détectée facilement sur la structure de données. La table 1 donne des conditions syntaxiques suffisantes (mais pas forcément nécessaires) pour qu'une affectation soit libre ou imposée. Le lecteur remarquera que la notion d'affectation imposée généralise celle de littéral unitaire et que la notion d'affectation libre généralise celle de littéral pur. Intuitivement, l'affectation de la valeur v à la variable x est imposée s'il existe une porte qui serait immédiatement falsifiée si la valeur $1 - v$ était affectée à x. L'affectation de la valeur v à la variable x est libre si pour toutes les portes dans lesquelles x apparaît en entrée, cette affectation ne fait que renforcer la satisfaction potentielle de la porte. On voit donc qu'il y a une dualité profonde entre affectations libres et affectations imposées.

D'autre part, si l'affectation d'un des x_i à la valeur v_i est libre alors c'est le cas pour toutes les variables non affectées de la porte. Par conséquent, dès que la condition est remplie, on peut parcourir la liste et mettre à jour des compteurs qui pour chaque variable maintiennent le nombre de portes dans lesquelles elle n'est pas libre positivement et négativement. Dès qu'un de ces deux compteurs passe à 0, la variable peut être librement affectée à l'autre valeur. Ici encore, il n'est pas difficile de voir que la gestion des compteurs se fait en temps linéaire amorti sur la taille de l'instance.

Au coût du pré-traitement près (qui est très faible), on a donc des algorithmes de propagation des variables et de détection des sous-formules astreintes à prendre une valeur unique et des sous-formules dont la valeur est libre aussi efficaces que ceux dont on dispose grâce à la structure de matrice creuse pour les ensembles de clauses [42].

Extension de la notion d'« autark » La notion d'autark (ou de partition du modèle) [22, 35, 36] se gé-

néralise de différentes façons. La plus simple est sans doute la suivante : soient C un circuit et σ une affectation partielle, σ est un autark de C si aucune des variables affectées par σ n'apparaît dans une équation non-satisfaite de C. En clair, σ sépare C en deux : des équations satisfaites et des équations dans lesquelles les variables affectées par σ n'apparaissent pas. Il est clair que si σ est un autark alors C est satisfiable si et seulement si C_σ l'est. La détection des autarks se fait suivant le même principe et avec le même coût (temps linéaire amorti) que sur les ensembles de clauses, c'est à dire en maintenant, pour chaque variable, un compteur du nombre d'équations non satisfaites dans lesquelle la variable apparaît [42].

Avec l'extension des notions de littéral unitaire, de littéral pur et d'autark, on dispose donc de toutes les briques permettant de mettre au point des algorithmes efficaces pour traiter les principales classes polynômiales du problème SAT. Cette question a été étudiée en détails dans [24, 25]. Notons que le fait de disposer d'un formalisme plus puissant devrait permettre d'étendre ces classes polynômiales.

4 Algorithmes énumératifs

On peut distinguer deux grandes classes d'algorithmes complets décidant de la satisfiabilité d'une formule booléenne :

- Ceux appliquant sous une forme ou sous une autre le principe de résolution (c'est à dire travaillant par réécriture de la formule considérée).
- Ceux qui explorent (énumèrent) de façon systématique l'espace de recherche.

Cette dernière classe s'est avérée de loin la plus efficace en pratique. Des travaux récents ont montré que les mises en œuvre les plus efficaces ne sont que des variantes de la procédure bien connue de Davis, Logeman et Loveland (DPL)[19]. Notons, que cette procédure peut être adaptée pour compiler des bases de connaissances propositionnelles (e.g. en recherchant une couverture d'impliquants (impliqués) premiers) [43, 10, 5].

La procédure de DPL Dans le cadre de cet article, nous limitons notre étude à l'extension de la procédure DPL au formalisme proposé (voir Fig. 2).

La procédure DPL se décompose en deux règles essentielles : la règle de simplification (R_1) et la règle de séparation (R_2). Son extension aux circuits nécessite une adaptation de ces deux règles. L'extension de la règle de simplification a été discutée dans la section précédente.

Les heuristiques La sélection de la prochaine variable à affecter est un facteur important dans l'efficacité de la procédure énumérative DPL. Une bonne heuristique peut réduire sensiblement la taille de l'arbre

Procédure *DPL(C)*
entrée : C un circuit (ensemble de portes)
sortie : vrai si C est satisfiable ; faux sinon
début
  R_1 *: règle de simplification*
  Filtrage et propagation
  si C est vide
    alors retourner vrai
  sinon si il existe une porte falsifiée
    alors retourner faux
  sinon début
    R_2 *: règle de séparation*
    choix de la variable x et de la valeur v à affecter
    si $DPL(C_{x\leftarrow v})$
      alors retourner vrai
    sinon retourner $DPL(C_{x\leftarrow 1-v})$
  fin
fin

FIG. 2: La procédure DPL

de recherche. De nombreux travaux ont été consacrés à l'amélioration de la règle de séparation (choix de la variable de branchement). Citons parmi elles les plus récentes : C-SAT [20], Tableau [15], POSIT, Satz [33] et Relsat[2].

Dans le cadre CNF, l'heuristique la plus utilisée est sans doute celle proposée par Jeroslow et Wang [30] qui consiste à choisir la variable apparaissant le plus souvent dans les clauses les plus courtes. Plus précisément, la variable x_i choisie est celle maximisant $w(x_i) + w(\neg x_i)$, avec $w(x_i) = \sum_{C\in S, x_i\in C} w(C)$, et $w(C) = 2^{-|c|}$. Les nombreuses variantes proposées en suite diffèrent essentiellement par leur fonction poids.

Pour adapter cette heuristique, on définit dans un premier temps la fonction $w(P)$ associée à une porte P. Il est évident que la porte définie par l'équation (1) peut s'écrire sous forme d'une conjonction de deux inéquations linéaires $ILP(6)$ et $ILN(7)$:

$$\sum_{1\leq i\leq n} \alpha_i x_i \geq \beta_{min} \tag{6}$$

et

$$\sum_{1\leq i\leq n} \alpha_i \neg x_i \geq (\gamma - \beta_{max}) \tag{7}$$

avec

$$\sum_{1\leq i\leq n} \alpha_i = \gamma$$

On remarque alors que l'équation précédente fait apparaître à la fois les occurrences positives et négatives, on retrouve de manière évidente la notion classique de littéraux purs (i.e. x_i (respectivement $\neg x_i$) est pur si

et seulement si $\forall P \in C$ tel que $x_i \in P$, $\beta_{max} = \gamma$ (respectivement $\beta_{min} = 0$)).

Ci-dessous, la description de la fonction $f(v)$:

Soit C un circuit,

$$f_w(v) = \sum_{\forall P \in C,\ v \in P} w(P) \tag{8}$$

où $w(P)$ est défini de la façon suivante[5] :

$$\text{- si } s = 1,\ w(P) = \begin{cases} w(ILP) & \text{si } v = x \\ w(ILN) & \text{si } v = \neg x \end{cases}$$

$$\text{- si } s = 0,\ w(P) = \begin{cases} w(ILN) & \text{si } v = x \\ w(ILP) & \text{si } v = \neg x \end{cases}$$

Il reste à définir le poids associé à une inéquation linéaire de la forme :

$$\sum_{1 \leq i \leq n} \alpha_i x_i \geq \beta \tag{9}$$

Il est à noter que la transformation (sans introduction de nouvelles variables) d'une inéquation linéaire en CNF peut amener à un nombre exponentiel de clauses [26]. Pour se rapprocher le plus possible des poids utilisés dans le cadre CNF pour pondérer une clause, nous avons juste besoin de connaître le nombre et la taille des clauses obtenues. Pour ce faire nous utilisons une propriété définie dans [3], montrant la transformation d'une formule de cardinalité en clauses. Nous rappelons qu'une formule de cardinalité peut être considérée comme une inéquation où tous les coefficients ont une valeur égale à un. Dans notre cas, nous considérons le coefficient associé à une variable comme une indication du nombre de répétitions de la variable. Nous obtenons donc pour l'inéquation linéaire (9) une approximation du nombre et de la longueur des clauses :

- la longueur des clauses est : $\gamma - \beta + 1$,
- le nombre de clauses est : $C_\gamma^{\gamma-\beta+1}$

avec $\gamma = \sum_{1 \leq i \leq n} \alpha_i$.

Nous proposons ci-dessous, les différentes fonctions de pondération d'une inéquation linéaire :

$$w_1 = (\alpha_i \times C_\gamma^{\gamma-\beta+1}) \times 2^{-(\gamma-\beta+1)}$$

$$w_2 = (\frac{\alpha_i}{\gamma - \beta + 1}) \times 2^{-(\gamma-\beta+1)}$$

Rappelons que dans le cas CNF, une clause C est pondérée par un poids de $2^{|C|}$. Dans les deux pondérations précédentes, $|C|$ est remplacé par $\gamma - \beta + 1$ qui est une généralisation évidente. Le facteur multiplicatif permet d'accentuer notre préférence pour les inéquations linéaires nécessitant de nombreux coefficients pour les satisfaire.

La fonction de calcul du score d'une variable x est obtenue par :

$$F_{w_i}(x) = \tag{10}$$
$$f_{w_i}(x) + f_{w_i}(\neg x) + \alpha \times \min(f_{w_i}(x), f_{w_i}(\neg x))$$

(avec $\alpha = 1.5$)

5 Recherche stochastique

Les techniques de recherche stochastique consistent à tester différentes affectations des variables. Les affectations à tester sont obtenues en modifiant de proche en proche une affectation initiale (en général en inversant la valeur d'une variable, d'où le nom de réparation locale donné aussi à ces méthodes). L'affectation initiale ainsi que les modifications sont obtenues plus ou moins pseudo-aléatoirement : on cherche à faire les meilleurs choix possibles tout en laissant un degré de liberté afin d'assurer une bonne couverture des affectations possibles. Ces méthodes incomplètes ont été utilisées avec succès pour résoudre le problème SAT [44]. Le schéma général d'une procédure de recherche stochastique est décrit Fig. 3.

Procédure *RL(C)*
entrée : C un circuit (ensemble de portes)
sortie : Un modèle, si trouvé
  pour i **allant de** 1 **à** max. essais **faire**
    choisir une affectation initiale I
    pour j **allant de** 1 **à** max. réparations **faire**
      si I satisfait C
        alors retourner vrai
      sinon
        choix de la variable x à inverser
        $I = I$ avec x inversée
      finsi
    finpour
  finpour
  retourner faux
fin

FIG. 3: Procédure générique de recherche stochastique

L'adaptation de la procédure précédente aux circuits ne présente pas de difficulté majeure. On remarque tout d'abord que les variables peuvent prendre deux valeurs (vrai ou faux), alors qu'elles peuvent en prendre trois dans les procédures énumératives (vrai, faux ou non affectée). Quoi qu'il en soit, la modification de la valeur d'une variable se fait de la même façon dans les deux types de procédure. La partie la plus délicate de l'algorithme concerne la sélection des variables candidates à l'inversion, parmi lesquelles une est choisie pseudo-aléatoirement. Cela demande

[5] Les variables du circuit apparaissant uniquement à gauche du symbole d'égalité sont affectées à 1.

d'attribuer une note à chaque variable, et de maintenir les notes le plus efficacement possible. Une extension naturelle du calcul des notes mis en œuvre dans GSAT [44] pourrait être la suivante : Soient I l'interprétation courante et x une variable de C, alors $w(x) = w_+(x) - w_-(x)$, où $w_+(x)$ (resp. $w_-(x)$) dénote le nombre de portes de C actuellement falsifiées (resp. satisfaites) par I et qui deviendraient satisfaites (resp. falsifiées), si la valeur de x était inversée. Les compteurs $w_+(x)$ et $w_-(x)$ peuvent être mis à jour efficacement en utilisant la même technique que pour la détection des littéraux unitaires (voir section 3).

6 Diagrammes Binaires de Décision

Le calcul du BDD codant une équation de la forme (1) ne pose pas de problème particulier. On calcule les BDDs E_i codant les e_i, puis on applique un algorithme énumératif (de type "forward checking") qui sépare sur la valeur des e_i et construit les BDDs en remontant. Par exemple, si l'on sépare sur la valeur de e_1 (codé par le BDD E_1) et que les BDDs construits pour $e_1 = 1$ et $e_1 = 0$ sont respectivement R_1 et R_0, le BDD codant la formule sera $(E_1 \wedge R_1) \vee (\neg E_1 \wedge R_0)$. Or, la procédure de calcul du BDD codant la fonction $(f \wedge g) \vee (\neg f \wedge h)$ à partir des BDDs codant respectivement f, g et h est précisément la procédure centrale de paquetages BDD [6].

En pratique, on a tout intérêt à mémoriser les résultats intermédiaires de façon à ne pas refaire deux fois le même calcul. Supposons par exemple que l'on veuille calculer le BDD associé à l'équation :

$$s = \#(\alpha_1.e_1, \alpha_2.e_2, \alpha_3.e_3, \alpha_4.e_4, \quad (11)$$
$$\ldots, \alpha_k.f_k) \in [\beta_{min}, \beta_{max}] \quad (12)$$

Supposons de plus que $\alpha_1 + \alpha_2 = \alpha_3 + \alpha_4$. Alors le même résultat intermédiaire sera obtenu pour les affectations $e_1 \leftarrow 1$, $e_2 \leftarrow 1$, $e_3 \leftarrow 0$, $e_4 \leftarrow 0$, et $e_1 \leftarrow 0$, $e_2 \leftarrow 0$, $e_3 \leftarrow 1$, $e_4 \leftarrow 1$. Il est donc inutile de calculer deux fois le BDD correspondant. On a là un cas typique d'utilisation des chaînes d'addition de Knuth [32].

7 Conclusion

Le calcul propositionnel est un excellent compromis entre puissance d'expression et efficacité des algorithmes de résolution associés. Cependant, la plupart des algorithmes proposés dans la littérature travaillent sur des ensembles de clauses (formule CNF). Or, tous les problèmes en calcul propositionnel ne se codent pas naturellement par des formules CNF, et le passage vers une forme CNF fait perdre des informations structurelles importantes.

Dans cet article, nous avons montré qu'en considérant des circuits combinatoires à plusieurs niveaux et construits avec des fonctions de seuil, il est possible d'étendre au maximum le formalisme sans changer (ou presque) les algorithmes, et surtout les structures de données. En considérant les classes d'algorithmes, nous avons ainsi montré que leur adaptation au formalisme proposé ne présente pas de difficulté particulière.

Ce travail est un premier pas vers la mise en place d'un formalisme booléen plus général. Il ouvre de nombreuses perspectives (dont certaines sont en cours d'étude) :

- la validation de l'approche sur des problèmes réels,
- la mise en place d'heuristiques pour prendre en compte certaines informations structurelles (e.g. les dépendances,...),
- la mise en place de techniques de compilation de formules booléennes (recherche de couverture d'impliquants (impliqués) premiers).
- l'extension des algorithmes pour résoudre des problèmes d'optimisation (ce que permettent les poids associés aux variables dans les équations).

⋮

–

Références

[1] P. Barth. A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimisation. Technical Report MPI-I-95-2-003, Max-Planck-Institute für informatik, January 1995.

[2] R.J. Bayardo Jr. and R.C. Schrag. Using CSP Look-Back Techniques to Solve Real-World SAT Instances. In *Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, pages 203–208, Providence, Rhode Island, 27-31 July 1997.

[3] B. Benhamou and L. Saïs. Two proof procedures for a cardinality based language in propositional calculus. In P. Enjalbert, E. W. Mayr, and K. W. Wagner, editors, *STACS*, pages 71–82, Caen, France, February 1994. Springer-Verlag.

[4] A. Billionnet and A. Sutter. An efficient algorithm for the 3-satisfiability problem. *Operation Researchs Letters*, 12(1) :29–36, 1992.

[5] Y. Boufkhad, É. Grégoire, P. Marquis, B. Mazure, and L. Saïs. Tractable Cover Compilations. In *Proc. of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, volume 1, pages 122–127, Nagoya, Japan, August 1997.

[6] K. Brace, R. Rudell, and R. Bryant. Efficient Implementation of a BDD Package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*. IEEE 0738, 1990.

[7] R. Bryant. Graph Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8) :677–691, August 1986.

[8] J.R. Burch, E.M. Clarke, K.L. McMillan, and D.L. Dill. Sequential Circuit Verification Using Symbolic Model Checking. In *Proceedings of the 27th ACM/IEEE Design Automation Conference, DAC'90*, pages 16–51, June 1990.

[9] S.R. Buss. Polynomial size proofs of the propositional pigeonhole principle. *Journal of Symbolic Logic*, 52 :916–927, 1987.

[10] T. Castell and M. Cayrol. Une nouvelle méthode de calcul des implicants et des impliqués premiers. In *Actes des Journées sur la Résolution Pratique de Problèmes NP-Complets, CNPC'96*, pages 153–169, Teknea, 1996.

[11] P. Codognet and D. Diaz. A Simple and Efficient Boolean Solver for Constraint Logic Programming. *Journal of Automated Reasoning*, 17 :97–128, 1996.

[12] O. Coudert. Two-level minimization : an overview. *INTEGRATION, the VLSI Journal*, 17 :97–140, October 1994.

[13] O. Coudert, C. Berthet, and J-C. Madre. Verification of Synchronous Sequential Machines Based on Symbolic Execution. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407, pages 365–373. LNCS, 1989.

[14] O. Coudert and J.-C. Madre. Implicit and Incremental Computation of Primes and Essential Primes of Boolean Functions. In *Proceedings of the 29th ACM/IEEE Design Automation Conference, DAC'92*, June 1992.

[15] J. M. Crawford and L. D. Auton. Experimental Results on the Crossover Point in Random 3-SAT. *Artificial Intelligence*, 81 :31–57, 1996.

[16] M. Dalal. Efficient Propositional Constraint Propagation. In *Proc. of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 409–414, San Jose, California, 12-16 July 1992.

[17] M. Dalal. Tractable Deduction in Knowledge Representation Systems. In *Proceedings of the 3^{th} International Conference on Principles of Knowledge Representation and Reasoning, KR'92*, pages 393–402, 1992.

[18] M. Dalal and D. W. Etherington. A hierarchy of tractable satisfiability problems. *Information Processing Letters*, 44(4) :173–180, 10 December 1992.

[19] M. Davis, G. Logemann, and D. Loveland. A Machine Program for Theorem Proving. *CACM*, 5 :394–397, 1962.

[20] O. Dubois, P. Andre, Y. Boufkhad, and J. Carlier. SAT versus UNSAT. In *Second DIMACS Implementation Challenge*, 1994.

[21] B. Dunham and H. Wang. Towards feasible solutions of the tautology problem. *Annals of Mathematical Logic*, 10 :117–154, 1976.

[22] S. Even, A. Itai, and A. Shamir. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM J. Comput.*, 5 :691–703, 1976.

[23] M.R. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Freeman, San Fransisco, 1979.

[24] R. Génisson and A. Rauzy. Aspects algorithmiques des classes polynomiales du problème sat et des problèmes de satisfaction de constraintes. In *Actes du $10^{ième}$ congrès Reconnaissance de Formes et Intelligence Artificielle, RFIA'96*, pages 97–108, AFECT-AFIA, 1996.

[25] R. Génisson and A. Rauzy. Efficient Horn renaming : yet a Davis and Putnam's procedure. In *Résolution Pratique de Problèmes NP-Complets, CNPC'96*, pages 169–184, Teknea, 1996. Also, Technical Report 1092-95, LaBRI – URA CNRS 1304 – Université Bordeaux-I, 1995.

[26] F. Granot and P.L. Hammer. On the use of boolean functions in 0-1 programming. In *Methods of Operations Research*, volume 12, pages 154–184, 1971.

[27] J.N. Hooker. A quantitative approach to logical inference. *Decision Support Systems*, 4 :45–69, 1988.

[28] J.N Hooker. Generalized resolution for 0-1 linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 6 :271–286, 1992.

[29] T. Ibaraki, A. Kogan, and K. Makino. Functional Dependencies in Horn Theory. Technical Report 98-18, DIMACS, March 1998.

[30] R. G. Jeroslow and J. Wang. Solving propositional satisfiability problems. In *Annals of Mathematics and Artificial Intelligence*, pages 167–187, 1990.

[31] H. Kautz, D. McAllester, and B. Selman. Exploiting Variable Dependency in Local Search. In *Abstracts of the Poster Sessions of IJCAI-97*, Nagoya, Japan, 1997.

[32] D.E. Knuth. *The Art of Computer Programming*, volume 2, Seminumerical Algorithms. Addison Wesley, 1969.

[33] C. Li and Anbulagan. Heuristic Based on Unit Propagation for Satisfiability Problems. In *Proc. of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 366–371, Nagoya, Japan, 1997.

[34] M. Minoux. LTUR : A Simplified Linear-Time Unit Resolution Algorithm for Horn Formulae and its Computer Implementation. *Information processing Letter*, 29 :1–12, 1988.

[35] B. Monien and E. Speckenmeyer. Solving Satisfiability in Less than 2^n Steps. *Discrete Applied Math*, 10 :287–295, 1985.

[36] L. Oxusoff and A. Rauzy. *L'Évaluation Sémantique en Calcul Propositionnel.* PhD thesis, GIA – Université de Aix-Marseille II, January 1989.

[37] A. Pagès and M. Gondrand. Fiabilité des systèmes. In *Collection de la Direction des Études et Recherches d'Électricité de France*, volume 39. Eyrolles, 1980. ISSN 0399-4198.

[38] C.H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1984.

[39] D.A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2 :293–304, 1986.

[40] A. Rauzy. New Algorithms for Fault Trees Analysis. *Reliability Engineering & System Safety*, 5(59) :203–211, 1993.

[41] A. Rauzy. On the Random Generation of 3-SAT-Instances. Technical Report 1060–95, LaBRI – URA CNRS 1304 – Université Bordeaux I, 1995.

[42] A. Rauzy. Polynomial restrictions of SAT : What can be done with an efficient implementation of the Davis and Putnam's procedure. In U. Montanari and F. Rossi, editors, *Proceedings of the International Conference on Principle of Constraint Programming, CP'95*, volume 976 of LNCS, pages 515–532, Springer Verlag, 1995.

[43] R. Schrag. Compilation for Critically Constrained Knowledge Bases. In *Proc. of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, volume 1, pages 510–515, Portland, Oregon, 4-8August 1996.

[44] B. Selman, D. Mitchell, and H. Levesque. A New Method for Solving Hard Satisfiability Problems. In *Proc. of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446, San Jose, California, 12-16 July 1992.

[45] R.E. Tarjan. Amortized Computational Complexity. *SIAM Journal on Algebraic and Discrete Methods*, 6(2) :306–318, 1985.

[46] G.S. Tseitin. On the complexity of derivation in propositional calculus. In A. O. Slisenko, editor, *Structures in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125, 1970.

[47] A. van Gelder. A Satisfiability Tester for Non-clausal Propositional Calculus. *Information and Computation*, 79 :1–21, 1988.

[48] F. Vlach. Simplification in a Satisfiability Checker for VLSI Applications. *Journal of Automated Reasoning*, 10 :115–136, 1993.

[49] J.M. Wilson. Compact normal forms in propositional logics and integer programming formulations. *Computer and Operatuons Research*, 90 :309–314, 1990.

Tabu Search for SAT

Parue dans les actes de « *The Fourteenth National Conference on Artificial Intelligence (AAAI'97)* », Providence (Rhode Island, USA), juillet 1997, pages 281 à 285.

Tabu Search for SAT *

Bertrand Mazure **Lakhdar Saïs** **Éric Grégoire**

CRIL – Université d'Artois
rue de l'Université SP 16
F-62307 Lens Cedex France
{mazure,sais,gregoire}@cril.univ-artois.fr

Abstract

In this paper, tabu search for SAT is investigated from an experimental point of view. To this end, TSAT, a basic tabu search algorithm for SAT, is introduced and compared with Selman et al. Random Walk Strategy GSAT procedure, in short RWS-GSAT. TSAT does not involve the additional stochastic process of RWS-GSAT. This should facilitate the understanding of why simple local search methods for SAT work. It is shown that the length of the tabu list plays a critical role in the performance of the algorithm. Moreover, surprising properties about the (experimental) optimal length of the tabu list are exhibited, raising interesting issues about the nature of hard random SAT problems.

Introduction

SAT, i.e. checking the satisfiability of a boolean formula in conjunctive normal form, is a canonical NP-complete problem (Cook 1971). Moreover, it is a fundamental problem in mathematical logic, automated reasoning, artificial intelligence and various computer science domains like VLSI design.

Recently, there has been a renewal of interest in understanding the nature of the difficulty of SAT (see e.g. (Chvátal & Szemerédi 1988; Dubois & Carlier 1991; Mitchell, Selman, & Levesque 1992)). At the same time, several authors have proposed new –but amazingly simple and efficient– algorithms allowing for a breakthrough in the class of computer-solvable SAT instances (see e.g. (Selman, Levesque, & Mitchell 1992; Gu 1992; Selman, Kautz, & Cohen 1993; DIMACS 1993)).

More precisely, a class of very hard SAT instances has been characterized. This class is made of random generated K-SAT instances whose probability of being satisfiable is close to 0.5 and are located at the critical point of a phase transition. These problems are most often beyond the reach of the most efficient techniques derived from conventional algorithms, like Davis and Putnam's procedure (Davis & Putnam 1960). In order to address them, two families of algorithms have been designed recently. The first one is made of logically complete techniques that aim at proving the inconsistency of SAT instances (see e.g. (Dubois *et al.* 1996; Crawford & Auton 1993)). The second one is formed of incomplete techniques based on local reparations that attempt to find a model for SAT instances. Several authors have proposed very simple local search algorithms that prove surprisingly good in solving hard large satisfiable problems (Selman, Levesque, & Mitchell 1992; Selman, Kautz, & Cohen 1993; Gent & Walsh 1993).

In this paper, tabu search for SAT is investigated from an experimental point of view. To this end, TSAT, a basic tabu search algorithm for SAT, is introduced and compared with Selman et al. Random Walk Strategy GSAT, in short RWS-GSAT (Selman, Kautz, & Cohen 1993). TSAT proves extremely competitive in the resolution of many problems, in particular hard random K-SAT instances at the critical point of the phase transition. Actually, the FTP available GSAT already contained a basic tabu option but has not been described in the literature to our best knowledge. Moreover, no investigation of the fine-tuning of its essential parameters had ever been conducted.

In the next section, the canonical K-SAT fixed-clause-length random generation model is reviewed, with emphasis on the phase transition. Then, Selman et al. RWS-GSAT algorithm is presented. TSAT is then motivated and presented. Extensive experimentations are conducted about the optimal lengths of the tabu list with respect to several criteria, leading to surprising findings. A comparison between the performance of RWS-GSAT and TSAT is then conducted. These results were first presented for a very restricted audience in 1995 (Mazure, Saïs, & Grégoire 1995). As a conclusion, some promising ideas for further research are given.

* Copyright © 1997. American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

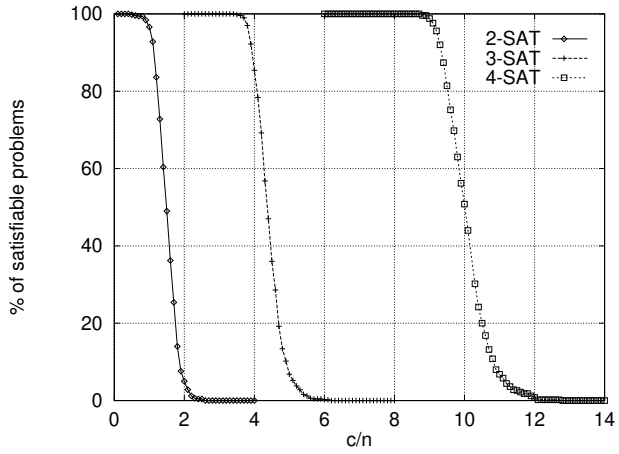

Figure 1: phase transition phenomena

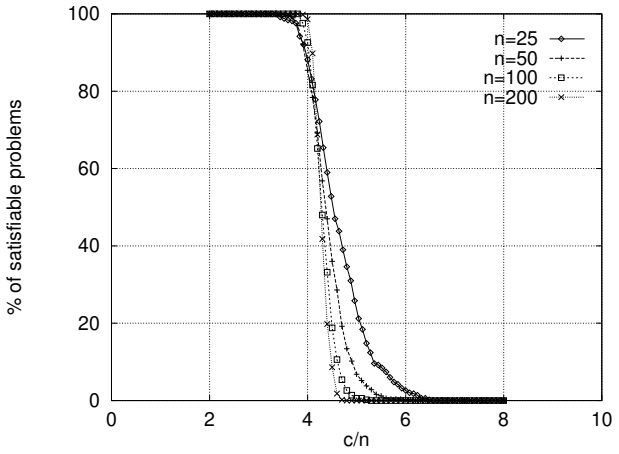

Figure 2: 3-SAT

Hard random SAT instances

SAT consists in checking the satisfiability of a boolean formula in conjunctive normal form (CNF). Let us recall here that any propositional formula can be translated into a CNF that is equivalent for SAT, thanks to a linear time algorithm (see e.g. (Siegel 1987)). A CNF formula is a set (interpreted as a conjunction) of clauses, where a clause is a disjunction of literals. A literal is a positive or a negated propositional variable. An interpretation of a boolean formula is an assignment of truth values to its variables. A model of a formula is an interpretation that satisfies the formula.

Although SAT is NP-complete, theoretical and experimental results show good average-case performance for several classes of SAT instances (see e.g. (Franco & Paull 1983)).

However, hard random instances of SAT have been observed at a phase transition. Let us illustrate this phenomenon in the usual K-SAT fixed-clause-length model (see e.g. (Chvátal & Szemerédi 1988; Cheeseman, Kanefsky, & Taylor 1991; Crawford & Auton 1993; Dubois & Carlier 1991; Mitchell, Selman, & Levesque 1992)), a random generation model where the number of literals per clause is a given value K and the sign of each literal is also randomly generated (with a 0.5 probability).

In Figures 1 and 2, we see the phase transition observed by many authors. The probability of satisfiability decreases abruptly from 1 to converge towards 0 in a phase transition as the c/n ratio increases (where c represents the number of clauses and n the number of variables). The location of the phase transition depends on the length of clauses and on the number of variables. In Figure 1, we see how the curves move to the right as this length increases. In Figure 2, we see how the curves straighten when the number of variables increases. It has been shown experimentally that instances at these phase transitions where the probability of being satisfiable is 0.5 are really hard problems. Actually, it has been proved that these problems are exponential for resolution (Chvátal & Szemerédi 1988).

Random Walk Strategy GSAT

Let us now briefly review Selman et al. GSAT algorithm (Selman, Levesque, & Mitchell 1992). This algorithm performs a greedy local search for a satisfying assignment of a set of propositional clauses. The algorithm starts with a randomly generated truth assignment. It then changes ("flips") the assignment of the variable that leads to the largest increase in the total number of satisfied clauses. Such flips are repeated until either a model is found or a preset maximum number of flips (MAX-FLIPS) is reached. This process is repeated as needed up to a maximum of MAX-TRIES times.

In the sequel, we shall consider a more recent and efficient version of GSAT, i.e. Random Walk Strategy GSAT (in short RWS-GSAT)(Selman, Kautz, & Cohen 1993). This variant of GSAT selects the variable to be flipped in the following way: it either picks with probability p a variable occurring in some unsatisfied clause or follows, with probability $1 - p$, the standard GSAT scheme, i.e. makes the best possible local move.

Clearly, this very simple algorithm is logically incomplete and belongs to the local search procedures family. However, it is surprisingly efficient in demonstrating that CNF formulas are satisfiable, in particular K-SAT instances at the phase transition.

```
Procedure  GSAT
Input:  a set of clauses S, MAX-FLIPS, MAX-TRIES
Output:  a satisfying truth assignment of S, if found
  Begin
    for i := 1 to MAX-TRIES do
       I :=  a randomly generated truth assignment
       for j := 1 to MAX-FLIPS do
          if I  satisfies S then return I
          x :=  a propositional variable such that
                a change in its truth assignment
                gives the largest increase (possibly
                negative) in the number of clauses
                 of S  that are satisfied by I
          I := I with the assignment of x reversed
       end-for
    end-for
    return  "no satisfying assignment found"
  End
```

Figure 3: GSAT algorithm: basic version

Restricting randomness

Let us stress that the Random Walk Strategy introduces an additional level of randomness in basic GSAT and thus makes an analytical study of GSAT more difficult to conduct. Another randomness property of GSAT lies in the selection of the variable to be flipped. Indeed, as Selman et al. stress it: "Another feature of GSAT is that the variable whose assignment is to be changed is chosen *at random* from those that would give an equally good improvement. Such nondeterminism makes it very unlikely that the algorithm makes the same sequence of changes over and over (Selman, Levesque, & Mitchell 1992)".

Moving further towards the goal of avoiding recurrent flips, we investigate the use of tabu search (Glover 1989; 1990) for SAT by experimenting with an algorithm of our own, called TSAT. TSAT makes a systematic use (no aspiration criteria) of a tabu list of variables in order to avoid recurrent flips and thus escape from local minima. This technique was also expected to allow for a better and more uniform coverage of the search space. Let us stress that this use of a tabu list is systematic during the search process in the sense that the tabu list is updated each time a flip is made. TSAT keeps a fixed length –chronologically-ordered FIFO– list of flipped variables and prevents any of the variables in the list from being flipped again during a given amount of time. Accordingly, the tabu list contains *variables* and, for efficiency reasons, does not keep track of forbidden interpretations, explicitly.

The efficiency of most local search procedures depends heavily on a good setting of their parameters. For instance, Selman et al. suggest specific values for GSAT parameters like MAX-TRIES, MAX-FLIPS and, very importantly, the probability p presented above (Selman, Kautz, & Cohen 1993).

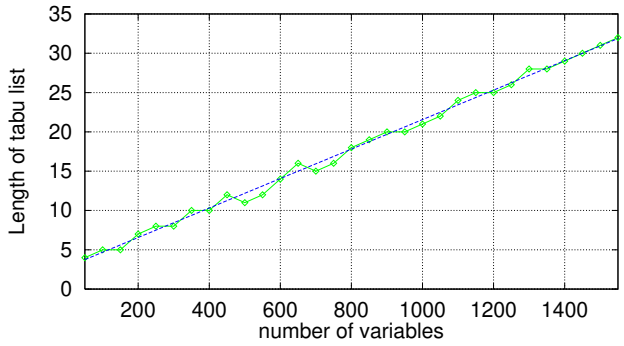

Figure 4: Optimal length of tabu lists for 3-SAT at the critical point of phase transition ($c/n = 4.25$)

In the next section, the main parameter of TSAT is fine-tuned for random K-SAT problems, namely the length of the tabu list, using the main parameters of the problem: i.e. the number of variables and the length and number of clauses.

Experimental fine-tuning of TSAT: peculiar findings

In order to find optimal lengths of the tabu list, the following extensive experimentations have been conducted. First, the 3-SAT framework has been considered. According to the standard fixed-length-clause model, 500 instances were randomly generated at the phase transition for every number of variables ranging from 50 to 1000 (by steps of 50, with their best estimated c/n ratios). The number of instances has been limited to 100 for every number of variables between 1000 and 1500 (also by steps of 50). For each such instance, TSAT has been run, varying the length of the tabu list from 1 to 50. In Figure 4, the (experimentally obtained) optimal length of the tabu list with respect to the number of variables is given. In the considered range of number of variables, this curve appears to be linear in the number of variables. Experimental result:

$$\boxed{\text{optimal length of tabu list} = 0.01875 \;\; n + 2.8125}$$

where n is the number of variables.

Moreover,

- a slight departure from the optimal length leads to a corresponding graceful degradation of the performance of TSAT. A more important distance from this optimal length leads to a dramatic performance degradation.
- these lengths remain optimal for random-generated instances outside the phase transition (the optimal

problems		Nb.	RWS-GSAT				TSAT			
n	c	inst.	time (sc.)	flips	solved	ratio	time (sc.)	flips	solved	ratio
100	430	500	.18	2803	88%	31.85	.11	1633	93%	17.60
200	860	500	1.99	18626	73%	255.85	.73	9678	74%	130.78
400	1700	500	15.03	204670	100%	2046.70	11.51	145710	100%	1457.10
600	2550	500	19.59	250464	62%	4013.85	13.92	167236	65%	2580.80
800	3400	500	140.61	1809986	67%	26854.39	99.45	1143444	71%	16150.34
1000	4250	500	369.88	4633763	57%	81009.84	292.10	3232463	62%	51802.29
2000	8240	50	3147.26	26542387	16%	1658899.19	3269.15	29415465	40%	735386.63

Table 1: TSAT vs. RWS-GSAT

size depends only on the number n of involved variables according to the above equation).

The above tests for 4-SAT (100 instances for each number of variables ranging from 50 to 600 (by steps of 50)) and similar values for the optimal lengths have been obtained.

TSAT vs. RWS-GSAT

Extensive experimental comparisons between RWS-GSAT and TSAT have been conducted for 3-SAT instances at the phase transition. Both algorithms have been implemented in a common platform written in C under Linux for Pentium PC, available from the authors. Our local implementation of Random Walk GSAT proves as efficient as Selman's one.

Both algorithms are best compared with respect to the percentage of solved problems and with respect to the number of performed flips. Let us stress that the tabu list is implemented as a circular list whose FIFO access is made in constant time. Time and number of flips in the Table 1 are average ones for solved instances. The percentage of solved problems is actually relative to the 50% (which is an approximation) of tested instances that are expected to be satisfiable since they are selected at the phase transition. The (average cumulated flips for solved instances)/(percentage of solved problems) ratio is also given in order to conduct a fair comparison when one of the algorithms solves more instances than the other one. MAX-TRIES is 5, MAX-FLIPS is n^2 and $p = 0.5$ for RWS-GSAT. These two last values are the (experimentally) best ones for GSAT with respect to random 3-SAT problems (Parkes & Walser 1996; Selman, Kautz, & Cohen 1993).

Table 1 summarizes the results and shows the good performance of TSAT. Let us stress that the given number of flips corresponds to the cumulated number of performed flips during the successive tries. However, most of the time, TSAT just required one try.

Let us note that the competitiveness of TSAT increases with the number of variables. This is illustrated

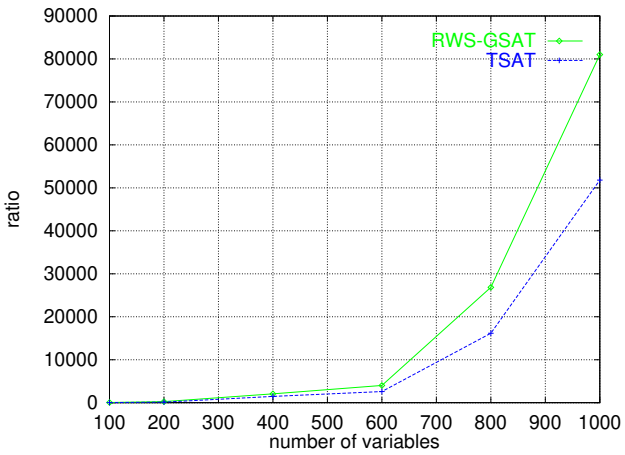

Figure 5: Results for 3-SAT instances at the phase transition obtained by RWS-GSAT and TSAT

in Figure 5). Results for 2000 variables are the most significant ones but could not have been inserted in the diagram because of the huge difference (see Table 1).

Conclusion

TSAT, a basic tabu search algorithm for SAT, has been proposed and compared with Selman et al. Random Walk Strategy GSAT procedure. TSAT makes a systematic use of a tabu list and takes out one of the randomness properties of RWS-GSAT. TSAT proves very competitive in the resolution of many problems, in particular hard random K-SAT instances. Quite surprisingly, the optimal length of the tabu lists for these random problems proves (experimentally) linear with respect to the number of variables. This linearity was quite unexpected, as well as the fact that the length does only depend on the number of variables. This finding is certainly worth further research. Intuitively, the length of the tabu list could be related to some extent to the height of local extrema. We are currently working on that, trying to relate this feature to the nature of really hard

random SAT problems. Also, as TSAT uses a basic form of tabu search, we are currently working on more sophisticated tabu strategies (Glover 1989; 1990) and extending them to other related problems (in this respect, see also the related work by (Battiti & Protasi 1996)). In particular, we are experimenting with TSAT with respect to structured examples (as those suggested in (DIMACS 1993)), using tabu lists whose lengths are dynamically fine-tuned. Another promising path of research consists in extending TSAT into a logically complete technique that keeps the power of local search for satisfiable instances. In this respect, (Mazure, Saïs, & Grégoire 1996) is a first promising step.

Acknowledgments

This work has been supported in part by the Ganymède II project of the Contrat de Plan Etat/Nord–Pas-de-Calais, and by the IUT de Lens. We thank the reviewers for their useful comments.

References

Battiti, R., and Protasi, M. 1996. Reactive Search, a history-based heuristic for MAX-SAT. In *Proceedings of the Workshop on Satifiability Problem*. Siena, Italy.

Cheeseman, P.; Kanefsky, B.; and Taylor, W. 1991. Where the Really Hard Problems are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91)*, 163–169.

Chvátal, V., and Szemerédi, E. 1988. Many Hard Examples for Resolution. *Journal of the Association for Computing Machinery* 33(4):759–768.

Cook, S. 1971. The Complexity of Theorem Proving Procedures. In *Proceedings of the third Ann. Symp. on Theory of Computing, ACM*, 151–158.

Crawford, J., and Auton, L. 1993. Experimental results on the cross-over point in satisfiability problems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93)*, 21–27.

Davis, M., and Putnam, H. 1960. A Computing Procedure for Quantification Theory. *Journal of the Association for Computing Machinery* 7:201–215.

DIMACS. 1993. *Proceedings of the Second Challenge, organized by the Center for Discrete Mathematics and Computer Science of Rutgers University.*

Dubois, O., and Carlier, J. 1991. Probabilistic Approach to the Satisfiability Problem. *Journal of Theoretical Computer Science* 81:65–75.

Dubois, O.; André, P.; Boufkhad, Y.; and Carlier, J. 1996. SAT versus UNSAT. In Johnson, D., and Trick, M., eds., *Cliques, Coloring and Satisfiability, Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Science. American Mathematical Society.

Franco, J., and Paull, M. 1983. Probabilistic analysis of the Davis and Putnam Procedure for Solving the Satisfiability Problem. *Journal of Discrete Applied Math.* 5:77–87.

Gent, I., and Walsh, T. 1993. Towards an Understanding of Hill-climbing Procedures for SAT. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93)*, 28–33.

Glover, F. 1989. Tabu search - Part I. *ORSA Journal of Computing* 1(3):190–206.

Glover, F. 1990. Tabu search - Part II. *ORSA Journal of Computing* 2(1):4–32.

Gu, J. 1992. Efficient Local Search for Very Large-Scale Satisfiability Problems. *SIGART Bulletin* 3(1):8–12.

Mazure, B.; Saïs, L.; and Grégoire, E. 1995. A New Local Search Algorithm for SAT: Performance and Analysis. In *Proceedings of the CP'95 Workshop on Studying and Solving Really Hard Problems*, 127–130.

Mazure, B.; Saïs, L.; and Grégoire, E. 1996. Detecting Logical Inconsistencies. In *Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics (AI/MATH-96)*, 116–121.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and Easy Distributions of SAT Problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, 459–465.

Parkes, A., and Walser, J. 1996. A Resampling Method to Optimize Maxflips in Local Search: Theory and Application. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, 356–361.

Selman, B.; Kautz, H.; and Cohen, B. 1993. Local Search Strategies for Satisfiability Testing. In *Proceedings of DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*, 1993.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, 440–446.

Siegel, P. 1987. Représentation et utilisation des connaissances en calcul propositionnel. Thèse d'État, Groupe d'Intelligence Artificielle, Université d'Aix-Marseille II.

Boosting complete techniques thanks to local search methods

Parue dans « *Annals of Mathematics and Artificial Intelligence* », volume 22, pages 319–331, 1998.

Boosting complete techniques thanks to local search methods

Bertrand MAZURE, Lakhdar SAÏS and Éric GRÉGOIRE

CRIL – Université d'Artois, rue de l'Université SP 16, F-62307 LENS Cédex, France
E-mail: {mazure,sais,gregoire}@cril.univ-artois.fr

In this paper, an efficient heuristic allowing one to localize inconsistent kernels in propositional knowledge-bases is described. Then, it is shown that local search techniques can boost the performance of logically complete methods for SAT. More precisely, local search techniques can be used to guide the branching strategy of logically complete Davis and Putnam's like techniques, giving rise to significant performance improvements, in particular when addressing locally inconsistent problems. Moreover, this approach appears very competitive in the context of consistent SAT instances, too.

Keywords: SAT, NP-completeness, local search methods, logical inconsistency, logical completeness

1. Introduction

A fundamental problem in logical knowledge-based representation and automated theorem proving systems lies in the difficulty of checking and handling forms of inconsistency. In particular, any local contradiction in a standard logical knowledge-based system makes it wholly inconsistent: any piece of information (and its contrary) can be inferred from it under sound and complete standard rules of deduction.

At the same time, checking in the general propositional case whether a formula is satisfiable or not -namely, the SAT problem-, can be extremely time-consuming. Indeed, although theoretical and experimental results show good average-case performance for several classes of SAT instances (see e.g. [10]), SAT is NP-complete [3].

Recently, there has been a renewal of interest in understanding the nature of the difficulty of SAT (see e.g. [2] and [9]). At the same time, several authors

have proposed new -but amazingly simple and efficient- algorithms allowing for a breakthrough in the class of computer- solvable consistent SAT instances (see e.g. [17], [18], [7] and [14]). However, since these algorithms are based on local search techniques, they are logically incomplete in that they cannot be used directly to prove in a definitive manner that a formula is inconsistent.

Actually, the most efficient complete techniques (like those derived from Davis and Putnam's one -in short DP- [6]) that are used to prove that a formula is inconsistent exhibit a somewhat limited practical scope with respect to really large and hard unsatisfiable SAT instances. Moreover, we cannot hope for such logically complete techniques to exhibit a polynomial behaviour in all situations unless P = NP.

In this respect, the contribution of this paper is twofold. The behaviour of local search algorithms has been analysed in the context of inconsistent SAT instances. Some recurrent phenomena have been pointed out when locally inconsistent problems were considered, i.e. problems whose inconsistency can be related to one of their subparts. In this context, two main results have been derived.

- First, an efficient heuristic has been discovered, allowing one to localize the inconsistent kernels in many locally inconsistent problems. This result is clearly of prime importance since inconsistent knowledge in real-life applications is often based on contradictions that are just local.
- This heuristic can boost the performance of complete techniques, in particular when dealing with large inconsistent SAT instances. More precisely, by combining this heuristic with the power of local search methods and with the completeness of standard SAT techniques, very good results are obtained with respect to classes of both consistent and inconsistent hard SAT instances.

The paper is organized as follows. First, some technical background about SAT and local search methods is briefly recalled. Then, it is shown that these search methods can help us to localize the inconsistent kernels of large SAT instances. Families of algorithms combining logically complete techniques with local search methods are then proposed. The experimental performance of a basic combination schema is illustrated on hard problems taken from standard benchmarks [7]. The scope of these results is then discussed before further promising paths of research are motivated.

2. Technical Background

SAT consists in checking the satisfiability of a boolean formula in conjunctive normal form (CNF). A CNF formula is a set (interpreted as a conjunction) of clauses, where a clause is a disjunction of literals. A literal is a positive or negated propositional variable.

An interpretation of a boolean formula is an assignment of truth values to its variables. A model is an interpretation that satisfies the formula. Accordingly, SAT consists in finding a model of a CNF formula when such a model does exist or in proving that such a model does not exist.

Recently, there has been a renewal of interest in designing efficient methods for hard SAT instances. On the one hand, several authors have improved logically complete techniques like DP. However, these techniques remain of a somewhat limited practical scope with respect to really large and hard SAT instances. In the sequel, we shall refer to improved versions of DP: namely, a version making use of the FFIS heuristics (i.e. First-Fail In Shortened Clauses) by [16] and C-SAT [8], which appeared to be the most efficient complete technique for SAT at the last DIMACS challenge [7].

On the other hand, logically incomplete techniques based on local search have been shown particularly efficient in proving large and hard consistent problems. Let us now briefly recall one of these methods, namely Selman et al.'s GSAT algorithm [17], [18]. This algorithm performs a greedy local search for a satisfying assignment of a set of propositional clauses. The algorithm starts with a randomly generated truth assignment. It then changes ("flips") the assignment of the variable that leads to the largest increase in the total number of satisfied clauses. Such flips are repeated until either a model is found or a preset maximum number of flips (MAX-FLIPS) is reached. This process is repeated as needed up to a maximum of MAX-TRIES times.

In the sequel, we shall consider two more recent variants of GSAT:

- First, the Random Walk Strategy GSAT [18], which outperforms basic GSAT procedures. This variant of GSAT selects the variable to be flipped in the following way: it either picks with probability p a variable occurring in some unsatisfied clause or follows, with probability $1-p$, the standard GSAT scheme, i.e. makes the best possible local move.
- Second, TSAT [14], which departs from basic GSAT by making an optimized use of a tabu list of variables in order to avoid recurrent flips and thus escape

Algorithm 1. GSAT: basic version

```
Procedure GSAT
Input : a set of clauses S, MAX-FLIPS, and MAX-TRIES
Output : a satisfying truth assignment of S, if found
Begin
   for i := 1 to MAX-TRIES
      I := a randomly generated truth assignment;
      for j := 1 to MAX-FLIPS
         if I satisfies S then return I;
         x := a propositional variable such that a change in its truth
              assignment gives the largest increase (possibly negative)
              in the number of clauses of S that are satisfied by I;
         I := I with the truth assignment of x reversed;
      end for;
   end for;
   return "no satisfying assignment found";
End.
```

from local minima. More precisely, TSAT keeps a fixed length -chronologically-ordered FIFO- list of flipped variables and prevents any of the variables in the list from being flipped again during a given amount of time. TSAT proves very competitive in most situations [14].

These very simple logically incomplete algorithms, which belong to the local search procedures family, are surprisingly efficient in demonstrating that CNF formulas are consistent.

3. Using GSAT-like Techniques to Detect and Locate Local Inconsistencies

In this section, it is shown that GSAT-like techniques can be used to localize inconsistent kernels of SAT instances, although the scope of such logically incomplete algorithms was generally expected to concern consistent problems, only.

The following test has been repeated very extensively and the following results have been obtained extremely often.

When TSAT (or any other GSAT-like algorithm) is run on a SAT instance, the following phenomenon can be observed when the algorithm fails to prove that it is consistent. A trace of TSAT is recorded: for each clause, taking each flip as a step of time, the number of times during which this clause is falsified is updated. A similar trace is recorded for each literal occurring in the SAT instance, counting the number of times it appears in the falsified clauses. Intuitively, it seemed to us that the most often falsified clauses should normally belong to an inconsistent kernel of the SAT instance if this instance is actually inconsistent. Likewise, it seemed to us that the literals that exhibit the highest scores should also take part in this kernel. Actually, these hypotheses prove experimentally correct extremely often.

This phenomenon can be summarized as follows. When GSAT-like algorithms are run on a locally inconsistent SAT instance, then the above counters allow us to split the SAT problem into two parts: a consistent one and an unsatisfiable one. For example, the clausal representation of the well-known inconsistent pigeons-holes problems [19] (8 pigeons; 56 variables and 204 clauses) has been mixed with the 8-queens problems (64 variables and 736 clauses), each problem making use of its own variables. The representation of each problem contains two kinds of clauses: namely, the positive ones (i.e. made of positive literals, only) and the binary negative ones (i.e. made of two negative literals). For example, the basic representation of the pigeons-holes problem asserts, on the one hand, that each pigeon should be in one hole (by means of positive clauses), and, on the other hand, that two pigeons cannot share a hole (using negative clauses). The number of times the different clauses have been falsified is shown in Figure 1. A significant gap can be seen between the scores of the pigeons-holes clauses and the scores of the 8-queens ones. Actually, the two parts of the mixed problem are clearly identified.

Moreover, in this specific case, for each part of the clausal representation (for example, the positive clauses of the pigeons-holes problem), the concerned clauses exhibit extremely close scores. In the diagram, the medium score are mentioned (actually, the four mentioned scores belong to [17189..18003], [639..828], [182..230] and [1..38], respectively for TSAT with MAX-TRIES = 20 and MAX-FLIPS = (number of variables)^2 . The span of these intervals shortens when the computing resources given to TSAT are increased). These close scores show us

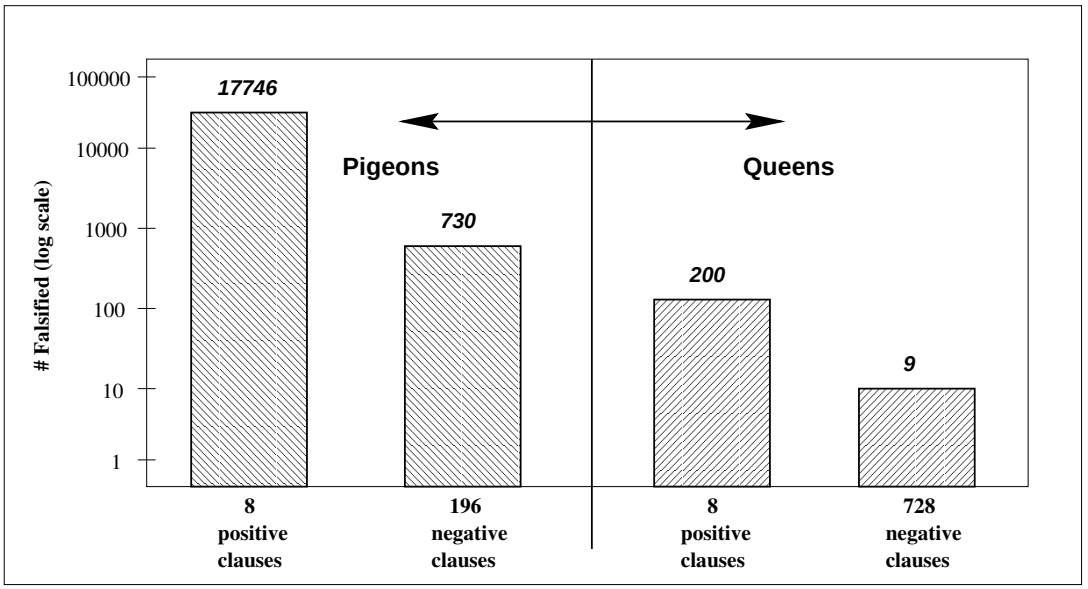

Figure 1. 8-Pigeons-holes + 8-Queens problems

that TSAT also exhibits the complete symmetry of each part of the representation of the pigeons-holes and queens problems. Actually, the trace of GSAT-related algorithms allows us to detect symmetries in SAT instances.

For less symmetrical and non-symmetrical problems, a significant gap between the scores of two parts of the clausal representation is still obtained, differentiating a probable inconsistent kernel from the remaining part of the problem. Let us stress that this phenomena also appear when both the consistent and the inconsistent parts of the SAT instance share the same variables.

Strangely enough, it appears thus that the trace of GSAT-like algorithms delivers the probable inconsistent kernel of locally inconsistent problems and sometimes allows us to detect and locate the presence of symmetries in SAT instances.

4. A Direct Approach: Focus on the Kernel

The most straightforward use of the above discovered feature to solve locally inconsistent problems is the following one. Use GSAT-like algorithms to circumscribe the probable inconsistent kernel. Then, apply complete techniques to this kernel to prove its unsatisfiability and thus, consequently, the inconsistency of the global problem. The scores delivered by the GSAT-like algorithm can be used to guide the branching strategy performed by the complete technique, by selecting

the literals with the highest scores first.

Obviously, this simple schema can only be used when the discovered probable inconsistent kernel is of manageable size and when the gap between the score of the clauses of the kernel and the score of the remaining clauses is large enough. Also, the clauses can be sorted according to their decreasing scores; incremental complete techniques can be applied on them until unsatisfiability is proved. In this respect, clauses outside the discovered kernel could also be taken into account. A somewhat similar approach has been defined independently in [5].

5. A Basic Combination Schema

Let us now describe a basic algorithm using GSAT-like procedures to guide the branching strategy of logically-complete algorithms based on DP.

TSAT [14] is run to deliver the next literal to be assigned by DP. This literal is selected as the one with the highest score as explained above. Such an approach can be seen as using the trace of TSAT as an heuristic for selecting the next literal to be assigned by DP, and a way to extend the partial assignment made by DP towards a model of the SAT instance when this instance is satisfiable. Each time DP needs to select the next variable to be considered, such a call to TSAT can be performed with respect to the remaining part of the SAT instance. This algorithm is given in Algorithm 2.

Algorithm 2. DP + TSAT: basic version

Procedure *DP + TSAT*
Input : a set of clauses S
Output : a satisfying truth assignment of S, if found
or a definitive statement that S is inconsistent
Begin
Unit_propagate(S);
if the empty clause is generated **then return** (false);
else if all variables are assigned **then return** (true)
else begin
if TSAT(S) succeeds **then return** (true)
else begin
p := the most often falsified literal during TSAT;
return (DP+TSAT(S$\wedge$p) $\vee$ DP+TSAT(S$\wedge\neg$p));

```
                    end;
                end;
    End
```

6. Experimental Results[1]

In this section, experimental results about the application of the above new logically complete algorithm to many classes of problems are presented. Let us stress that our goal in conducting these tests was simply to check the feasibility of using GSAT-like techniques to guide DP. In this respect, a basic form of combination has been tested, which can be improved in many directions as this will be described in the next section.

First, DP+TSAT has been run and compared with DP + FFIS and C-SAT (with default option) on various large inconsistent problems from the DIMACS benchmarks [7]. This benchmark consists of various SAT instances: problems from real-life applications, academic and random ones, many of them being out of reach of the most efficient techniques, in particular complete ones. In Table 1, a significant sample of our extensive experimentations is given, showing the obtained dramatical performance improvement, in particular for classes of inconsistent SAT instances. Also, very good results are obtained for consistent instances; indeed, DP+TSAT is as efficient as local search techniques since DP+TSAT begins with a call to them. Extensive results on DIMACS benchmarks are given in the appendix.

Let us comment a few examples from Table 1.

The BF1355-638 problem (2177 variables and 4768 clauses) is an actual problem from circuit fault analysis proposed by Allen Van Gelder and Yumi Tsuji. C-SAT and DP+FFIS failed to prove that this problem is inconsistent, within 73H and 17H CPU time, respectively. On the other hand, DP+TSAT

[1] All the algorithms mentioned in this paper are implemented in a common platform, available from the authors, written in C under Linux 1.1.53 (except that we reused C-SAT, the original DIMACS'93 implementation by Dubois). We have thus implemented the TSAT, DP+TSAT and DP+FFIS procedures, this latter one being at least as efficient as Rauzy's original one. All experimentations have been conducted on 133 Pentium PCs.

Table 1
DIMACS problems[2]

Instances	Sat	Size		Inc. Ker.[3]		C-Sat	DP+FFIS			DP+TSAT		
		Var.	Cla.	Var.	Cla.	time	assign.	choices	time	assign.	choices	time
AIM series:												
1_6-no-3	No	100	160	51	57	13s32	3E+07	2E+06	214s71	178	16	0s26
1_6-yes1-2	Yes	100	160	***	***	0s00	495858	30052	4s39	77	6	0s08
2_0-no-1	No	100	200	18	19	313s24	4E+07	2E+06	349s52	46	5	0s10
2_0-yes1-1	Yes	100	200	***	***	8s54	706388	31274	7s41	81	8	0s15
1_6-no-1	No	200	320	52	55	20116s30	***	***	>8h	240	16	0s58
1_6-yes1-3	Yes	200	320	***	***	>7h	***	***	>9h	232	11	0s32
2_0-no-3	No	200	400	35	37	>20h	***	***	>15h	120	10	0s42
2_0-yes1-1	Yes	200	400	***	***	>8h	2E+09	7E+07	21859s45	291	27	1s21
1_6-no-1	No	50	80	20	22	0s19	12072	895	0s09	72	8	0s06
1_6-yes1-1	Yes	50	80	***	***	0s07	1540	84	0s01	37	6	0s05
2_0-no-1	No	50	100	21	22	0s30	54014	2759	0s43	52	5	0s05
2_0-yes1-1	Yes	50	100	***	***	0s19	2878	176	0s03	11	3	0s03
BF series:												
0432-007	No	1040	3668	674	1252	463s67	9E+08	6E+06	19553s44	115766	870	85s25
1355-075	No	2180	6778	82	185	88035s05	317628	2047	18s88	4602	28	26s23
1355-638	No	2177	4768	83	154	>73h	***	***	>17h	6192	32	32s57
2670-001	No	1393	3434	79	139	11s45	***	***	>25h	490692	4822	519s40
SSA series:												
0432-003	No	435	1027	306	320	1s70	133794	1570	1s79	1338	16	0s80
2670-130	No	1359	3321	552	669	2053s72	***	***	>33h	2E+07	79426	8040s64
2670-141	No	986	2315	579	1247	3689s65	3E+08	2E+06	6350s77	1E+07	92421	6639s44
7552-038	Yes	1501	3575	***	***	195s75	***	***	>13h	29	1	0s34
7552-158	Yes	1363	3034	***	***	97s59	1639	78	0s19	12	1	0s29
7552-159	Yes	1363	3032	***	***	98s82	1557	84	0s21	12	1	0s25
7552-160	Yes	1391	3126	***	***	159s75	1457	76	0s18	1	1	0s30

takes 32 sec. only.

The AIM200-1_6-yes1-3 problem (200 variables and 320 clauses) by [13] is a 3-SAT instance. All AIM problems exhibit exactly one model when satisfiable. DP+TSAT proves within 0.32 sec. that the above mentioned AIM problem is satisfiable while DP + FFIS and C-SAT gave up after 9 and 7H, respectively.

The AIM200-2_0-no-3 problem (200 variables and 400 clauses) by [13] is an inconsistent 3-SAT instance. DP+TSAT takes 0.42 sec. to prove that this problem is inconsistent, whereas we gave up with DP+FFIS and C-SAT after 15 and 20 H, respectively.

[2] In the table, "$> n$ H" means that we gave up after the problem had not been solved within n hours of CPU time.

[3] "Inc. Ker." in the table means "Inconsistent Kernel".

7. Natural Optimizations

Although the above mentioned results are extremely positive, it should be clear that the tests were only conducted for checking the feasibility of mixing local search techniques with logically complete methods. We did not try to optimize the way this mixing is performed. Further significant performance improvements can be expected by fine-tuning the parameters of the involved local search techniques and by defining an optimal balance between the time spent by DP and by the local search techniques. More precisely, the following natural optimizations can be envisioned.

- In DP+TSAT, the literal that exhibits the highest score in the trace of TSAT was selected to guide DP. In this respect, a call to TSAT is performed each time DP needs to select a literal to assign. At the opposite, just one call to TSAT can be made and decreasing scores of literals can be used to guide DP at each step. Between these two extreme points of view, an optimal attitude must be found, limiting the number of calls to GSAT-like algorithms. This optimal balance should depend on both the form of the trace given by this GSAT-like algorithm and on the point reached in the search tree by DP.
- A second possible optimization lies in the fine-tuning of the local search parameters with respect to the remaining problem left by DP.

8. Actual Scope of These Techniques

Clearly, the results presented in this paper concern consistent and locally inconsistent problems, i.e. problems whose inconsistency can be related to one of their subparts. Let us define the dual concepts of local and global inconsistency for SAT instances.

Definition 1.

A SAT instance S is *globally inconsistent*

iff

S is inconsistent and $\forall S' \subset S : S'$ is consistent.

When an inconsistent SAT instance S is not globally inconsistent, it is said to be locally inconsistent.

In this paper, locally inconsistent SAT instances have been considered. Indeed, such a form of inconsistency is of prime importance in actual applications. Very often, inconsistency is due to the accidental presence of a few pieces of contradictory information about a given subject. Clearly, several measures of locality for inconsistency can be defined, making use of e.g. the (size of the inconsistent kernel)/(size of the SAT instance) ratio. We are currently analyzing how the form and properties of the trace of local search techniques can be experimentally related to graded notions of locality for inconsistency. Let us stress that the techniques presented in this paper do not require this ratio to be negligible, as the ratio of the combination of the pigeons-holes and 8-queens problems illustrates it. Also, the size of the kernel need not be small.

In Table 1 and in the Appendix, we give the size of (one of) the globally inconsistent kernels for each inconsistent problem that DP+TSAT managed to solve. To obtain this information, first we used the trace of TSAT to get a first inconsistent kernel using a dichotomy-like approach. Then, using complete methods, we reduced the kernel to a subpart that we proved to be globally inconsistent. Let us stress that most often this resulting kernel proved to be close to the initial one, showing once again the relevance of the heuristic described in this paper.

As expected, DP+TSAT managed to prove inconsistent problems, where one globally inconsistent kernel is of moderate size. Let us note that sometimes DP+FFIS and C-SAT did not manage to solve them: see for example the AIM-200-2_0-no-3 where the size of one globally inconsistent kernel is just made of 37 clauses refering to 35 variables. Interestingly enough, DP+TSAT also gave rise to performance improvement with respect to problems involving a large globally inconsistent kernel: see for example the bf0432-007 problem that DP+TSAT proved to be inconsistent, addressing a globally inconsistent kernel made of 1252 clauses making use of 674 variables.

Obviously, large globally inconsistent problems are the most difficult to handle; they are often generated in an artificial manner since they are scarce in real life applications (see e.g. the pigeons-holes problem [4], Tseitin and Urqhart's formulas [19], etc.). Really significant progress in dealing with large globally inconsistent problems requires a better understanding of their nature and properties. However, the results presented in this paper also apply to globally inconsistent problems; at least to some extent, as we have illustrated it earlier when we discussed the size of the discovered inconsistent kernels of DIMACS benchmarks. Also, when DP+TSAT (which is once again a rough non-optimized combination

schema) did not give rise to better CPU-time on globally inconsistent problems like Dubois' ones, it allowed smaller search trees to be generated. In the same vein, we hope that some progress could be made with respect to hard random problems. In particular, we hope that some little progress could be obtained in the treatment of inconsistent K-SAT instances at the transition phase in the fixed-length clause model [9], at least, as far as locally inconsistent instances are still actually under consideration. On the other hand, we are very optimistic in making good progress in solving inconsistent random K-SAT instances at the right of the transition phase.

9. Conclusion

The contribution of this paper is twofold. On the one hand, an efficient heuristic-based technique has been proposed, allowing one to detect and locate local inconsistencies in sets of propositional clauses. We think that such a technique should be useful with respect to many computer science domains. For example, this should make the handling of local inconsistencies in logical knowledge bases possible. On the other hand, using local search techniques, the feasibility of boosting complete techniques to prove SAT instances has been demonstrated, in particular large locally inconsistent ones that were out of reach of previous approaches. Moreover, the technique presented in this paper also appears competitive for solving classes of consistent SAT instances. Additionally, further possible optimizations that could lead to additional significant performance improvements have been discussed.

Acknowledgements

This work has been supported by the Ganymède II project of the Contat de Plan Etat/Nord–Pas-de-Calais, by the MESR (Ministère de l'Enseignement Supérieur et de la Recherche) and by the IUT de Lens. We express our gratitute to our colleague J.-L. Coquidé for his help in providing us with access to computing facilities at the IUT de Lens, allowing our extensive tests to be performed.

References

[1] P. Cheeseman and B. Kanefsky and W.M. Taylor, Where the Really Hard Problems are, in: *Proc. IJCAI-91*, pp. 163-169, 1991.

[2] V. Chvátal and E. Szemerédi, Many Hard Examples for Resolution, in: *Journ. of the ACM*, vol. 33, no. 4, pp. 759-768, 1988.

[3] S. Cook, The Complexity of Theorem-Proving Procedures, in: *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, pp. 151-158, 1971.

[4] S. Cook, A Short Proof of the Pigeon Hole Principle Using Extended Resolution, in: *SIGACT News*, vol. 8, pp. 28-32, 1976.

[5] J.M. Crawford, Solving Satisfiability Problems Using a Combination of Systematic and Local Search, in: *Working notes of the DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*, 1993.

[6] M. Davis, H. Putnam, A Computing Procedure for Quantification Theory, in: *Journ. of the ACM*, vol. 7, pp. 201-215, 1960.

[7] DIMACS, Second challenge organized by the Center for Discrete Mathematics and Computer Science of Rutgers University in 1993 (The benchmarks used in our tests can be obtained by anonymous ftp from Rutgers University DIMACS Center: ftp://dimacs.rutgers.edu/pub/challenge/sat/benchmarks/cnf).

[8] O. Dubois and P. André and and Y. Boufkhad and J. Carlier, SAT versus UNSAT, in: *Working notes of the DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*, 1993.

[9] O. Dubois and J. Carlier, Probabilistic Approach to the Satisfiability Problem, in: *Theoretical Computer Science*, vol. 81, pp. 65-75, 1991.

[10] J. Franco and M. Paull, Probabilistic Analysis of the Davis and Putnam Procedure for Solving the Satisfiability Problem, in: *Discrete Applied Math.*, vol. 5, pp. 77-87, 1983.

[11] I.P. Gent and T. Walsh, Towards an Understanding of Hill-climbing Procedures for SAT, in: *Proc. AAAI-93*, pp.28-33, 1993.

[12] I.P. Gent and T. Walsh, The SAT Phase Transition, in: *Proc. ECAI-94*, pp. 105-109, 1994.

[13] K. Iwama and E. Miyano, Test-Case Generation with Proved Securities, in: *Working notes of the DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*, 1993.

[14] B. Mazure and L. Saïs and É. Grégoire, Tabu Search for SAT, in: *Proc. AAAI-97*, pp. 281-285, July 1997. (A preliminary version appeared as: TWSAT: a New Local Search Algorithm for SAT. Performance and Analysis, in: *CP95 Workshop on Studying and Solving Really Hard Problems*, Cassis (France), September 1995.)

[15] D. Mitchell and B. Selman and H. Levesque, Hard and Easy Distributions of SAT Problems, in: *Proc. AAAI-92*, pp. 459-465, 1992.

[16] A. Rauzy, On the Complexity of the Davis and Putnam's Procedure on Some Polynomial Sub-Classes of SAT, in: *LaBRI Technical Report 806-94*, Université de Bordeaux 1, 1994.

[17] B. Selman and H. Levesque and D. Mitchell, A New Method for Solving Hard Satisfiability Problems, in: *Proc. AAAI-92*, pp. 440-446, 1992.

[18] B. Selman and H.A. Kautz and B. Cohen, Local Search Strategies for Satisfiability Testing, in: *Working notes of the DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*, 1993.

[19] G.S. Tseitin, On the Complexity of Derivations in Propositional Calculus, in: Slisenko A.O. (ed.), *Structures in Constructive Mathematics, Part II*, pp. 115-125, 1968.

Appendix

In the next table:

- "Inc. Ker." means "Inconsistent Kernel".
- "$> n$ H" means that we gave up after the problem had not been solved within n hours of CPU time,
- "***" in:
  - "Inc. Ker." column means that the instance is satisfiable,
  - other columns means that the method fails to solve the instance,
- "???" in Inc. Ker. column means that DP+TSAT fails to detect an inconsistent kernel.

Table 2
DIMACS problems

Instances	Sat	Size		Inc. Ker.		C-Sat	DP+FFIS			DP+TSAT		
		Var.	Cla.	Var.	Cla.	time	assign.	choices	time	assign.	choices	time
Dubois series:												
dubois10	No	30	80	30	80	0s22	20564	2063	0s15	8728	641	1s30
dubois11	No	33	88	33	88	0s44	41044	4111	0s31	21800	1691	3s34
dubois12	No	36	96	36	96	0s87	82004	8207	0s59	25228	1835	4s48
dubois13	No	39	104	39	104	1s79	163924	16399	1s18	47236	3091	8s69
dubois14	No	42	112	42	112	3s67	327764	32783	2s49	99888	7693	14s39
dubois15	No	45	120	45	120	7s39	655444	65551	5s15	147180	9507	26s45
dubois16	No	48	128	48	128	15s22	1E+06	131087	10s09	303108	20733	65s97
dubois17	No	51	136	51	136	31s70	3E+06	262159	19s69	348068	23167	76s86
SSA series:												
0432-003	No	435	1027	306	320	1s70	133794	1570	1s79	1338	16	0s80
2670-130	No	1359	3321	552	669	2053s72	***	***	>33h	2E+07	79426	8040s64
2670-141	No	986	2315	579	1247	3689s65	3E+08	2E+06	6350s77	1E+07	92421	6639s44
6288-047	No	10410	34238	???	???	>24h	***	***	>24h	***	***	>24h
7552-038	Yes	1501	3575	***	***	195s75	***	***	>13h	29	1	0s34
7552-158	Yes	1363	3034	***	***	97s59	1639	78	0s19	12	1	0s29
7552-159	Yes	1363	3032	***	***	98s82	1557	84	0s21	12	1	0s25
7552-160	Yes	1391	3126	***	***	159s75	1457	76	0s18	1	1	0s30
BF series:												
0432-007	No	1040	3668	674	1252	463s67	9E+08	6E+06	19553s44	115766	870	85s25
1355-075	No	2180	6778	82	185	88035s05	317628	2047	18s88	4602	28	26s23

Instances	Sat	Size		Inc.	Ker.	C-Sat	DP+FFIS			DP+TSAT		
		Var.	Cla.	Var.	Cla.	time	assign.	choices	time	assign.	choices	time
1355-638	No	2177	4768	83	154	>73h	***	***	>17h	6192	32	32s57
2670-001	No	1393	3434	79	139	11s45	***	***	>25h	490692	48220	519s4
AIM series (100 variables):												
1_6-no-1	No	100	160	43	47	0s89	7E+06	323296	50s30	174	13	0s22
1_6-no-2	No	100	160	46	52	0s09	3E+06	167445	23s38	178	19	0s34
1_6-no-3	No	100	160	51	57	13s32	3E+07	2E+06	214s71	178	16	0s26
1_6-no-4	No	100	160	43	48	0s05	1E+07	728908	97s59	126	14	0s25
1_6-yes1-1	Yes	100	160	***	***	0s04	135966	6663	1s09	138	8	0s12
1_6-yes1-2	Yes	100	160	***	***	0s00	495858	30052	4s39	77	6	0s08
1_6-yes1-3	Yes	100	160	***	***	0s04	604	34	0s01	112	12	0s18
1_6-yes1-4	Yes	100	160	***	***	0s04	159646	7707	1s46	124	6	0s08
2_0-no-1	No	100	200	18	19	313s24	4E+07	2E+06	349s52	46	5	0s10
2_0-no-2	No	100	200	35	39	72s19	3E+07	1E+06	294s09	108	9	0s20
2_0-no-3	No	100	200	25	27	274s14	9E+06	394649	80s77	68	6	0s12
2_0-no-4	No	100	200	26	31	0s05	3E+07	1E+06	233s29	80	9	0s19
2_0-yes1-1	Yes	100	200	***	***	8s54	706388	31274	7s41	81	8	0s15
2_0-yes1-2	Yes	100	200	***	***	1s85	238794	10305	2s79	91	10	0s20
2_0-yes1-3	Yes	100	200	***	***	5s19	138	10	0s00	84	6	0s12
2_0-yes1-4	Yes	100	200	***	***	0s70	270	18	0s00	163	9	0s13
3_4-yes1-1	Yes	100	340	***	***	0s10	996	30	0s02	1	2	0s08
3_4-yes1-2	Yes	100	340	***	***	0s15	2366	74	0s06	0	1	0s00
3_4-yes1-3	Yes	100	340	***	***	0s10	5484	193	0s14	0	1	0s01
3_4-yes1-4	Yes	100	340	***	***	0s10	196	17	0s01	0	1	0s02
6_0-yes1-1	Yes	100	600	***	***	0s15	100	4	0s01	0	1	0s01
6_0-yes1-2	Yes	100	600	***	***	0s27	248	7	0s01	0	1	0s00
6_0-yes1-3	Yes	100	600	***	***	0s12	224	11	0s01	0	1	0s01
6_0-yes1-4	Yes	100	600	***	***	0s12	212	11	0s01	0	1	0s00
AIM series (200 variables):												
1_6-no-1	No	200	320	52	55	20116s30	***	***	>8h	240	16	0s58
1_6-no-2	No	200	320	77	80	0s17	***	***	>15h	262	24	0s88
1_6-no-3	No	200	320	77	83	0s70	***	***	>8h	302	33	1s15
1_6-no-4	No	200	320	44	46	0s04	***	***	>17h	184	16	0s61
1_6-yes1-1	Yes	200	320	***	***	>12h	210	10	0s01	222	10	0s25
1_6-yes1-2	Yes	200	320	***	***	0s04	682	34	0s02	240	14	0s44
1_6-yes1-3	Yes	200	320	***	***	>7h	***	***	>9h	232	11	0s32
1_6-yes1-4	Yes	200	320	***	***	0s04	6E+08	3E+07	5567s85	244	13	0s42
2_0-no-1	No	200	400	49	53	>7h	***	***	>15h	136	11	0s47
2_0-no-2	No	200	400	46	50	>20h	***	***	>8h	186	15	0s67
2_0-no-3	No	200	400	35	37	>20h	***	***	>15h	120	10	0s42
2_0-no-4	No	200	400	36	42	0s07	***	***	>8h	144	13	0s55
2_0-yes1-1	Yes	200	400	***	***	>8h	2E+09	7E+07	21859s45	291	27	1s21
2_0-yes1-2	Yes	200	400	***	***	8273s44	2E+08	7E+06	2809s87	290	29	1s18
2_0-yes1-3	Yes	200	400	***	***	1197s97	4960	217	0s09	444	20	0s98

Instances	Sat	Size		Inc.	Ker.	C-Sat	DP+FFIS			DP+TSAT		
		Var.	Cla.	Var.	Cla.	time	assign.	choices	time	assign.	choices	time
2_0-yes1-4	Yes	200	400	***	***	54296s32	272472	11783	4s79	319	27	1s07
3_4-yes1-1	Yes	200	680	***	***	0s52	210956	5409	6s88	0	1	0s06
3_4-yes1-2	Yes	200	680	***	***	0s29	8896	272	0s37	0	1	0s07
3_4-yes1-3	Yes	200	680	***	***	0s70	1302	36	0s05	2	3	0s22
3_4-yes1-4	Yes	200	680	***	***	0s35	267786	6270	8s44	0	1	0s05
6_0-yes1-1	Yes	200	1200	***	***	0s57	6110	75	0s25	0	1	0s01
6_0-yes1-2	Yes	200	1200	***	***	0s22	10284	134	0s47	0	1	0s01
6_0-yes1-3	Yes	200	1200	***	***	0s54	16704	214	0s72	0	1	0s01
6_0-yes1-4	Yes	200	1200	***	***	0s34	16412	232	0s72	0	1	0s04
AIM series (50 variables):												
1_6-no-1	No	50	80	20	22	0s19	12072	895	0s09	72	8	0s06
1_6-no-2	No	50	80	28	32	0s02	11408	782	0s10	86	9	0s07
1_6-no-3	No	50	80	28	31	0s07	54610	4525	0s41	92	12	0s09
1_6-no-4	No	50	80	18	20	0s02	7230	447	0s05	56	7	0s06
1_6-yes1-1	Yes	50	80	***	***	0s07	1540	84	0s01	37	6	0s05
1_6-yes1-2	Yes	50	80	***	***	0s02	5674	384	0s05	50	3	0s02
1_6-yes1-3	Yes	50	80	***	***	0s04	50	5	0s01	50	4	0s04
1_6-yes1-4	Yes	50	80	***	***	0s02	70	5	0s01	1	2	0s01
2_0-no-1	No	50	100	21	22	0s30	54014	2759	0s43	52	5	0s05
2_0-no-2	No	50	100	28	30	0s07	19342	974	0s17	80	7	0s07
2_0-no-3	No	50	100	22	28	0s22	15254	814	0s13	76	6	0s06
2_0-no-4	No	50	100	18	21	0s02	30034	1645	0s24	60	6	0s06
2_0-yes1-1	Yes	50	100	***	***	0s19	2878	176	0s03	11	3	0s03
2_0-yes1-2	Yes	50	100	***	***	0s07	432	29	0s01	0	1	0s00
2_0-yes1-3	Yes	50	100	***	***	0s10	7616	446	0s07	12	3	0s02
2_0-yes1-4	Yes	50	100	***	***	0s04	92	8	0s00	0	1	0s00
3_4-yes1-1	Yes	50	170	***	***	0s04	496	20	0s01	0	1	0s00
3_4-yes1-2	Yes	50	170	***	***	0s07	406	13	0s01	0	1	0s00
3_4-yes1-3	Yes	50	170	***	***	0s02	378	16	0s01	0	1	0s01
3_4-yes1-4	Yes	50	170	***	***	0s05	282	13	0s00	0	1	0s00
6_0-yes1-1	Yes	50	300	***	***	0s09	124	4	0s01	0	1	0s00
6_0-yes1-2	Yes	50	300	***	***	0s07	282	8	0s01	0	1	0s01
6_0-yes1-3	Yes	50	300	***	***	0s09	82	4	0s01	0	1	0s00
6_0-yes1-4	Yes	50	300	***	***	0s07	76	4	0s01	0	1	0s00

About the Use of Local Consistency in Solving CSPs

À paraître dans les actes de « *The twelfth IEEE International Conference on Tools with Artificial Intelligence (ICTAI'00)* », Vancouver (British Columbia, Canada), novembre 2000, « IEEE Computer Society ».

About the Use of Local Consistency in Solving CSPs

Assef Chmeiss and Lakhdar Saïs
CRIL - Université d'Artois - IUT de Lens
Rue de l'université - SP 16
62307 Lens Cedex
{chmeiss, sais}@cril.univ-artois.fr

Abstract

Local consistency is often a suitable paradigm for solving Constraint Satisfaction Problems. Many local consistency based techniques (such as Arc Consistency) have been proposed and widely studied in order to reduce the search space. In this paper, we show how search algorithms could be improved, thanks to a smart use of two filtering techniques (Path Consistency and Singleton Arc Consistency). In the first part of this paper, we propose a possible way to get benefits from using a partial form of path consistency during the search. In the second part, we show how local treatment based on Singleton Arc Consistency (SAC) can be used to achieve more powerful filtering as well as to refine variable ordering heuristics.

1 Introduction

Constraint-satisfaction problems (CSPs) involve the assignment of values to variables which are subject to a set of constraints. Examples of CSPs are map coloring, conjunctive queries in a relational database, line drawings understanding, pattern matching in production rules systems, combinatorial puzzles, etc. CSP is known as NP-complete. CSPs are normally solved by different versions of backtrack search. In this case, if d is the size of domains (maximum number of values in domains D_i) and n is the number of variables, the theoretical time complexity of search is then bounded by the size of the search space, that is d^n. Consequently, many works have been proposed to optimize the classical backtrack procedure, as constraint propagation, intelligent backtracking or network decompositions. Currently, it seems that the most efficient algorithm is the MAC procedure which was introduced by Sabin and Freuder [20]. Other approaches use decomposition techniques based on structural properties of the CSP. These methods take advantage of the fact that the tractability of CSPs is intimately connected to the topological structure of their underlying constraint graph[6] [8]. Another decomposition technique based on the micro-structure of the constraints graph has been proposed by Jégou[14]. Other works concerning the characterization of tractable classes have been proposed, they exploit the constraint semantics (functional constraints, etc...) or structural graph properties (trees). We mention that some filtering techniques are complete on a part of these polynomial classes. For example, arc consistency is sufficient to solve tree graphs CSPs, 0/1/All constraints[16] can be solved in polynomial time using path consistency. However, practical use of local consistency techniques in order to improve the search algorithms, must make a compromise between their efficiency (reduction of search space) and their space and/or time complexity induced by the additional treatments. Note that the filtering power (consistency level) is directly related to its time complexity. This point explains the fact that arc consistency and path consistency are the most studied concepts. The most efficient general CSPs search algorithm (MAC), has been obtained by maintaining arc consistency during search [20][2]. Furthermore, the most important progress in the resolution of SAT problem (satisfiability of a boolean formula in conjunctive normal form) has been obtained thanks to a better exploitation of unit propagation (as local treatment). For example CSAT, POSIT, etc. Others have been achieved, using a limited form of resolution (for example, bounded resolution, resolution on binary clauses). We point out that resolution (production of resolvants) has been for a long time avoided for reasons of time and space complexity, it seems now to be a promising way of improving the performance of SAT algorithms.

The last point leads us to look for the similarity between resolution and path consistency which can be seen as a constraint production and verification process. Besides, for the same reasons as for resolution, integrating path consistency in CSPs search algorithms has been up to now avoided.

The aim of this paper is twofold. On the one hand, we propose an extension of MAC algorithm in exploiting a partial form of path consistency (section 3); On the other hand, we

propose in section 4 a local treatment based on singleton arc consistency filtering which allows to achieve more powerful filtering. Also, we show how such a treatment can be used to deduce new and pertinent information allowing to refine ordering variable heuristics. Basing on preliminary experiments, some observations about the usefulness of the proposed approach are presented in section 5. A conclusion on this work and perspectives are given in section 7.

2 Definitions and notations

A finite CSP (Constraint Satisfaction Problem)[17] is defined as a four tuple $\mathcal{P} = (X, D, C, R)$. X is a set of n variables $x_1, x_2, \ldots x_n$. D is a set of finite domains $D_1, D_2, \ldots D_n$, and C is a set of m constraints. Here we only consider binary CSPs, that is, the constraints are defined on pairs of variables $\{x_i, x_j\}$ and will be denoted C_{ij}. To each constraint C_{ij}, we associate a subset of the Cartesian product $(D_i \times D_j)$ which is denoted R_{ij} and specifies which values of the variables are compatible with each other. R is the set of all R_{ij}. A *solution* is an assignment of values to variables which satisfies all the constraints. For a binary CSP $\mathcal{P}$, the pair (X, C) is a graph called the *constraint graph*. Given a CSP, the problem is either to decide if a solution exists, to find a solution, or to find all solutions.

We recall below the definition of arc and path consistency filtering :

Definition 1 *A domain D_i of D is arc-consistent iff, $\forall a \in D_i, \forall x_j \in X$ such that $C_{ij} \in C$, there exists $b \in D_j$ such that $R_{ij}(a, b)$. A CSP is arc-consistent iff $\forall D_i \in D, D_i \neq \emptyset$ and D_i is arc-consistent.*

Definition 2 *A pair of variables $\{x_i, x_j\}$ is path-consistent iff $\forall (a, b) \in R_{ij}, \forall x_k \in X$, there exists $c \in D_k$ such that $R_{ik}(a, c)$ and $R_{jk}(b, c)$. A CSP is path-consistent iff $\forall x_i, x_j \in X$ the pair $\{x_i, x_j\}$ is path-consistent.*

While filtering based on arc-consistency removes values from domains if they do not satisfy arc-consistency, filtering based on path-consistency removes pairs of values from relations if they do not satisfy path-consistency. So, if a constraint is not defined between a pair of variables, the universal relation is then considered. As a consequence, the constraint graph can be completed in such cases.

3 M*AC+*: an extension of MAC by Path Consistency

It was shown that maintaining arc consistency during search can significantly improve the efficiency of solving algorithms. On the contrary, path consistency technique, although it is very useful for some applications, namely temporal reasoning in interval algebra of Allen[1], has never been used during the search phase in binary CSPs algorithms. This is due to its time complexity and to complicated data structures used by PC algorithms.

Nevertheless, it was shown that path consistency is sufficient to treat some CSPs polynomial classes. So, 0/1/All CSPs can easily be solved with the help of a path consistency preprocessing algorithm. Also, path consistent CSPs of which the constraints graph is of width 2 [11] can be solved using a backtrack-free algorithm.

Path consistency can be compared to resolution, a widely used principle in the theorem proving community, in the sense that it permits to produce new constraints (clauses or resolvants). Recently, this principle has been integrated in the enumerative methods (as Davis and Putnam procedure) which has significantly improved their efficiency, particularly on structured real world problems. We can cite the resolution on binary clauses, bounded resolution (used as preprocessing phase), etc.

This similarity with SAT leads us to think that the addition of new constraints could improve the performance of search algorithms on finite binary CSPs.

In this section, we propose a way to exploit, during the search, a *partial directed* form of path consistency propagation. This filtering allows us to verify some constraints and deduct others.

We recall below some definitions :

Definition 3 *A variable $x_i \in X$ is singleton value if $|D_i| = 1$*

Definition 4 *A variable $x_i \in X$ is singleton variable if its degree equals 1*

Definition 5 *A constraints graph is directed-path-consistency [7] with respect to the order $(x_1, x_2, \ldots, x_n)$, if for every couple of variables (x_i, x_j) and every couple of values (a, b) s.t. $R_{ij}(a, b)$, for all $k > i, j$, there is a value $c \in D_k$, s.t. $R_{ik}(a, c)$ and $R_{jk}(b, c)$*

We now introduce some definitions and properties which will be used in the following:

Definition 6 *A variable $x_i \in X$ is doubleton variable if its degree equals 2*

Definition 7 *(inference) Let $\mathcal{P} = (X, D, C, R)$ a CSP, $C_{ij} \in C$ and $C_{ik} \in C$. C_{jk} is implied by C_{ij} and C_{ik} (denoted $(C_{ij}, C_{ik}) \Rightarrow C_{jk}$) if $\forall (a, b) \in R_{jk}, \exists v \in D_i$ s.t. $(v, a) \in R_{ij}$ and $(v, b) \in R_{ik}$.*

Property 1 *Let $\mathcal{P} = (X, D, C, R)$ a CSP, $x_i \in X$ a doubleton variable, s.t. $(C_{ij}, C_{ik}) \Rightarrow C_{jk}$, then $\mathcal{P} = (X, D, C, R)$ is consistent if and only if $\mathcal{P}' = (X - \{x_i\}, D - \{D_i\}, C - \{C_{ij}, C_{ik}\}, R - \{R_{ij}, R_{ik}\})$ is consistent.*

Proof:

- $\Rightarrow$) $\mathcal{P}$ is consistent, $\exists s = (v_1, v_2, \ldots, v_i, \ldots, v_n) \in D_1 \times D_2 \times \ldots \times D_i \times \ldots \times D_n)$ a solution of $\mathcal{P}$. It is obvious that the solution s' obtained from s after removing x_i satisfies $\mathcal{P}'$, since all constraints of $\mathcal{P}'$ are also in $\mathcal{P}$.
- $\Leftarrow$) since $\mathcal{P}'$ is consistent, $\exists s' = (v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n) \in D_1 \times D_2 \times \ldots \times D_{i-1} \times D_{i+1} \times \ldots \times D_n)$ a solution of $\mathcal{P}'$. s' can be extended to s a solution of $\mathcal{P}$. In fact, since x_i is a doubleton, $(C_{ij}, C_{ik}) \Rightarrow C_{jk}$, then $s = (v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_n) \in D_1 \times D_2 \times \ldots \times D_{i-1} \times D_i \times D_{i+1} \ldots \times D_n)$ s.t. $(v_i, v_j) \in R_{ij}$ and $(v_i, v_k) \in R_{ik}$ is a solution of $\mathcal{P}$. Such value v_i exists (see definition7).

$\diamond$

Note that the previous property can be used in different ways (preprocessing, during the search) by search algorithms.

Let $\mathcal{P}$ be a CSP, $x_i \in X$ a doubleton variable s.t. $C_{ij} \in C$ and $C_{ik} \in C$. To exploit property1, we distinguish two cases :

1. $C_{jk} \in C$, we must *verify* that the constraint C_{jk} is implied by (C_{ij}, C_{ik}).
2. $C_{jk} \notin C$, we must *add*[1] the constraint C_{jk} s.t. (C_{ij}, C_{ik}) $\Rightarrow C_{jk}$. Note that when C_{jk} defines the universal relation, its addition is useless.

We underline that applying directed path-consistency is sufficient to exploit property1. In fact, for a doubleton variable x_i, it is not necessary to verify a complete path consistency on x_i and its two neighbors x_j and x_k.

We propose here an extended version M*AC*+ of MAC search algorithm. In addition to singleton variables, it exploits dynamically property1 (doubleton variables). So, we guarantee that, at every level of the search tree, the problem restricted on uninstantiated variables contains neither singleton (values or variables) nor doubleton. In other words, for every uninstantiated variable x_i, we have $|D_i| > 1$ and $degree(x_i) > 2$.

[1] a couple (a, b) is added to R_{jk} if $\exists v \in D_i$ s.t. $(v, a) \in R_{ij}$ and $(v, b) \in R_{ik}$

A general scheme of the algorithm can be described by the following steps :

1. establish arc consistency
2. disconnect singleton values, singleton variables and doubleton variables (property1)
3. choose a variable and instantiate it
4. goto 1

We give below a more detailed description of the algorithm :

```
MAC+(in var) return boolean
begin
    dVar← ∅
    consistent ← AC(var) and DisconnectVar(var, dVar)
    if(consistent and Solve(var\dVar)) return true
    return false
end

Solve(in var) return boolean
begin
    if(var = ∅) return true
    else
      x_i ← ChooseVar(var)
      while(D_i ≠ ∅)
        v_i ← ChooseVal(D_i)
        Instantiate(x_i, v_i)
        consistent ← AC(var) and DisconnectVar(var\{x_i}, dVar)
        if(consistent and Solve(var\{x_i∪ dVar}))
          return true
        ConnectVar(var, dVar)
        UpdateAC(x_i, var)
        D_i ← D_i\{v_i}
      endwhile
      UpdateCurrentLevel(x_i)
      return false
    endif
end
```

var(resp. *dVar*) represents the set of uninstantiated variables (resp. disconnected variables).

We briefly describe the functions used by M*AC*+ :

- *DisconnetVar* disconnects all singleton and doubleton variables. This function can detect an inconsistency at the time of verification of a constraint by directed path-consistency.
- *ConnectVar* reconnects all disconnected variables by the previous function
- *AC* maintains an arc consistent CSP

- *UpdateAC* restores all modifications made by AC
- *UpdateCurrentLevel*: restores the variable domain when a backtrack occurs

For the sake of clarity, we chose to present an algorithm which treats the decision problem. If the problem is consistent, we can extend the partial solution to a complete one, in the same way as in MACE [21].

Let us mention that, in addition to singleton variables (used in MACE), M*AC*+ exploits doubleton variables using directed path consistency (property 1). Another important point, is that M*AC*+ can add at most one constraint for each disconnected doubleton. So, the number of added constraints is very limited.

4 SAC as local processing

Among several filtering techniques proposed over the last years, arc consistency is considered as the most useful in practice. For example, maintaining an arc consistent CSP during the search has led to the most efficient and popular constraints solving technique (MAC).

Recently, similar efficient local processing (based on unit-propagation) have been successfully used in solving the satisfiability problem. Indeed, most efficient implementations of the Davis, Logemann and Loveland procedure(DLL)[4] have been obtained thanks to a smart use of unit-propagation based techniques. The unit propagation process is used either to detect inconsistencies, or to deduce new literals and clauses. Let us, mention (among others), CSAT[9], tableau [3], POSIT[10] and Satz[15].

In the CSP framework, we believe that similar results could be obtained using arc consistency based propagation techniques such as singleton arc consistency [5]. Obviously, to reach this, we need to control the use of such propagation process.

We define $\mathcal{P} \downarrow_{\{(x_i,v)\}}$ (resp. $\mathcal{P} \uparrow_{\{(x_i,v)\}}$) as a CSP $\mathcal{P}$ with D_i reduced to a singleton value v (resp. as a CSP $\mathcal{P}$ where the value v is removed from D_i).

Definition 8 *Let $\mathcal{P} = (X, D, C, R)$ be a CSP, and $x_i \in X$,*

- *a value $v \in D_i$ is singleton arc consistent(SAC) iff $\mathcal{P} \downarrow_{\{(x_i,v)\}}$ has an arc consistent sub domain.*
- *a variable x_i is singleton arc consistent iff $\forall v \in D_i$ v is SAC*
- *a CSP $\mathcal{P}$ is singleton arc consistent, iff $\forall x_i \in X, D_i \neq \emptyset$, x_i is SAC*

In this section, we point out a possible way to exploit SAC propagation-based technique during the search. More precisely, if enforcing SAC on a given value leads to a domain wipe-out, such a value is definitely removed, otherwise a set of removed values is reported.

In the following, we show that, in addition to the pruning capabilities of SAC, some interesting information can be obtained from the propagation process and used:

- to produce additional domain reductions
- to avoid useless singleton arc consistency checks
- to refine the variable ordering heuristics

We give below some basic definitions.

Definition 9 *Let $\mathcal{P}$ and $\mathcal{P}'$ two CSP,*

- *$Sol(\mathcal{P})$ as the set of solutions of $\mathcal{P}$*
- *let $s \in Sol(\mathcal{P})$, $s(x_i)$ is the value assigned to x_i*
- *$\mathcal{P} \equiv \mathcal{P}'$ iff $Sol(\mathcal{P}) = Sol(\mathcal{P}')$*

Definition 10 *Let $\mathcal{P}$ be a CSP singleton arc consistent,*

- *$Red_{sac}(\mathcal{P}, x_i, v) = \{(x_j, w)$ s.t. $i \neq j$ and w is removed from D_j by applying SAC on $\mathcal{P} \downarrow_{\{(x_i,v)\}}\}$ denotes the set of domains reductions implied when enforcing SAC on the value $v \in D_i$.*
- *$Red_{sac}(\mathcal{P}, x_i) = \bigcap_{v \in D_i} Red_{sac}(\mathcal{P}, x_i, v)$ the set of removed values when enforcing SAC on the variable x_i.*
- *$Red_{sac}(\mathcal{P}) = \bigcup_{x_i \in X} Red_{sac}(\mathcal{P}, x_i)$*
- *$Sing_{sac}(\mathcal{P}, x_i, v) = \{(x_j, w)$ s.t. $i \neq j$ and $D_j = \{w\}\}$ is the set of singleton value obtained when enforcing SAC on the value $v \in D_i$.*

Remark 1 *$\forall (x_i, v) \in Red_{sac}(\mathcal{P})$, it is obvious that $\nexists s \in Sol(\mathcal{P})$ s.t. $s(x_i) = v$*

Let us now wonder whether more pruning can be obtained thanks to a fine analysis of the SAC propagation process.

Property 2 *Let $\mathcal{P}$ be a CSP singleton arc consistent, then $\mathcal{P} \equiv \mathcal{P} \uparrow_{Red_{sac}(\mathcal{P})}$*

Proof: Obviously, since $\mathcal{P} \uparrow_{Red_{sac}(\mathcal{P})}$ is obtained by removing from $\mathcal{P}$ inconsistent values, so $Sol(\mathcal{P} \uparrow_{Red_{sac}(\mathcal{P})}) = Sol(\mathcal{P})$ $\diamond$

The previous property tells us that, when enforcing SAC, additional values must definitely be removed from the domains of the CSP. So, with this additional analysis we can remove much more values than SAC.

In practice, enforcing SAC on a CSP $\mathcal{P}$ and computing $Red_{sac}(\mathcal{P})$ can be done simultaneously. Indeed, only SAC values for a given variable x_i are considered.

To get more benefits, we think that it is necessary to enforce SAC only on a manageable set of variables (selected using extra criteria, such as heuristics). In this respect, to avoid useless efforts when enforcing SAC, we give below a simple adaptation of a result obtained in SAT [13].

Property 3 *Let $\mathcal{P}$ be a CSP, $xi \in X$ and $v \in D_i$ such that v is SAC, $\forall (x_j, w) \in Sing_{sac}(\mathcal{P}, x_i, v)$, then w is SAC.*

Proof: Evident. $\diamond$

The cost of maintaining singleton arc consistent CSP can be improved in practice, thanks to the previous property. Indeed, when enforcing SAC on a given value, we deduce a set of SAC values. So, the number of singleton arc consistency checks is reduced.

We show now how to exploit SAC during the search. Let us start with some basic considerations. First, it is well known that the efficiency of CSP solving techniques is heavily related to the variable ordering heuristic. Secondly, we have noticed a great fluctuation in the behavior of CSP solving techniques on the same class of random CSP instances. In other words, on random CSPs, most of the best variable ordering heuristics (s.t. min(domain), min(domain)/max(degree), etc.) do not achieve a good discrimination between variables because of the uniform nature of the instances. So, the used ordering rule generally delivers a sub-set of variables.

Let H be an ordering variable heuristic:

- $\alpha = min_{x_i \in X}\{F(x_i)\}$, where $F(x_i)$ is the score of x_i (for example, $F(x_i) = |D_i|$ or $F(x_i) = \frac{|D_i|}{degre(x_i)}$),
- $\mathcal{L} = \{x_i \in X$ such that $F(x_i) = \alpha\}$ is a list of variables, all with the same score

Let $w(v, x_i)$ be the number of removed values when enforcing SAC on $\mathcal{P} \downarrow_{\{(x_i,v)\}}$. Obviously, the higher the value $w(v, x_i)$ is, the smaller the size of search space is, when x_i is instantiated to a value v.

To select a variable in $\mathcal{L}$, we use a second level heuristic H' defined as follow:

- $\alpha' = max_{x_i \in \mathcal{L}}\{F'(x_i)\}$, s.t. $F'(x_i) = \frac{\sum_{v \in D_i} w(v,x_i)}{|D_i|}$,
- $\mathcal{L}' = \{x_i \in \mathcal{L}$ s.t. $F'(x_i) = \alpha'\}$

For a given variable x_i, the function $F'(x_i)$ computes the average reduction on the size of the CSP. Consequently, the variable with a maximal value $F'(x_i)$, has more chance to maximize the reduction of the search tree. Moreover, we can modify the function $F'(x_i)$ in order to get a more balancing search tree (as done in SAT), for example, by using product operator instead of sum.

We briefly describe now a possible application of SAC as a local consistency technique:

1. apply the heuristic H on the uninstantiated variables
2. enforce SAC on the variables in $\mathcal{L}$
3. apply a second level ordering heuristic H'
4. randomly select a variable from $\mathcal{L}'$

In the second step (2), by removing some domains values, local inconsistencies could be detected.

Remark 2 *As in [12], the different $w(x, v)$ can also be used to define a domain value ordering.*

The results given in this section could be considered as the first step to the possible use of more powerful local consistency techniques. Also, they can be easily integrated in M*AC*+.

5 Preliminary experiments and discussion

We have implemented the M*AC*+ algorithm and conducted some preliminary experiment on two kinds of CSPs. The first class contains random CSPs generated using the standard generation model, the second class contains random CSPs with additional doubleton variables. We have compared M*AC*+ and MAC (M*AC*+ without doubleton variables) using the same ordering heuristic (min dom and min dom/deg). Let us relate some observations that are representative of the M*AC*+ behavior. We have observed that M*AC*+ is better on sparse CSPs. The improvements are more significant on unsatisfiable instances than on satisfiable ones. We believe that the efficiency of our algorithm might be more significant on real world problems. Indeed, real-life applications have special structure with non-uniform density. Furthermore, for inconsistent problems, the inconsistency is almost related to one of their subparts. In other words, an inconsistent problem can contain an easy part (e.g. tractable subproblem) and a hard one. Since most of the variable ordering heuristics use syntactic information such as domain size and/or variable degree, we are not safe from a wrong choice. In such a case, the disconnection of singleton and doubleton variables would be of prime importance, because it avoids repetitive solving of the hard part. To confirm this assumption, we have slightly modified the uniform random generator by adding extra doubleton variables. On such instances, the performance of M*AC*+ is significantly better.

6 Related works

The concept of hidden variables proposed by Rossi [19], gives a general way to determine redundant variables. A variable is redundant if its removal does not change the set of solutions. So, when a CSP is arc consistent, a singleton value (variable) can be considered as redundant. In [2], singleton values have been efficiently exploited in the MAC implementation, and singleton variables have been used in [18] and [21]. In addition to singleton value (variable), we have shown how to exploit doubleton variables during the search by using a partial form of path consistency.

7 Conclusion and perspectives

In this paper we have shown how filtering techniques such as path and singleton arc consistency can be exploited in order to reduce the search space. To conclude, let us summarize and motivate them briefly again.

- In the first part we have presented an interesting way to exploit a partial form of path consistency during the search. Using singleton and doubleton variable properties, M*AC*+ algorithm disconnects some useless variables and consequently simplifies the problem and avoids repetitive search. Empirical results show that this approach is particularly useful on sparse and inconsistent CSPs. Significant improvements are obtained on instances generated using a modified version of the random model generator. This confirms our intuition that M*AC*+ might be more effective on CSPs modeling real-life applications.

- In the second part, we have proposed a way to exploit SAC as local treatment and also to improve variable ordering heuristics. Its application on a subset of variables verifying the same criterion allows us to reduce the treatment cost. Also, we have shown that it is possible to extend the filtering capabilities of SAC thanks to an analysis of the propagation process.

As a future work, we plan to use M*AC*+ to compute a compact representation of the set of solutions. In fact, every partial solution found by M*AC*+ represents a subset of solutions.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26-11:832–843, 1983.

[2] C. Bessire and J.-C. Rgin. Mac and combined heuristics: Two reasons to forsake fc (and cbj?) on hard problems. In *CP'96, LNCS 1118*, pages 61–75, 1996.

[3] J. M. Crawford and L. D. Auton. Experimental results on the crossover point in satisfiablity problems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93)*, pages 21–27, 1993.

[4] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Journal of the Association for Computing Machinery*, 5:394–397, 1962.

[5] R. Debruyne and C. Bessire. Some practicable filtering techniques for the constraint satisfaction problems. In *IJCAI'97*, pages 412–417, 1997.

[6] R. Dechter. Enhancement schemes for constraint-satisfaction problems: Backjumping, learning and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

[7] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1988.

[8] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[9] O. Dubois, P. André, Y. Boufkhad, and J. Carlier. Sat versus unsat. In D. Johnson and M. Trick, editors, *Second DIMACS Challenge, 1993*, DIMACS Series in Discrete Mathematics and Theorical Computer Science, American Mathematical Society, pages 415–436, 1996.

[10] J. W. Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, University of Pennsylvania, Department of Computer and Information Science, 1995.

[11] E. C. Freuder. A sufficient condition for backtrack-free search. *Journal of ACM*, 29(1):24–32, 1982.

[12] D. Frost and R. Dechter. Look-ahead value ordering for constraint satisfaction problems. In *IJCAI'95*, pages 572–578, 1995.

[13] B. B. J. Audemard and P. Siegel. La mthode d'avalanche aval: une mthode numerative pour sat. In *JNPC'99*, Lyon France, 1999.

[14] P. Jgou. Decomposition of domains based on the structure of finite constraints satisfaction problems. In *AAAI'93*, pages 731–736, 1993.

[15] C. M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 366–371, Nagoya (Japan), Aug. 1997.

[16] D. A. C. M.C. Cooper and P. G. Jeavons. Characterizing tractable constraints. *Artificial Intelligence*, 65:347–361, 1994.

[17] U. Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Inform. Sci*, 7:95–132, 1974.

[18] J. C. Rgin. *Developpement d'Outils Algorithmiques pour l'intelligence Artificielle: Application à la Chimie Organique*. PhD thesis, Université de Montpellier II, 1995.

[19] F. Rossi. Redundant hidden variables in finite domain constraint problems. *Constraint processing, Springer-Verlag, LNCS*, (923), 1995.

[20] D. Sabin and E. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *ECAI'94*, 1994.

[21] D. Sabin and E. C. Freuder. Understanding and improving the mac algorithm. In *CP'97*, 1997.

Tractable Cover Compilations

Parue dans les actes de « *The Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)* », Nagoya (Japon), août 1997, pages 122 à 127.

Tractable Cover Compilations*

Yacine Boufkhad[1], **Éric Grégoire**[2], **Pierre Marquis**[2], **Bertrand Mazure**[2], **Lakhdar Saïs**[2,3]

[1] LIP6
Université Paris 6
4, place Jussieu
F-75252 Paris Cedex 05, FRANCE
boufkhad@laforia.ibp.fr

[2] CRIL [3] IUT de Lens
Université d'Artois
Rue de l'Université, S.P. 16
F-62307 Lens Cedex, FRANCE
{gregoire,marquis,mazure,sais}@cril.univ-artois.fr

Abstract

Tractable covers are introduced as a new approach to equivalence-preserving compilation of propositional knowledge bases. First, a general framework is presented. Then, two specific cases are considered. In the first one, partial interpretations are used to shape the knowledge base into tractable formulas from several possible classes. In the second case, they are used to derive renamable Horn formulas. This last case is proved less space-consuming than prime implicants cover compilations for every knowledge base. Finally, experimental results show that the new approaches can prove efficient w.r.t. direct query answering and offer significant time and space savings w.r.t. prime implicants covers.

1 Introduction

Different approaches have been proposed to circumvent the intractability of propositional deduction. Some of them restrict the expressive power of the representation language to tractable classes, like the Horn, reverse Horn, binary, monotone, renamable Horn, q-Horn, nested clauses formulas [Dowling and Gallier, 1984; Lewis, 1978; Boros *et al.*, 1994; Knuth, 1990]. Unfortunately, such classes are not expressive enough for many applications. Contrastingly, *compilation approaches* apply to full propositional-logic knowledge bases (KBs for short). Thanks to an off-line pre-processing step, a KB Σ is compiled into a formula Σ^* so that on-line query answering can be performed tractably from Σ^*. Many approaches to compilation have been proposed so far, mainly [Reiter and De Kleer, 1987; Selman and Kautz, 1991; 1994; del Val, 1994; Dechter and Rish, 1994; Marquis, 1995; del Val, 1995; 1996; Marquis and Sadaoui, 1996; Schrag, 1996].

*This work has been supported in part by the Ganymède II project of the Contrat État/Région Nord–Pas-de-Calais.

In this paper, a new approach to equivalence-preserving compilation, called *tractable covers*, is introduced. In short, a tractable cover of Σ is a finite set $\mathcal{T}$ of tractable formulas Φ (disjunctively considered) s.t. $\Sigma \equiv \mathcal{T}$. Tractable covers of Σ are equivalence-preserving compilations of Σ: a clause c is a logical consequence of Σ iff for every Φ in $\mathcal{T}$, c is a logical consequence of Φ. Since Φ is tractable, each elementary test $\Phi \models c$ can be computed in time polynomial in $|\Phi| + |c|$. The point is to find out tractable Φs that concisely represent (i.e. cover) the largest sets of models of Σ, so that $|\mathcal{T}|$ remains limited. To some extent, the present work could then be related to other model-based approaches to knowledge representation and reasoning, like [Khardon and Roth, 1996].

First, a general framework is presented, which can take advantage of most tractable classes, simultaneously. In many respects, it generalizes the prime implicants cover technique recently used for compilation purpose [Schrag, 1996]. Then, the focus is laid on tractable covers that can be computed and intensionally represented thanks to (partial) interpretations. Two specific cases are considered. In the first one, partial interpretations are used to shape the KB into formulas from several possible classes. In the second one, they are used to derive renamable Horn formulas. The last one is proved less space-consuming than prime implicants covers [Schrag, 1996] for every KB. Since tractable covers of Σ are equivalence-preserving compilations, their size may remain exponential in $|\Sigma|$ unless $NP \subseteq P/poly$ [Selman and Kautz, 1994], which is very unlikely. However, experimental results show that the new approaches can prove efficient w.r.t. direct query answering and offer significant time and space savings w.r.t. prime implicants covers.

2 Formal Preliminaries

A literal is a propositional variable or a negated one. A clause (resp. a term) is a finite set of literals, representing their disjunction (resp. conjunction). A Horn (resp. reverse Horn) clause contains at most one literal that is

positive (resp. negative). A binary clause contains at most two literals. A KB Σ is a finite set of propositional formulas, conjunctively considered. $Var(\Sigma)$ is the set of variables occurring in Σ. Every KB can be rewritten into a conjunction of clauses (into CNF for short) while preserving equivalence. A KB Σ is said renamable Horn iff there exists a substitution σ over literals l built up from $Var(\Sigma)$, s.t. $\sigma(l) = \overline{l}$ and $\sigma(\Sigma)$ is Horn.

Interpretations and models are defined in the usual way. Let us stress that, quite unconventionally, (partial) interpretations will be represented as terms, i.e. as satisfiable sets of literals (vs. sets of variables). An implicant of a KB Σ is a partial interpretation α s.t. $\alpha \models \Sigma$. A prime implicant of Σ is an implicant π of Σ s.t. for all implicants α of Σ s.t. $\pi \models \alpha$, we have $\alpha \models \pi$. The set of all prime implicants of Σ (up to logical equivalence) is noted $PI(\Sigma)$. A prime implicant cover of Σ is any subset S of $PI(\Sigma)$ (disjunctively considered) s.t. $S \equiv \Sigma$.

3 Tractable Cover Compilations

3.1 The General Framework

First, let us make precise what classes of tractable formulas will be considered:

Definition 3.1 *A list Cs of* classes C of tractable propositional formulas *(w.r.t. cover compilation) is s.t.:*

- *There exists a polytime algorithm $TRACTABLE?$ for checking whether any formula Φ belongs to C.*
- *There exists a polytime algorithm $QUERY?$ for checking whether $\Phi \models c$ holds for any Φ in C and any clause c.*
- *For every C in Cs, for every formula Φ of C and every term p, there exists C' in Cs s.t. $(\Phi \wedge p) \in C'$.*

Since classes of tractable formulas are not finite sets in the general case, they are intensionally represented by ordered pairs of decision procedures $\langle TRACTABLE?, QUERY? \rangle$.

Interestingly, the great majority of classes of formulas tractable for SAT (the well-known propositional satisfiability decision problem) are also tractable for cover compilations. Especially, this is the case for the Horn, reverse Horn, binary, renamable Horn, q-Horn and nested clauses classes.

We are now ready to define the notion of a tractable cover of a propositional KB.

Definition 3.2 *Given a CNF-KB Σ and a finite list Cs of classes of tractable formulas w.r.t. cover compilation (represented intensionally), a* tractable cover *of Σ w.r.t. Cs is a finite set $\mathcal{T}$ of satisfiable formulas (disjunctively considered) s.t.:*

- *For every $\Phi_i \in \mathcal{T}$, there exists C in Cs s.t. Φ_i belongs to C, and*
- $\mathcal{T} \equiv \Sigma$.

In the following, we will assume that Cs contains at least the class $\{t \text{ s.t. } t \text{ is a term}\}$. This ensures that there always exists at least one tractable cover of Σ, namely a prime implicants one.

In this paper, we focus on tractable covers that can be *intensionally represented*, using (partial) interpretations. The corresponding explicit covers can be generated on-line from the intensional ones in polynomial time.

3.2 Carver-Based Tractable Covers

A first way to derive covers consists in *simplifying* the KB using partial interpretations. For any partial interpretation $p = \{l_1, \ldots, l_n\}$, the KB Σ simplified w.r.t. p is the set $\Sigma_p = (((\Sigma_{l_1})_{l_2}) \ldots)_{l_n}$, where $\Sigma_l = \{c \setminus \{\neg l\} \text{ s.t. } c \in \Sigma \text{ and } c \cap \{l\} = \emptyset\}$.

Definition 3.3 *Given a CNF-KB Σ and a finite list Cs of classes of tractable formulas (represented intensionally):*

- *A* carver *of Σ w.r.t. Cs is a partial interpretation p s.t. there exists C in Cs s.t. Σ_p belongs to C and Σ_p is satisfiable.*
- *A* carver-based tractable cover *of Σ w.r.t. Cs is a set PC of carvers of Σ w.r.t. Cs (disjunctively considered) s.t. $(PC \wedge \Sigma) \equiv \Sigma$.*
- *A* carver-based cover compilation *of Σ is a pair $\mathcal{C} = \langle \Sigma, PC \rangle$, where PC is a carver-based tractable cover of Σ w.r.t. Cs.*

Interestingly, the space required to store a carver p is always lower or equal to the space needed by the corresponding tractable formula $\Phi = p \wedge \Sigma_p$ (with a $O(|\Sigma|)$ factor in the worst case). At the on-line query answering stage, the price to be paid is a $O(|\Sigma \wedge p|)$ time complexity overhead per carver p but this does not question the tractability of query answering.

As an example, let $\Sigma = \{\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4, x_1 \vee x_2 \vee x_4\}$ and let Cs contain the Horn and the binary classes. The set $PC = \{\{x_1\}, \{\bar{x}_1\}\}$ is a carver-based tractable cover of Σ w.r.t. Cs since $\Sigma_{\{x_1\}} = \{\bar{x}_2 \vee \bar{x}_3 \vee x_4\}$ is Horn and $\Sigma_{\{\bar{x}_1\}} = \{x_2 \vee x_4\}$ is binary. If Cs reduces to the renamable Horn class, $\{\emptyset\}$ is a carver-based tractable cover of Σ w.r.t. Cs since Σ belongs to the renamable Horn class.

Clearly enough, carver-based cover compilations are equivalence-preserving compilations:

Proposition 3.1 *Let Σ be a CNF-KB, $\mathcal{C} = \langle \Sigma, PC \rangle$ be a carver-based cover compilation of Σ and let c be a clause. $\Sigma \models c$ iff for every p in PC, $((p \models c)$ or $(\Sigma_p \models c))$, which can be decided in time polynomial in $|\mathcal{C}| + |c|$.*

Although the class(es) to which Σ_p belongs can be polynomially recognized on-line for each carver p of PC, in practice, they are determined once only at the off-line compiling stage. Accordingly, each carver p found during the compiling process is indexed with a reference $label(p)$ to the concerned class in Cs. A significant amount of time in on-line query answering can be saved via such indexing.

Interestingly, carver-based cover compilations can lead to exponential space savings w.r.t. prime implicants

cover compilations for some KBs. An extreme case consists of tractable KBs Σ that remain invariant under carver-based cover compilation but are such that every prime implicant cover is exponential in the size of Σ.

3.3 Hyper-Implicant Covers

Formulas in covers must be tractable and exhibit as many models of Σ as possible. Interestingly, several classes C of tractable formulas admit model-theoretic characterizations and features that can be exploited. For the binary, Horn, reverse Horn, renamable Horn classes, among others, class membership is equivalent to the existence of some specific interpretation(s). Formally, a CNF-KB Σ is renamable Horn [Lewis, 1978] iff there exists an interpretation p over $Var(\Sigma)$ s.t. for every clause c of Σ, at most one literal of c does not belong to p. Interestingly, the renamable Horn class includes both the Horn, reverse Horn, and satisfiable binary KBs as proper subsets.

As the semantical characterizations above indicate it, literals that belong both to p and to Σ play a central role that we can take advantage of. In order to derive satisfiable renamable Horn formulas, every clause c of Σ is shortened using a model $p = m$ of Σ: only the literals occurring in m are kept, plus an additional literal of c (when possible). In this way, every shortened clause is such that every literal of m (except possibly one) belongs to it. Accordingly, the resulting set of clauses, called *hyper-implicant* of Σ, is satisfiable and renamable Horn.

Formally, let m be a model of Σ. For every clause c in Σ, let c^m be the clause consisting of the literals common to c and m. Let $P(\Sigma, m)$ denote the set of clauses of Σ s.t. for every clause c in $P(\Sigma, m)$, c and c^m are identical. Let $N(\Sigma, m)$ be $\Sigma \setminus P(\Sigma, m)$. For every clause c in $N(\Sigma, m)$, let l_c^m denote a literal of c s.t. $\overline{l_c^m}$ belongs to m. Obviously, several candidates l_c^m may exist in the general case.

Definition 3.4 *Given a CNF-KB Σ:*

- *A* hyper-implicant *of Σ (w.r.t. to a model) m of Σ is a formula $\Sigma^m = (\bigwedge_{c \in P(\Sigma,m)} c) \wedge (\bigwedge_{c \in N(\Sigma,m)} (c^m \vee l_c^m))$.*
- *A* hyper-implicant cover *$\mathcal{H}$ of Σ is a set of hyper-implicants of Σ (disjunctively considered) s.t. $\mathcal{H} \equiv \Sigma$.*

Importantly, the size of any hyper-implicant Σ^m of Σ is strictly lower than the size of Σ, except when Σ is renamable Horn.

Hyper-implicant covers are equivalence-preserving compilations:

Proposition 3.2 *Let Σ be a CNF-KB, $\mathcal{H}$ be a hyper-implicant cover of Σ and c be a clause. $\Sigma \models c$ iff for every hyper-implicant Σ^m of $\mathcal{H}$, we have $\Sigma^m \models c$. This can be decided in time linear in $|\mathcal{H}| + |c|$.*

Assuming that the clauses in $\Sigma = \{c_1, \ldots, c_n\}$ are totally ordered, hyper-implicant covers can be represented intensionally by Σ and sets of pairs $\langle m, [l_{c_1}^m, \ldots, l_{c_n}^m] \rangle$ where m is a model of Σ and $l_{c_i}^m$ ($i \in 1..n$) is $false$ if $c_i \in P(\Sigma, m)$ and is the literal of $c_i \setminus m$ which is kept, otherwise. Each Σ^m can be derived on-line from Σ and $\langle m, [l_{c_1}^m, \ldots, l_{c_n}^m] \rangle$ in linear time.

As an example, let $\Sigma = \{x_1 \vee x_3 \vee x_4, x_1 \vee \bar{x}_2 \vee x_3, x_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4, \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4, \bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee x_4\}$. The hyper implicant cover represented by Σ and the two pairs $\langle \{x_1, \bar{x}_2, \bar{x}_3, x_4\}, [x_3, x_3, x_2, \bar{x}_1, \bar{x}_1] \rangle$ and $\langle \{\bar{x}_1, x_2, x_3, \bar{x}_4\}, [x_1, \bar{x}_2, \bar{x}_3, \bar{x}_2, x_4] \rangle$, contains two hyper-implicants: $\{x_1 \vee x_4 \vee x_3, x_1 \vee \bar{x}_2 \vee x_3, x_1 \vee \bar{x}_3 \vee x_2, \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_1, \bar{x}_3 \vee x_4 \vee \bar{x}_1\}$ and $\{x_3 \vee x_1, x_3 \vee \bar{x}_2, x_2 \vee \bar{x}_4 \vee \bar{x}_3, \bar{x}_1 \vee \bar{x}_4 \vee \bar{x}_2, \bar{x}_1 \vee x_2 \vee x_4\}$.

Intuitively, any hyper-implicant Σ^m of Σ is a concise representation of some prime implicants of Σ. To be more specific, the prime implicants of Σ which are entailed by m are prime implicants of Σ^m.

Proposition 3.3 *For every model m of Σ, let $PI_m(\Sigma)$ be the set of prime implicants of Σ s.t. for every π in $PI_m(\Sigma)$, we have $m \models \pi$. Then, $PI_m(\Sigma) \subseteq PI(\Sigma^m)$.*

Consequently, Σ^m covers every model of Σ covered by a prime implicant of Σ entailed by m.

Hyper-implicants covers are economical representations w.r.t. prime implicants ones. Not withstanding the fact that hyper-implicants of Σ are smaller than Σ, the number of hyper-implicants in a cover is lower than the number of implicants in a cover:

Proposition 3.4 *For every prime implicant cover of Σ containing t prime implicants there exists a hyper-implicant cover of Σ containing at most $\lfloor \frac{t+1}{2} \rfloor$ hyper-implicants.*

In the example above, the hyper-implicant cover contains 2 formulas, while the smallest prime implicant cover of Σ consists of 6 prime implicants. Actually, experiments show significant savings w.r.t. the sizes of the prime implicants covers for most KBs (cf. Section 5).

4 Computing Compilations

(Partial) interpretations giving rise to intensionally-represented tractable covers are computed using systematic search, thanks to a Davis/Putnam-like procedure `DPTC`. This procedure is closely related to Schrag's `DPPI` algorithm [Schrag, 1996]. `Cs` is empty for the hyper-implicant case.

```
DPTC(Σ,Cs):
1   PC ← ∅;
2   DP*(Σ,Cs,∅);
3   Return(⟨Σ,PC⟩).

DP*(Σ,Cs,p):
1   If not PRUNING(Σ,p) then
2      UNIT_PROPAGATE(Σ,p);
3      If ∅ ∈ Σ then return;
4      If (p ⊨ Σ)
```

```
5          then PROCESS_IMPLICANT(p,Σ,Cs)
6               return;
7       l← CHOOSE_BEST_LITERAL(Σ);
8       DP*(Σ,Cs,p∪{l});
9       DP*(Σ,Cs,p∪{¬l}).
```

The CHOOSE_BEST_LITERAL branching rule and UNIT_PROPAGATE procedures are standard Davis/Putnam features. In our experiments, the branching rule by [Dubois *et al.*, 1996] is used. The main role of DP* is to find implicants of Σ in the whole search tree. PRUNING and PROCESS_IMPLICANT depend on the considered approach. From the found implicants, (partial) interpretations (carvers and implicit representations of hyper-implicants) are derived thanks to PROCESS_IMPLICANT; they are collected into the global variable PC.

4.1 The Carver-Based Case

```
PRUNING_C(Σ,new_p):
1   If there exists p ∈ PC s.t. (new_p ⊨ p)
2      then return(true) else return(false).

PROCESS_IMPLICANT_C(new_p,Σ,Cs):
1   Carver ← DERIVE_C(new_p,Σ,Cs);
2   For every p ∈ PC do
3       If (p ⊨ Carver) then remove p from PC;
4   Put Carver in PC.

DERIVE_C(p,Σ,Cs):
1   Prime ← ONE_PRIME(p,Σ);
2   For every literal l ∈ Prime do
3    For every ⟨TRACTABLE?,QUERY?⟩ in Cs do
4      If TRACTABLE?(Σ_{Prime\{l}})
         then Return(DERIVE_C(Prime\{l},Σ,Cs));
5   Return(Prime).
```

Whenever an implicant p of Σ is found, a prime implicant Prime of Σ s.t. $p \models$ Prime is extracted from p, thanks to the ONE_PRIME procedure. ONE_PRIME considers every literal l of p successively, check whether l is necessary (i.e. if there exists a clause c of Σ s.t. $c \cap p = \{l\}$), and remove l from p if l is not necessary (see [Castell and Cayrol, 1996][Schrag, 1996] for details). These prime implicants are then tentatively simplified; all the literals of each Prime are considered successively; for every literal l of Prime, $\Sigma_{Prime\setminus\{l\}}$ is checked for tractability using the recognition procedures given in Cs. If $\Sigma_{Prime\setminus\{l\}}$ is found tractable, l is ruled out from Prime and is kept otherwise. Once all the literals of Prime have been considered, the resulting simplified Prime (indexed by the label of the corresponding class in Cs) is checked against the set PC of carvers collected so far. Only maximal carvers w.r.t. $\models$ are kept in PC. Interestingly, the set PC of carvers stored during the traversal of the DP search tree is used to prune this tree, thanks to the PRUNING_C procedure; indeed, whenever a candidate (partial) interpretation new_p is elected, it can be immediately removed if new_p entails one of the current carvers.

Clearly enough, both the literal ordering and the recognition procedure ordering used in DERIVE_C can greatly influence the cover generated in this way.

4.2 The Hyper-Implicant Case

```
PRUNING_H(Σ,new_p):
1   If there exists p ∈ PC s.t. (Σ^{new_p} ⊨ Σ^p)
2      then return(true) else return(false).

PROCESS_IMPLICANT_H(new_p,Σ,_):
1   Model ← DERIVE_H(new_p,Σ);
2   For every p ∈ PC do
3       If (Σ^{Model} ⊨ Σ^p) then remove p from PC;
4   Put Model in PC.

DERIVE_H(p,Σ):
1   Model ← p;
2   For every variable v ∈ Var(Σ) do
3       If ({v} ∩ p = ∅) and ({¬v} ∩ p = ∅)
4          then put v with its most frequent
                sign in Σ to Model;
5   Return(Model).
```

Each time an implicant p of Σ is found, a model Model of Σ s.t. Model $\models p$ is derived from p. The variables that do not occur in p are added with the sign occurring the most frequently in Σ. When Σ^{Model} is computed for the first time, a list $[l^m_{c_1}, \ldots, l^m_{c_n}]$ is attached to $Model$. The DERIVE_H procedure is both simpler and more efficient than DERIVE_C (roughly, it is not more time-consuming than the prime implicant extraction achieved by ONE_PRIME). The tractable formulas Σ^p corresponding to the models p collected in PC are used to prune the search tree (PRUNING_H). Only the ps giving rise to the logically weakests Σ^p are kept in PC. Since the renamable Horn formulas Σ^{Model} and Σ^p stem from the same KB Σ, checking whether ($\Sigma^{Model} \models \Sigma^p$) can be performed in a very efficient way.

Both the literals chosen to extend the found implicants to models p of Σ, and the literals l^m_c selected in clauses of Σ^p may have a significant impact on the hyper-implicants Σ^p that are generated.

5 Experimental Results

In contrast to SAT, only few benchmarks for knowledge compilation can be found in the literature (with the well-developed experimental framework of [Schrag, 1996] as an exception). Actually, no comprehensive analysis of what should be the nature of meaningful benchmarks for evaluating compilation approaches has ever been conducted. Clearly, benchmarks must be hard for query answering since the goal of knowledge compilation is to overcome its intractability. However, in contrast to [Schrag, 1996], we do not focus on hard SAT instances,

problems	#var	#cla	$\alpha_{\mathcal{P}}$ $\alpha_{\mathcal{C}}$ $\alpha_{\mathcal{H}}$	$\beta_{\mathcal{P}}$ $\beta_{\mathcal{C}}$ $\beta_{\mathcal{H}}$	$\lvert\mathcal{P}\rvert$ $\lvert\mathcal{C}\rvert$ $\lvert\mathcal{H}\rvert$
adder	21	50	5.08 2.09 0.20	— — 78.75	5376 1044 305
history-ex	21	17	2.00 0.75 0.14	— 1000.00 11.66	172 234 77
two-pipes	15	54	0.66 0.60 0.12	125.00 125.00 20.00	255 234 192
three-pipes	21	82	0.47 0.42 0.27	45.45 <1 17.50	441 373 284
four-pipes	27	110	0.36 0.25 0.20	23.80 18.51 29.16	675 528 380
regulator	21	106	0.51 0.29 0.17	26.31 20.83 29.99	861 530 422
selenoid	11	19	2.16 1.40 0.20	— — 17.49	297 115 94
valve	13	50	0.85 0.66 0.16	<1 <1 21	312 297 246

Table 1: Experimental results.

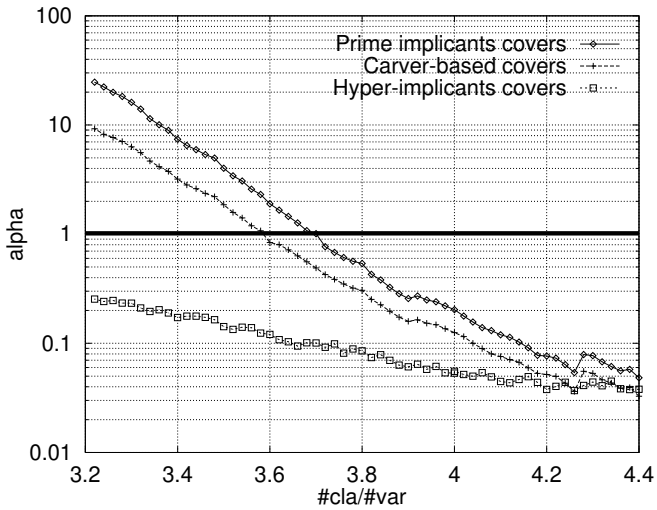

Figure 1: Ratios α.

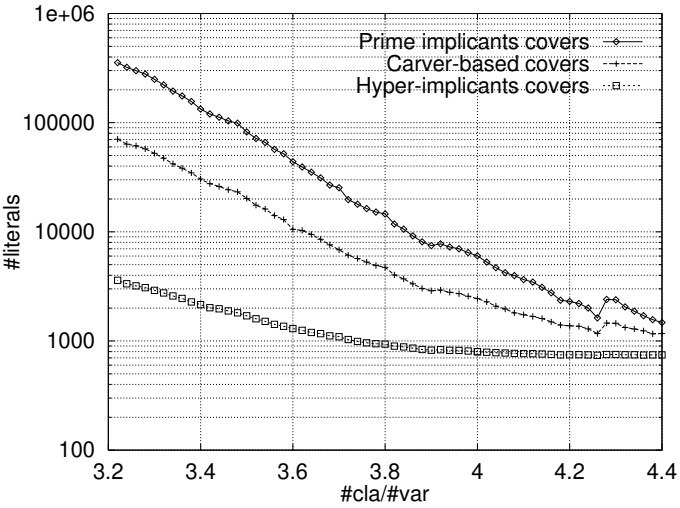

Figure 2: Sizes of the compilations.

only. Hard SAT instances (with respect to current algorithms) should be considered hard for query answering since at least one query (namely, the empty clause) is difficult. However, easy SAT instances can exhibit hard queries that differ from the empty clause.

Accordingly, we tested the tractable covers and the prime implicants approaches w.r.t. many KBs, including "standard" structured problems, taken from [Forbus and De Kleer, 1993], and random k-SAT problems [Dubois *et al.*, 1996], varying the $\#cla(use)/\#var(iable)$ ratio from the easy to the hard regions. Each KB Σ has been compiled using the 3 techniques. Then, 500 queries have been considered. In order to check the usefulness of the compilation process, we also answered these queries from Σ, using a direct, uncompiled, Davis/Putnam-based approach [Dubois *et al.*, 1996]. For each problem Σ and each compilation technique, the ratios $\alpha = Q_C/Q_U$ and $\beta = C/(Q_U - Q_C)$ have been computed. Q_C (resp. Q_U) is the time needed by the compiled (resp. uncompiled) approach to answer all the queries, and C is the compilation time. α (resp. β) tells us how much query time improvement we get from compilation (resp. how many queries are required to amortize the cost of compilation).

Table 1 reports some results of our extensive experiments. For each problem [Forbus and De Kleer, 1993], it lists results for the prime implicants, carvers, and hyper-implicants covers, successively. Especially, it gives the ratios α and β and the size (in literals) of the corresponding cover. The size of any tractable cover compilation is the size of Σ plus the size of the set of (partial) interpretations used as an implicit representation. For the carver-based approach, only the Horn, reverse Horn and binary classes have been considered. For the hyper-implicant approach, simplification of the cover (i.e. lines 2 to 4 of the `PROCESS_IMPLICANT`$_{\mathcal{H}}$ procedure) has not been implemented.

Results obtained on 50 variables random 3-SAT problems, where the ratio $\#cla/\#var$ varies from 3.2 to 4.4, are reported on the two next figures. 50 problems have

been considered per point and the corresponding scores averaged. Figure 1 (resp. Figure 2) gives aggregate values of ratios α (resp. sizes in literals) obtained for each compilation technique. $\alpha = 1$ separates the region for which compilation is useful from the region for which it is not.

At the light of our experiments, tractable covers prove better than prime implicants covers, both for structured and random k-SAT problems. Significant time savings w.r.t. query answering and significant space savings are obtained. Especially, tractable covers prove useful for many KBs for which prime implicants covers are too large to offer improvements w.r.t. query answering. More, the tractable cover approach allows the compilation of KBs which have so huge prime implicants covers $\mathcal{P}$ that $\mathcal{P}$ cannot be computed and stored. This coheres with the theoretical results reported in [Boufkhad and Dubois, 1996], showing that the average number of prime implicants of k-SAT formulas is exponential in their number of variables.

6 Conclusion

Both theoretical and experimental results show the tractable cover approach promising and encourage us to extend it in several directions. A first issue for further research is how to determine efficiently the best suited classes of tractable formulas for a given KB Σ. On the experimental side, an extensive evaluation of the carver-based technique equipped with more expressive tractable classes must be done. Extending the hyper-implicant approach to other tractable classes, especially the q-Horn one [Boros *et al.*, 1994], is another interesting perspective. Finally, fragments of tractable covers of Σ can serve as approximate compilations (lower bounds) of Σ in the sense of [Selman and Kautz, 1991; 1994; del Val, 1995; 1996]. Since the tractable cover approach allows disjunctions of tractable formulas from several classes, better approximations could be obtained.

References

[Boros *et al.*, 1994] E. Boros, P.L. Hammer, and X. Sun. Recognition of q-horn formulae in linear time. *Discrete Applied Mathematics*, 55(1):1–13, 1994.

[Boufkhad and Dubois, 1996] Y. Boufkhad and O. Dubois. Length of prime implicants and number of solutions of random r-cnf formulas. *(submitted)*, 1996.

[Castell and Cayrol, 1996] T. Castell and M. Cayrol. Computation of prime implicates and prime implicants by the davis and putnam procedure. In *Proc. ECAI'96 Workshop on Advances in Propositional Deduction*, pages 61–64, Budapest, 1996.

[Dechter and Rish, 1994] R. Dechter and I. Rish. Directional resolution: The davis-putnam procedure, revisited. In *Proc. KR'94*, pages 134–145, Bonn, 1994.

[del Val, 1994] A. del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In *Proc. KR'94*, pages 551–561, 1994.

[del Val, 1995] A. del Val. An analysis of approximate knowledge compilation. In *Proc. IJCAI'95*, pages 830–836, Montreal, 1995.

[del Val, 1996] A. del Val. Approximate knowledge compilation: The first-order case. In *Proc. AAAI'96*, pages 498–503, Portland (OR), 1996.

[Dowling and Gallier, 1984] W. Dowling and J. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.

[Dubois *et al.*, 1996] O. Dubois, P. Andre, Y. Boufkhad, and J. Carlier. Sat vs. unsat. In 2^{nd} *DIMACS Implementation Challenge*, volume 26 of *DIMACS Series*, pages 415–436. American Mathematical Society, 1996.

[Forbus and De Kleer, 1993] K.D. Forbus and J. De - Kleer. *Building Problem Solvers.* MIT Press, 1993.

[Khardon and Roth, 1996] R. Khardon and D. Roth. Reasoning with models. *Artificial Intelligence*, 87:187–213, 1996.

[Knuth, 1990] D.E. Knuth. Nested satisfiability. *Acta Informatica*, 28:1–6, 1990.

[Lewis, 1978] H.R. Lewis. Renaming a set of clauses as a horn set. *JACM*, 25:134–135, 1978.

[Marquis and Sadaoui, 1996] P. Marquis and S. Sadaoui. A new algorithm for computing theory prime implicates compilations. In *Proc. AAAI'96*, pages 504–509, Portland (OR), 1996.

[Marquis, 1995] P. Marquis. Knowledge compilation using theory prime implicates. In *Proc. IJCAI'95*, pages 837–843, Montreal, 1995.

[Reiter and De Kleer, 1987] R. Reiter and J. De Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *Proc. AAAI'87*, pages 183–188, Seattle (WA), 1987.

[Schrag, 1996] R. Schrag. Compilation for critically constrained knowledge bases. In *Proc. AAAI'96*, pages 510–515, Portland (OR), 1996.

[Selman and Kautz, 1991] B. Selman and H. Kautz. Knowledge compilation using horn approximations. In *Proc. AAAI'91*, pages 904–909, 1991.

[Selman and Kautz, 1994] B. Selman and H. Kautz. Knowledge compilation and theory approximation. *JACM*, 43(2):193–224, 1994.

Iterated syntax-based revision in a nonmonotonic setting

À paraître dans « *Frontiers in Belief Revision, Kluwer Academic Publishers* », M.-A. WILLIAMS et H. ROTT éditeurs, 2000, sous presse.

B. BESSANT É. GRÉGOIRE P. MARQUIS L. SAÏS

ITERATED SYNTAX-BASED REVISION IN A NONMONOTONIC SETTING

ABSTRACT: In this paper, a new syntax-based approach to iterated belief base revision is presented. It is developed within a nonmonotonic framework that allows a two-steps handling of inconsistency to be adopted. First, a disciplined use of nonmonotonic ingredients is made available to the knowledge engineer to prevent many inconsistencies that would occur if a standard logical interpretation and representation of beliefs were conducted. Remaining inconsistencies are considered unexpected and revised by weakening the formulas occurring in any minimally inconsistent subbase, as if they were representing exceptional cases that do not actually occur. While the computation of revised knowledge bases remains intractable in the worst case, our approach benefits from an efficient local search-based heuristic technique that empirically proves often viable, even in the context of very large propositional applications.

1 INTRODUCTION

Various approaches to belief revision have been proposed this last decade. Most of them have been investigated from a knowledge-level theoretical perspective [32], mainly. Despite a few noticeable exceptions (e.g. [34] [29] [11] [20]), their computational counterparts have rarely been considered. Worse, due to worst case complexity results, they are often believed to be intractable when large-scale applications are addressed. However, recent impressive empirical progress in propositional reasoning and search [36] does open real computational perspectives for syntax-based approaches to belief revision with respect to large applications. Also, many efforts have been devoted to establish formal connections between nonmonotonic logics and belief revision (see e.g. [9] [21] [25] [19]). In this paper, a knowlege[1] engineering-motivated approach to belief revision is adopted, showing that belief revision and nonmonotonic logics are not only two sides of the same coin [14] but that they can play useful synergetic roles in a unified framework that proves empirically computationally viable. More precisely, a two-steps policy to syntax-based revision of inconsistent beliefs is proposed. On the one hand, nonmonotonic ingredients are provided within a belief representation language giving rise to a gain of expressiveness, allowing a careful knowledge engineer to avoid many future inconsistencies that would occur if a standard deductive representation and inference of beliefs were selected. Inconsistencies that are not forecast and prevented in this way are then the object of a syntax-based belief revision process. In this respect, we can envision a whole family of syntax-based revision approaches: from a *full-meet* cautious one that weakens each formula in each minimally inconsistent subbase to regain consistency, to a *maxichoice* change policy that

[1] In this paper, *knowledge* is used for *belief*.

just weakens one smallest subset of formulas to restore consistency. In this paper, an empirically viable computational technique for the first approach is proposed. It makes use of a specific heuristic in the trace of local search techniques to check (in)consistency. Interestingly enough, it is shown that the nonmonotonic ingredients do not significantly affect the computational complexity results with respect to the revision process, neither from a worst case point of view, nor from an empirical one.

The paper is organized as follows. After some formal preliminaries, a specific nonmonotonic representation setting is described. Then, a new syntax-based approach to iterated belief base revision is introduced and motivated in this framework, together with a preferred-models inference relation for the resulting logic. The complexity and rationality issues for this cautious, full-meet approach to base revision are analyzed. Local search techniques for SAT are then specialized to match our specific needs. Typical results from extensive tests are reported. Finally, the specific position adopted in this paper with respect to the various facets of belief revision is summarized, and perspectives for further research are briefly exposed. Basic elements of computational complexity can be found in an appendix.

2 SOME FORMAL PRELIMINARIES

Let PS denote a denumerable set of *propositional symbols* (also called *variables* or *atoms*). For every subset V of PS, $PROP_V$ denotes the *propositional language* built up from the symbols of PS, the boolean constants $true$ and $false$ and the connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ in the standard way. The elements of $PROP_{PS}$ are called *formulas*.

Every propositional symbol of PS is also called a *positive literal* and a negated one a *negative literal*. The boolean constants are also viewed as positive literals. A *literal* is either a positive literal or a negative literal. Every finite disjunction of literals is called a *clause*. $Var(KB)$ denote the set of propositional variables occurring in the formula KB. For every propositional symbol v from PS, $KB_{v \leftarrow true}$ (resp. $KB_{v \leftarrow false}$) denotes the formula KB in which every occurrence of v is replaced by $true$ (resp. $false$).

An *interpretation* I (over $PROP_{PS}$) is a mapping that associates every propositional symbol to one of the two truth values of $BOOL = \{0, 1\}$ and $I(true) = 1$ and $I(false) = 0$. $[\![KB]\!](I)$ denotes the truth value taken by KB within I; it is defined in the usual compositional way. When I is s.t. $[\![KB]\!](I) = 1$, I is said to be a *model* of KB, noted $I \models KB$; otherwise, I is a *counter-model* of KB. The set of all models of KB is noted $\mathcal{M}(KB)$. When a formula has a model, it is said *consistent*. When a formula has no model (resp. no counter-model), it is said *inconsistent* (resp. *valid*). The binary relation $\models$ over $PROP_{PS}$ is defined by $KB \models f$ iff $\mathcal{M}(KB) \subseteq \mathcal{M}(f)$ holds. The *deductive closure* of KB is the set of all formulas that are logical

consequences of KB. Whenever $KB \models f$ and $f \models KB$ both hold, KB and f are said *logically equivalent*, noted $KB \equiv f$.

Every set of formulas is interpreted conjunctively; especially, when it is finite, a set of formulas is also considered as the formula that is the conjunction of its elements. It is well-known that every formula admits a conjunction of clauses equivalent to it; such a conjunction of clauses is called a *conjunctive normal form* (CNF for short) of the formula.

3 A NONMONOTONIC REPRESENTATION FRAMEWORK

This paper is more concerned with the knowledge engineering branch of artificial intelligence than with its cognitive and philosophical sides; building effective practical systems is here the main goal. In this respect, it is generally agreed that some trade-off can be required between the expressiveness of the representation language and the inference efficiency. Similarly, a specific combination of belief revision and nonmonotonic representation of beliefs and inference can be beneficial, especially when software quality properties are required.

Accordingly, a two-steps handling of inconsistent beliefs is advocated in this paper. First, the knowledge engineer is provided with a nonmonotonic language allowing him to prevent many future inconsistencies that would occur if a purely standard deductive logic were used. This language should be expressive enough to represent (prioritized) rules of default reasoning and exceptions, preventing these last ones from leading to inconsistency when they actually occur. In this respect, it is claimed here that when real-world large-scale applications are undertaken, the knowledge engineer can be required a disciplined use of nonmonotonic ingredients that are restricted, in exchange for the hope for predictable system behavior and good computational performance in many cases. However, the knowledge engineer cannot be required to forecast all such exceptions in this way. This is exactly where belief revision has to play a role in our framework. Belief revision will thus here just concern a mere remainder of the inconsistencies that would occur if a purely deductive logic were selected. This point will allow specific revision policies to be justified.

In the following, any belief base KB will be represented as a finite set (interpreted conjunctively) of propositional formulas from $PROP_{PS}$. The set of propositional symbols PS is partitioned into three denumerable sets, P, AB_{KB} and $AB_{Revision}$. P is used to represent the core knowledge of KB, while the two disjoint sets AB_{KB} and $AB_{Revision}$ contain McCarthy's prioritized Abnormality propositions, noted respectively by Ab_i and Ab_{R_i} [24]. Elements from AB_{KB} allow prioritized rules of default reasoning to be expressed together with exceptions (they encode structural abnormalities) while $AB_{Revision}$ is actually a sequence and is devoted to the revision policy.

Typically, a default rule expressing that a bird can fly (unless exceptions) will be of the form $\neg Bird \vee Ab_{flying_bird} \vee Fly$, representing the formula $Bird \wedge \neg Ab_{flying_bird} \Rightarrow Fly$. Ab_i propositions are intended to allow for exceptions without the need of expressing them explicitly. Such a framework will be interpreted under model-preference semantics *à la* Shoham [39]: $KB \mid\sim_{\leq} f$ iff $[\![f]\!](I) = 1$ in all preferred models I of KB, where the preferred models of KB are those that are minimal w.r.t. a given pre-order $\leq$. Intuitively, models where Ab_i propositions are minimized (i.e. are interpreted as 0) will be preferred. Many minimization schemata (especially prioritizing ones) giving rise to preferred-models inference relations $\mid\sim_{\leq}$ can be adopted in this context. The approaches of revision that will be developed in this paper are compatible with all of them.

4 BASE REVISION BY WEAKENING: THE FULL-MEET APPROACH

As the last example illustrates it, the knowledge engineer may not forecast possible exceptions to all rules (assume that he has asserted that all birds fly and that now we are given a penguin). When inconsistency does occur, several possible approaches to restore consistency can be devised. As we adopt here a knowledge engineering approach to artificial intelligence, we claim that the syntax of the encoded beliefs is important and that this should appear in the revision process. Accordingly, in our approach, the epistemic state of an agent (i.e. the set of all its beliefs) is represented as a belief base, i.e. a finite set of propositional formulas: the pieces of knowledge that can be inferred from such a set are the agent's beliefs. While such an approach to belief revision is syntax-sensitive (at least to the extent that it cannot be guaranteed that two equivalent KBs will always be revised in the same way), it is representationally feasible; this contrasts with the belief set approach and the model-based approach to belief revision: due to space considerations, representing the epistemic state of an agent by a deductively closed set of formulas, or by the corresponding set of models, is typically out of reach.

Our cautious approach to belief revision is based on the logical concept of weakening. Indeed, we believe that throwing out some beliefs in order to restore consistency is unnecessary destructive. Accordingly, we prefer "hidding" these beliefs by logically weakening them, allowing interesting perspectives with respect to iterated belief revision.

Intuitively, the way according to which beliefs are weakened amounts to considering these beliefs as exceptional ones that do not hold in regard of the newly introduced formula f.

Assume KB to be a nonmonotonic propositional belief base to be interpreted under some preferred-models semantics, given by a pre-order $\leq$

among interpretations. Initially, KB does not contain any occurrence of a symbol from $AB_{Revision}$. Let f be the so-called *revision formula*, i.e. a new piece of knowledge to be added into KB. A *severe revision* has to occur whenever $KB \cup \{f\}$ is inconsistent. The sequence of Abnormality propositions $AB_{Revision} = \{Ab_{R_1}, \ldots, Ab_{R_j}, \ldots\}_{(j<\omega)}$ is used to remove inconsistencies. More precisely, each time a severe revision is to be conducted, a subset of formulas in KB are weakened using the next available Ab_{R_j} in the sequence $AB_{Revision}$ as an additional disjunct in each of these clauses. Note that the elements of $AB_{Revision}$ are ordered in such a way that AB_{R_j} is after AB_{R_i} whenever $i < j$, so as to ensure that the most recently introduced abnormality symbol Ab_{R_j} occurs after Ab_{R_i} in $AB_{Revision}$.

Now, several revision policies can be defined depending on the way this set of formulas to be weakened is selected. Indeed, in order to break all inconsistencies, it is sufficient to weaken one formula in every of the possibly many subbases of KB that are inconsistent with f. Accordingly, several hitting sets of such subbases can be elected for being weakened in the general case. For the case where no additional information (e.g. an epistemic relevance ordering [28]) enables chosing one such hitting set, a *cautious* policy is proposed: in this situation, all the formulas in the set-theoretic union of the smallest subsets of KB inconsistent with f are weakened. But let us formalize this.

DEFINITION 1 (minimally inconsistent subbase). A *minimally inconsistent* subbase of KB w.r.t. f is any subset S of KB s.t. $S \cup \{f\}$ is inconsistent and no proper subset of it satisfies the same property.

Such subbases are sometimes called *entailment sets* for $\neg f$ (indeed, $S \cup \{f\}$ is inconsistent iff $\neg f$ is entailed by S).

DEFINITION 2 (kernel). The *kernel* $Ker(KB, f)$ of KB w.r.t. f is the set-theoretic union of all minimally inconsistent subbases of KB w.r.t. f

$Ker(KB, f)$ can also be defined in a dual way, focusing on the maximally consistent subbases of KB w.r.t. f (also called *remainders* of KB by $\neg f$). Indeed, $Ker(KB, f)$ is the set-theoretic complement in KB of the intersection of all such subbases (see Theorem 7.1 from [31]).

We are now ready to define our full-meet base revision operation:

DEFINITION 3 (full-meet revision). Assume that KB is the result of i previous severe revisions. $KB \circ_{full-meet} f = (Ker(KB, f) \vee Ab_{R_{i+1}}) \cup (KB \setminus Ker(KB, f)) \cup \{f\}$

In this definition, $Ker(KB, f) \vee Ab_{R_{i+1}}$ denotes the set of formulas $g \vee Ab_{R_{i+1}}$, where g is a formula from $Ker(KB, f)$. Clearly enough, the rank $i + 1$ of the current revision can be determined in time linear in the size of KB (it is sufficient to filter out the greatest i s.t. AB_{R_i} occurs in KB, and to increment it).

EXAMPLE 4. Let us illustrate the previous definition through a very simple example. Let $KB = \{\neg Bird \vee Fly, Bird\}$. This corresponds to the case where the knowledge engineer does not envision any exceptional bird, i.e. a bird that does not fly. We have:

1. $KB^{\circ_{full-meet}} \neg Fly = \{\neg Bird \vee Fly \vee Ab_{R_1}, Bird \vee Ab_{R_1}, \neg Fly\}$.
2. $(KB^{\circ_{full-meet}} \neg Fly)^{\circ_{full-meet}} Fly = \{\neg Bird \vee Fly \vee Ab_{R_1}, Bird \vee Ab_{R_1}, \neg Fly \vee Ab_{R_2}, Fly\}$.

In our approach, every formula from KB that belongs to the intersection of all maximally consistent subbases of KB w.r.t. f is kept in the revised base, while the remaining formulas are weakened. Thus, instead of focusing on the intersection of the *logical closures* of all (preferred[2]) maximally consistent subbases of KB w.r.t. f (as it is the case in many base revision approaches), we focus on the subbases themselves. This has two immediate, favourable consequences:

- Revised KBs can be efficiently represented in our approach (since the size of the revised KBs cannot exceed the size of the corresponding original KB plus the size of the revision formula plus the number of formulas occurring in the original KB).
- Revised KBs do not reduce to a single formula. Accordingly, iterated belief revision can be achieved without trivializing as soon as the first severe revision has been achieved.

These two valuable features are not shared by many syntax-based approaches to base revision. On the one hand, it is shown [5] that representations of revised KBs that are compact, i.e. of size polynomially bounded by the size of the original KB plus the size of the revision formula, are very unlikely to exist in the general case for various revision policies. Especially, while there can be exponentially many maximally consistent subbases of KB w.r.t. f, the size of their set-theoretic union is necessarily bounded by the size of KB. On the other hand, while the intersection of the logical closures of all maximally consistent subbases of KB w.r.t. f can be finitely encoded as the disjunction of all such subbases, the resulting revised base contains only one formula[3]; thus, the whole content of KB would be removed, would a second severe revision occur.

Because it is based on the computation of $Ker(KB, f)$, our approach is closely related to both WIDTIO base revision [41], and safe base revision [2]

[2]Several criteria based on subset maximality or cardinality maximality can be used to select some maximally consistent subbases in presence of an information preference, i.e. assuming that the KB is stratified.

[3]Or two, if the pair formed by such a disjunction and f is considered as the revised base.

(that coincide when no preference information is available [31], and share the two valuable features listed above). Indeed, in [41], a purely standard logic approach to revision (namely, WIDTIO, i.e. "When In Doubt Throw It Out") requires $Ker(KB, f)$ to be discarded in case of a severe revision: $KB \circ_{WIDTIO} f$ is defined as $KB \circ_{WIDTIO} f = (KB \setminus Ker(KB, f)) \cup \{f\}$.

Instead, we propose to weaken $Ker(KB, f)$. As we will see in the next section, this may have a great influence on the revised epistemic state in the context of iterated revisions.

5 INFERRING FROM A REVISED KNOWLEDGE BASE

Let $\leq$ be any pre-order among interpretations from which a preferred-models inference relation can be defined. Let us now define an inference relation $\mid\sim_{R,\leq}$ under which revised KBs will be interpreted. Actually, the inference relation will be nonmonotonic; it will interpret the Abnormality propositions from $AB_{Revision}$ and AB_{KB} separately, and successively. From a semantical point of view, the inference relation will use a concept of *preferred revision-preferred model*. Such models minimize abnormality according to two principles:

Principle 1.
Elements from $AB_{Revision}$ have a higher priority than elements from AB_{KB}. Elements of $AB_{Revision}$ encode unexpected abnormalities, corresponding to severe revisions. They must be interpreted in such a way that consistency is restored. This can easily constrain the set of possible truth values of the elements of AB_{KB}. Structural abnormalities from this latter set must also be minimized.

Principle 2.
The elements of $AB_{Revision}$ are prioritized, *à la* prioritized circumscription [24] but with one abnormality symbol per stratum, only. A higher priority is given to more recently introduced Abnormality propositions. Accordingly, every Ab_{R_i} is preferred $false$ (except the most recently introduced Abnormality proposition, which must be $true$ to revise the last state of inconsistency).

Formally, let us assume without loss of generality that PS is totally ordered (using any order) and that interpretations I over $PROP_{PS}$ are represented as words over $\{0, 1\}$ s.t. the i^{th} bit of I is 1 iff the i^{th} symbol of PS is interpreted as 1 in I. For every subset E of PS, let $I[E]$ denote the restriction of I to the symbols of E. On this ground, we can define a total pre-order among the models of KB:

DEFINITION 5 (revision-preference). Let I and J be two models of KB. I

is said at least as *revision-preferred* as J, noted $I \leq_R J$, iff $I[AB_{Revision}] \leq_{lex} J[AB_{Revision}]$ holds, where $\leq_{lex}$ is the reverse lexicographic ordering on words over $\{0,1\}$, induced by $0 < 1$.

The revision-preferred models of KB are the minimal models of KB w.r.t. $\leq_R$.

DEFINITION 6 (revision-preferred model). The set of all revision-preferred models of KB is defined by $\mathcal{M}_R(KB) = min(\mathcal{M}(KB), \leq_R)$.

It is worth noting that all revision-preferred models I of KB coincide on $AB_{Revision}$. Such models can be characterized in the following constructive way. Let Ab_{R_z} be the lastly introduced symbol of $AB_{Revision}$ in KB:

Procedure Revision-preferred-models
Input: KB, $AB_{Revision}$, Ab_{R_z}
Output: an interpretation I over $PROP_{AB_{Revision}}$
begin
 for every Ab_{R_i} in $AB_{Revision}$ from Ab_{R_z} downto Ab_{R_1} do
  if $KB_{Ab_{R_i} \leftarrow false}$ is consistent
   then $[\![Ab_{R_i}]\!](I) = 0$
   else $[\![Ab_{R_i}]\!](I) = 1$
 return I
end

All the remaining symbols from $AB_{Revision}$ are set to 0 in the revision-preferred models I of KB[4]. Now, among the revision-preferred models, the preferred ones are those that minimize the structural abnormalities, according to any preset preferred-models semantics:

DEFINITION 7 (preferred revision-preferred model). Let $\leq$ be any preorder among the models of KB. The set of all preferred revision-preferred models of KB is defined by $\mathcal{M}_{R,\leq}(KB) = min(\mathcal{M}_R(KB), \leq)$.

Finally, the corresponding preferred-models inference relation $|\sim_{R,\leq}$ can be defined as follows:

DEFINITION 8 (preferred revision-preferred inference). Let $\leq$ be any preorder among the models of KB. $KB \,|\sim_{R,\leq} f$ iff f is satisfied in all preferred revision-preferred models of KB.

Clearly enough, in the situation where no severe revision has been performed, no symbol from $AB_{Revision}$ occurs in KB. Thus, every model is a revision-preferred one and the preferred revision-preferred models of KB are just the preferred models of KB, i.e. those that are minimal w.r.t. $\leq$. Contrastingly, in the situation where no structural abnormalities have been

[4]This is required by the definition but is not very significant since such remaining symbols do not occur in KB.

envisioned (i.e. $AB_{KB} = \emptyset$), the preferred revision-preferred models of KB are its revision-preferred models.

EXAMPLE 9 (continued). For simplicity, let us assume that $AB_{KB} = \emptyset$. Let $KB = \{\neg Bird \vee Fly, Bird\}$. We have:

1. $KB \mid\sim_{R,\leq} Fly \wedge Bird$. The (preferred) revision-preferred models of KB coincide with the models of KB.

2. $KB \circ_{full-meet} \neg Fly = \{\neg Bird \vee Fly \vee Ab_{R_1}, Bird \vee Ab_{R_1}, \neg Fly\} \mid\sim_{R,\leq} \neg Fly$. Once again, the (preferred) revision-preferred models I of $KB \circ_{full-meet} \neg Fly$ coincide with its models and satisfy $[\![Ab_{R_1}]\!](I) = 1$.

3. $(KB \circ_{full-meet} \neg Fly) \circ_{full-meet} Fly = \{\neg Bird \vee Fly \vee Ab_{R_1}, Bird \vee Ab_{R_1}, \neg Fly \vee Ab_{R_2}, Fly\} \mid\sim_{R,\leq} Fly \wedge Bird$. This time, the (preferred) revision-preferred models I of the revised base do not coincide with its models. Every I is s.t. $[\![Ab_{R_2}]\!] = 1$ and $[\![Ab_{R_1}]\!] = 0$.

As evoked before, our framework does not enable taking into account preference information about what subbases should be kept. Indeed, the two revision principles given above are to be applied to any KB, thus the way unexpected abnormalities from $AB_{Revision}$ are minimized does not depend on KB. Nevertheless, some preference information about structural abnormalities can be incorporated within our framework; it is supported by the given pre-order $\leq$, and different pre-orders can easily give rise to different inference relations $\mid\sim_{R,\leq}$ from revised KBs. Furthermore, some flexibility can be achieved in our framework by modifying the status of some occurrences of Abnormality propositions. Indeed, when an unexpected, revision-generated exception has been revealed, the knowledge engineer can easily decide to view it afterwards as a structural abnormality. This modification can easily change what can be inferred.

6 RATIONALITY AND COMPLEXITY ISSUES

It is time now to weight the pros and the cons of our approach, especially w.r.t. rationality and complexity issues.

From a formal point of view, it is not easy to check our non-standard approach against usual rationality postulates for belief revision. On the one hand, we use a nonmonotonic inference framework instead of the standard deductive one. On the other hand, the ontology is not constant since additional Abnormality propositions are introduced during the successive severe revision steps. As a consequence, our approach does not satisfy the inclusion postulate proposed by [16]: in the general case, $KB \circ_{full-meet} f$ is not a subset of $KB \cup \{f\}$. Also, standard AGM rationality postulates apply to

belief sets, i.e. deductively closed sets of beliefs, whereas we consider *belief bases* (more precisely finite sets of formulas).

Adapting Nebel's technique [27] to our nonmonotonic framework, we can associate a belief set revision operator $*_{full-meet}$ to our belief base one. Let $Cn(KB) = \{h \text{ s.t. } KB \mid\sim_{R,\leq} h\}$ the nonmonotonic closure of KB given $\mid\sim_{R,\leq}$. A belief set, i.e. the set of beliefs of some agent, is the nonmonotonic closure of a belief base KB. The belief set operator $*_{full-meet}$ generated by $\circ_{full-meet}$ is given by $Cn(KB)^{*_{full-meet}} f = Cn(KB^{\circ_{full-meet}} f)$.

It is easy to show that $*_{full-meet}$ obeys the six basic rationality postulates by Alchourrón et al. [1]. However, the two supplementary postulates are not satisfied in the general case. This is not very surprising since WIDTIO base revision typically exhibits the same behaviour w.r.t. rationality postulates.

As evoked previously, our revision process does not entail an exponential increase of size for the revised KB as in many syntax-based approaches, especially [12], and it allows for iterated belief revision in a non-trivial way (unlike some techniques proposed so far, see e.g. [27]). Moreover, as it will be shown in the next section, it is far less destructive than WIDTIO in presence of iterated revision, hence adhering more closely to the principle of minimal change.

We believe that the way we hide pieces of information by weakening them can open new alternative research directions in the study of rationality postulates for iterated belief revision, which currently gives rise to an unsettled controversy (see e.g. [7] [18] [40] and [13]).

Let us now turn to computational aspects. Propositional belief base revision is intractable in the general case. This intractability concerns both the complexity of deciding whether a formula is implied by a revised belief base, and the complexity of representing a revised belief base. Thus, [31] analyzes in depth the first issue and shows that inference from a revised KB typically is at the first level or even at the second level of the polynomial hierarchy.

Like (full-meet) WIDTIO [41] and safe base revision [2], our approach suffers from a high worst case complexity. In order to simplify things, let us first assume that $AB_{KB} = \emptyset$ (in this situation, the chosen pre-order $\leq$ over models is not significant).

PROPOSITION 10. *Let KB, f be two formulas and $AB_{KB} = \emptyset$. Checking whether $KB \mid\sim_{R,\leq} f$ holds is Π_2^p-complete.*

Sketch of proof: In order to check that $KB \not\mid\sim_{R,\leq} f$, it is sufficient to guess a model I of KB, then check that it is a revision-preferred one, and that $I \not\models f$ holds. Clearly enough, checking that a given model I of KB is a revision-preferred one can be achieved in polynomial time on a Turing machine equipped with an NP oracle. Indeed, showing that I is not a revision-preferred model of KB can be achieved easily in nondeterministic polynomial time by guessing a model J of KB and checking that $J \leq_R I$

and $I \not\leq_R J$. Equivalently, I is a revision-preferred model of KB iff it coincides with all the revision-preferred models of KB on the symbols of $AB_{Revision}$, and the required truth values of these symbols can be computed thanks to a number of calls to SAT that is bounded by the size of KB (see the procedure `Revision-preferred-models` in the previous section). Accordingly, determining whether $KB \mid\sim_{R,\leq} f$ is in Π_2^p. Because inference from WIDTIO revised bases is a restriction of inference w.r.t. $\mid\sim_{R,\leq}$ (cf. next section), and inference from WIDTIO revised bases is Π_2^p-hard (see Lemma 6.2 from [11]), the Π_2^p-completeness of inference from $\mid\sim_{R,\leq}$ follows.

Now, in the case where AB_{KB} is not empty (so that the given preorder $\leq$ is significant), the complexity upper bound of the inference problem will be given by the complexity of checking whether a model I of KB is a preferred revision-preferred model of KB. If such a model checking problem belongs to coNP (which is the case for many preference relations $\leq$), checking $KB \mid\sim_{R,\leq} f$ still belongs to Π_2^p, hence it is Π_2^p-complete.

Focusing on the computation of revised KBs only, the main difficulty of both WIDTIO and our technique is the computation of $Ker(KB, f)$. Indeed, computing minimally inconsistent subbases of KB w.r.t. f can be hard; especially, we can easily prove that, given a finite set KB of propositional formulas and a revision formula f, checking whether a given subset of KB is a minimally inconsistent subbase of KB w.r.t. f is BH_2-complete. Moreover, from a worst case computational complexity point of view, checking the membership of a formula g from KB to $Ker(KB, f)$ is Σ_2^p-complete (this is a consequence of Theorem 8.2 from [11]). This makes the full-meet approach impractical in the worst case. Fortunately, from the practical side, things are not so bad.

7 APPROXIMATING KERNELS THANKS TO LOCAL SEARCH

A computationally valuable feature of our approach is the existence of an efficient local search-based heuristic for approximating kernels when KB and f are two CNF formulas (which is not so much an unrealistic assumption) [23].

Since this is just a heuristic technique, we cannot guarantee that the output of the proposed algorithm will actually be $Ker(KB, f)$ when the inputs are KB and f. Nevertheless, this does not question the practical utility of the technique. While local search techniques for SAT are not guaranteed to find out a model for every consistent KB, they enable proving the consistency of many of them, which cannot be achieved using a systematic method. In the same vein, experiments show that our heuristic technique can be used to derive or, at least, to focus the search on kernels in many situations.

Since our heuristic technique is based on local search, let us first recall the

most representative local search algorithm, namely Selman et al.'s GSAT [38] [37]. This algorithm performs a greedy local search for a satisfying assignment of a set of propositional clauses. The algorithm starts with a randomly generated truth assignment. It then changes ("flips") the assignment of the variable that leads to the largest increase in the total number of satisfied clauses. Such flips are repeated until either a model is found or a preset maximum number of flips (MAX-FLIPS) is reached. This process is repeated as needed up to a maximum of MAX-TRIES times.

Procedure GSAT
Input: a set of clauses S, $MAX-FLIPS$, and $MAX-TRIES$
Output: a satisfying truth assignment of S, if found
```
begin
 for i := 1 to MAX-TRIES do
  I := a randomly generated truth assignment
  for j := 1 to MAX-FLIPS do
   if I is a model of S then return I
   x := a propositional variable s.t. a change in its truth
        assignment gives the largest increase (possibly negative)
        in the number of clauses of S that are satisfied by I
   I := I with the truth assignment of x reversed
 return ''no satisfying assignment found''
end
```

Let us stress that some variant (e.g. random) moves are inserted in order to escape from local minima. In the following, TSAT, a local search technique of our own, is used [22]. It makes use of a tabu list forbidding recurrent flips and appears competitive in most situations. One of the nice features of TSAT is that it drops many stochastic properties of other local search techniques and is purely deterministic as far as the initial interpretation is selected in a non-random way. However, the following experimental results can be obtained using most local search techniques, and are not TSAT specific.

Although such techniques prove efficient in showing the consistency of hard propositional CNF belief bases, they are clearly logically incomplete in the sense that they do not cover the whole search space and cannot thus directly prove that a formula is inconsistent. In this respect, when KB is inconsistent, GSAT-like algorithms provide us with a way to approximate $MAXSAT$, i.e. finding a cardinality maximal subbase of KB. Indeed, at each step, the unsatisfied clauses form a hitting set of the minimally inconsistent subbases of KB. Such a powerful computational approach to a maxichoice base revision policy is not investigated here (see [15]), as we focus on the full-meet one.

Some of us have discovered recently that the trace of local search algorithms can allow one to determine kernels very often when they fail to prove

consistency within a preset amount of computing time [23]. More precisely, for each clause, taking each flip as a step of time, the number of times during which this clause is falsified is updated. A similar trace is also recorded for each literal occurring in the belief base, counting the number of times it has appeared in the falsified clauses. All the clauses from KB that are falsified t times more than the mean number of falsification per clause are delivered. t is an empirically fixed threshold. When the local search technique fails to prove consistency, it appears extremely often that the most often falsified clauses from KB form a superset of the set-theoretic union of the minimally inconsistent subbases of KB when KB is actually inconsistent. Likewise, literals that exhibit the highest scores take part in these subbases.

Accordingly, our heuristic is based on the following assumption:

Heuristic: The set S_{TSAT} of clauses from KB delivered by TSAT when it fails in finding a model of $KB \cup \{f\}$ is a superset of $Ker(KB, f)$.

Depending on the computational effort we are ready to spend, the validity of the heuristic can be tested by:

1. Checking that $KB \setminus S_{TSAT}$ is consistent. If this is not the case, the heuristic is wrong.

2. Checking that $S_{TSAT} \cup \{f\}$ is inconsistent. If this is not the case, the heuristic is wrong.

The consistency checks can be performed exactly (using a complete algorithm for SAT, e.g. a Davis-Putnam one [8]) or approximately using a local search algorithm (assuming that the input formula is inconsistent if no model has been found within a preset time). Note that the test phase can be achieved in practice quite efficiently.

In the case where the heuristic is found wrong, we can turn to a systematic, more computationally expensive technique, to compute $Ker(KB, f)$. Alternatively, we can modify the threshold or modify the parameters of T-SAT (the maximum numbers of tries and flips) and, in this latter case, run it again.

It is worth noticing that replacing $Ker(KB, f)$ by a superset S_{TSAT}, as it can be delivered by the first phase of our technique, does not question the consistency of the revised KB. It would just lead to an over-cautious revision policy that would weaken too many pieces of information. Such a policy could be adopted when limited computational resources justify a trade-off between the computational cost and a distance between the probable kernels and the real ones.

Let us stress that in the general case the heuristic provides us with an *approximation* of $Ker(KB, f)$ in the following sense: some formulas from

$Ker(KB, f)$ might not appear in S_{TSAT} and S_{TSAT} might contain formulas that do not belong to $Ker(KB, f)$. Clearly enough, if the heuristic is not found wrong, we cannot be sure that S_{TSAT} actually is a superset of $Ker(KB, f)$ (the test phase enables identifying *some* situations where the heuristic is wrong, but *not all of them*). However, under severe time constraints, such an approximation can prove useful as such.

Empirically, the validity of the heuristic depends on several aspects. First, in our nonmonotonic framework, inconsistency is the exception and should normally concern a few formulas, only. Moreover, in large real-world KBs, inconsistency is often due to a limited number of contradictory rules and data (unless total inconsistency of the modeled agent!), which allow a small $Ker(KB, f)$ to emerge very often. Such a phenomenon matches most philosophical and psychological models of beliefs. Also, the local search algorithm should "cover" the space of interpretations in the most uniform and systematic way. This can be achieved partly by a large number of tries, with initial interpretations that are sufficiently distant. Clearly, the accuracy of the heuristic depends also on the consumed computation time. We are currently working on an improved system that stops its computation when a stable (probable) $Ker(KB, f)$ has clearly emerged. Obviously enough, the accurracy of the heuristic also depends on the interdependence level between formulas from $Ker(KB, f)$ and the other ones: whenever a formula from $Ker(KB, f)$ is falsified, not too many formulas from $KB \setminus Ker(KB, f)$ should be falsified at the same time. This is related to a *knowledge locality* principle, which is often a valid assumption in actual KBs.

Finally, let us note that whenever the revision formula f does not contain any Abnormality proposition, no formula from KB among those that have been weakened so far can belong to $Ker(KB, f)$. Indeed, in this situation, every Ab_{R_i} literal occurs only positively in KB. Thus, it cannot be the case that a weakened formula belongs to a minimally inconsistent subbase of KB w.r.t. f (assuming that every Ab_{R_i} is *true* is always possible and sufficient to satisfy the weakened formulas). This remark is valuable from a computational point of view since it means that whenever sufficiently many severe revisions have been done, computing $Ker(KB, f)$ can be easier since the number of candidate formulas from KB will be smaller (all the formulas that have been weakened so far can be omitted).

The main remarkable feature of this approach lies in its effective tractability for very large applications. In spite of the fact that it does not require a subset of propositional clausal logic to be used (like the Horn one) as it is sometimes required and that it is nonmonotonic, it proves empirically efficient for very large KBs, at least when inconsistency can be related to a small subpart of KB.

Very comprehensive experimentations have been conducted to test the heuristic, using standard benchmarks for propositional reasoning and consistency checking that are taken from [10] or have been proposed in the

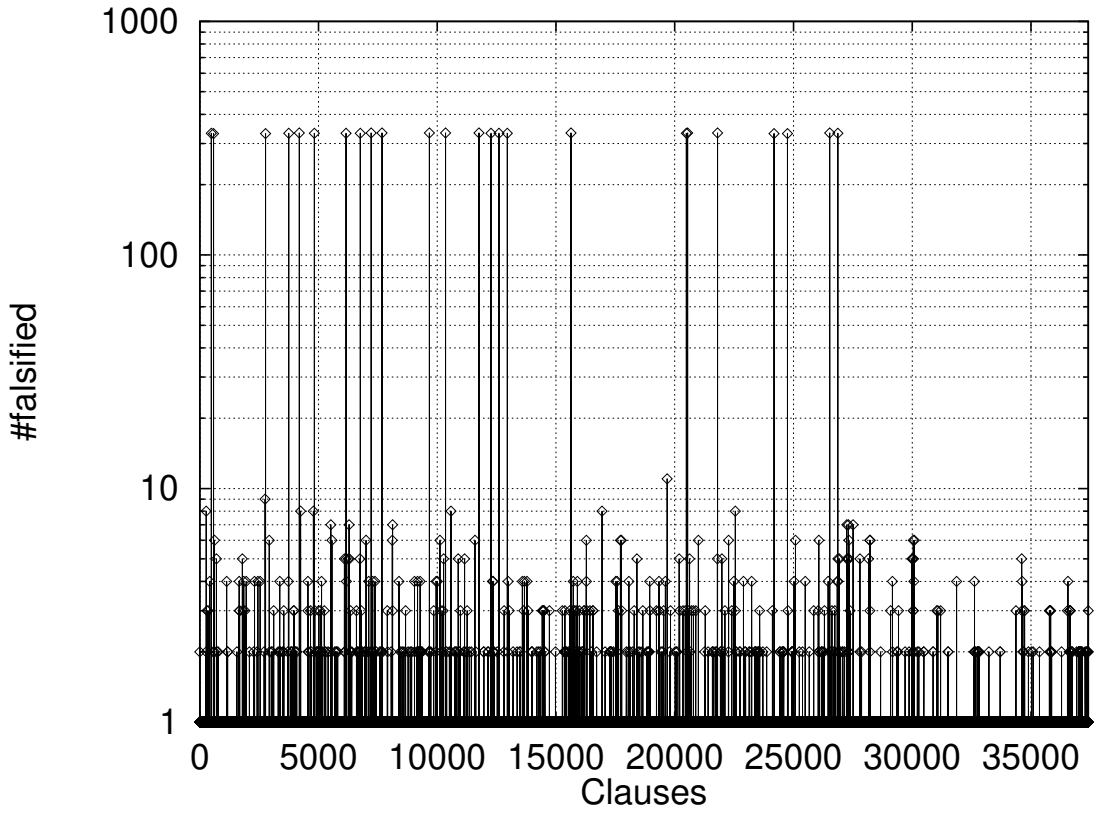

Figure 1. Monotonic framework.

meantime. All experiments have been conducted on Pentium 166 Mhz PCs. Let us just give a typical example. First, let us illustrate our approach when a purely monotonic knowledge is considered (i.e. $AB_{KB} = \emptyset$). We mixed a very large consistent propositional problem proposed by [17] representing a planning-related knowledge base with a globally inconsistent one proposed by [10]. These problems contain 37388 and 24 clauses (2883 and 9 variables), respectively. In our framework, we can interpret the first problem together with some clauses of the second one as representing a consistent KB, while the remaining part of the second one represents the clausal formula f to be inserted into KB. Clearly, $KB \wedge f$ is inconsistent.

TSAT (with a weight option [26] [37]) was run on $KB \wedge f$. Figure 1 shows (using a logarithmic scale) the trace (i.e. the score of the clauses) delivered after 2 seconds CPU computation time. The inconsistent subbase clearly emerges and is easily recognized. It consists of the 24 clauses with the highest scores. Weakening the subpart of this subbase that belongs to KB is now a straightforward task.

Let us now move to the nonmonotonic case. To our best knowledge, we are not aware of *large* propositional benchmarks in the nonmonotonic area representing real-world problems. Benchmarks in this domain have been

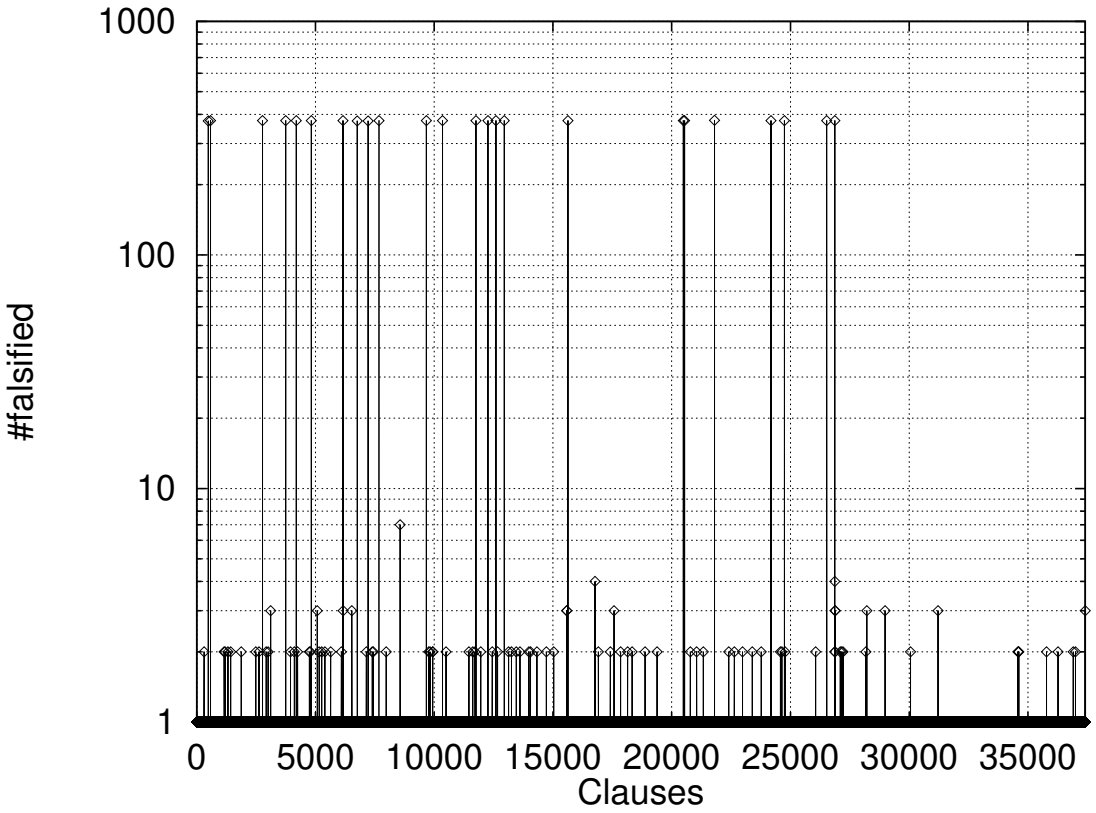

Figure 2. Nonmonotonic framework.

defined mainly in order to check the principles guiding the logics and not their computational properties. Therefore, we had to define our own benchmarks. For instance, we transformed monotonic ones so that they actually incorporate Abnormality propositions in some of their rules. Accordingly, the above knowledge base has been enriched in the following way. 10 % of clauses have been given their own additional literals Ab_i from AB_{KB}, representing possible future exceptions to the rules. TSAT was run again during 2 seconds. Figure 2 gives the trace of the computation. The introduction of Abnormality propositions does not modify the empirical results.

8 RELATED WORKS

As evoked previously, our approach to belief base revision is closely related to WIDTIO [41]. Especially, in the basic monotonic case and when only one-step of revision is considered, both our approach and WIDTIO coincide from an inferential point of view.

PROPOSITION 11. *Assume that KB, f and g do not contain any Abnormality proposition. $KB^{\circ_{WIDTIO}} f \models g$ iff $KB^{\circ_{full-meet}} f \mid\sim_{R,\leq} g$*

Sketch of proof: If $KB \cup \{f\}$ is consistent, then $Ker(KB, f) = \emptyset$, so $\mathcal{M}(KB \circ_{WIDTIO} f) = \mathcal{M}_{R,\leq}(KB \circ_{full-meet} f)$. Otherwise, every formula removed in the WIDTIO approach is weakened by Ab_{R_1} in our approach, and the converse also holds. Since Ab_{R_1} is necessarily *true* in the revised KB, the weakened formulas do not participate to inference. More formally, for every preferred revision-preferred model I of $KB \circ_{full-meet} f$, there exists a model J of $KB \circ_{WIDTIO} f$ s.t. for every symbol p of P, $[\![p]\!](I) = [\![p]\!](J)$, and *vice-versa*. The fact that g does not contain any Abnormality symbol is sufficient to end up the proof.

EXAMPLE 12 (continued). Let $KB = \{\neg Bird \vee Fly, Bird\}$. We have:

1. $KB \models Fly \wedge Bird$.
2. $KB \circ_{WIDTIO} \neg Fly = \{\neg Fly\} \models \neg Fly$.

However, our approach extends WIDTIO in at least two directions. First, WIDTIO is based on a purely standard logic framework. Second, our cautious approach is less "destructive" than WIDTIO in the sense that sources of inconsistency are "hidden" instead of being cancelled. This may lead to more "rationality", especially with respect to the principle of minimal change when iterated belief revision steps are processed. Indeed, WIDTIO is unnecessarily too much destructive.

EXAMPLE 13 (continued). $(KB \circ_{WIDTIO} \neg Fly) \circ_{WIDTIO} Fly = \{Fly\}$. Compared with our approach, let us note that $Bird$ is lost; actually when WIDTIO gives up a piece of information, it is forever (unless it is added again).

It is worth noting that the gain in expressiveness and "rationality" achieved by our approach (compared with WIDTIO) does neither necessarily lead to a degradation of the worst case complexity, with respect to inference from the revised KB. Indeed, inference from (full-meet) WIDTIO revised bases is Π_2^p-complete as well (see Theorem 5.21 from [31]). Moreover, once $Ker(KB, f)$ is derived, the revised base w.r.t. WIDTIO and our technique can be computed in polynomial time. Especially, removing inconsistency by weakening all its possible causes is not more expensive than deleting them.

Our approach can also be related to *linear base revision* from stratified bases [30]. In this approach, only one subbase of KB consistent with f has to be considered.

Let us associate to every (revised) belief base considered in our approach a stratified belief base s.t.:

- Originally (i.e. when no revision formula has been considered), all the formulas from KB are given the same priority, namely priority 0.

- Afterwards, when the i^{th} revision formula f is considered, every formula of $KB \setminus Ker(KB, f)$ is given priority i, f is also given priority i and all the remaining formulas keep their previous priorities.

Let KB_i denote the subset of KB consisting of all the formulas of priority i. Let z be the highest priority among the formulas in KB.

Procedure Inference-from-prioritized-KB
Input: a stratified KB
Output: a subset S_{max} of KB
```
begin
 S_max ← KB_z
 for every i from z − 1 downto 1 do
  if S_max ∪ KB_i is consistent
   then S_max ← S_max ∪ KB_i
 return S_max
end
```

A formula g is viewed as a consequence of such a stratified belief base KB iff it is a logical consequence of S_{max}. Interestingly, this occurs exactly when $KB \mid\sim_{R,\leq} g$, provided that $AB_{KB} = \emptyset$ and that the successive revision formulas f do not contain any Abnormality proposition. In other words, under these restrictive assumptions, inference in our approach and inference from such linearly revised stratified bases coincide. This can be easily explained by the fact that $AB_{Revision}$ is linearly ordered, and to each symbol Ab_{R_i} of $AB_{Revision}$ is associated a subset of the original KB, those formulas that are weakened by Ab_{R_i}, in so far as all the formulas from the subset can be considered as present in the revised KB when Ab_{R_i} is interpreted as $false$, and absent otherwise.

EXAMPLE 14 (continued).

1. We associate to the original $KB = \{\neg Bird \vee Fly, Bird\}$ the stratified base $KB^0 = \{\neg Bird \vee Fly(0), Bird(0)\}$. The corresponding set S^0_{max} is KB^0, from which $Bird$ and Fly are inferrable.

2. Now, KB is revised by $\neg Fly$. The corresponding revised base is associated to $KB^1 = \{\neg Bird \vee Fly(0), Bird(0), \neg Fly(1)\}$. The associated set S^1_{max} is $\{\neg Fly\}$, from which only $\neg Fly$ is inferrable.

3. Lastly, the resulting base is revised by Fly. The corresponding revised base is associated to $KB^2 = \{\neg Bird \vee Fly(2), Bird(2), \neg Fly(1), Fly(2)\}$. The associated set S^2_{max} is $\{\neg Bird \vee Fly, Bird, Fly\}$, from which both $Bird$ and Fly are inferrable.

While they give equivalent results in this restricted situation, our approach and linear base revision do not necessarily coincide in the general case, especially when AB_{KB} is not empty, or when the revision formulas may contain Abnormality symbols. Moreover, the fact that the ontology increases while severe revisions are performed enables *reasoning about the revision process itself* : the Ab_{R_i} symbols also play the role of temporal markers, allowing one to decide efficiently whether or not some initial piece of knowledge has been severely revised. For instance, the presence of the formula $\neg Bird \vee Fly \vee Ab_{R_1}$ in $(KB^{\circ_{full-meet}} \neg Fly)^{\circ_{full-meet}} Fly$ shows that the original piece of knowledge $\neg Bird \vee Fly$ has been subject to a severe revision.

9 CONCLUSIONS AND PERSPECTIVES

Most of the times, belief revision has been investigated from a theoretical point of view. As such, this domain has sometimes been criticized for its lack of computational theory-validating large-size applications. On the one hand, this paper is a modest contribution towards filling this missing facet of belief revision. On the other hand, non-conventional positions are adopted in this paper with respect to various facets of belief revision. As conclusive remarks, let us summarize and motivate them briefly again. We believe that each of them could open new fruitful paths of research.

- Belief revision should not be studied from the philosophical and cognitive sides of artificial intelligence only, but also from its (knowledge) engineering one. As such, specific new revision theories can be proposed that are guided by specific motivations: empirical computational efficiency, importance of the syntax and software quality properties, mainly.

- In this respect, we propose a combined use of nonmonotonic logics and belief revision. The nonmonotonic logic ingredients should be simple enough to allow an easy encoding of priorities, rules of default reasoning and exceptions. Inconsistencies should be prevented in advance as much as possible by an adequate encoding using this logic; severe revisions should concern inconsistencies that form a mere remainder of all inconsistencies that could occur if a standard representation and deductive logic were selected.

- Throwing out beliefs to restore consistency is unnecessary destructive. We propose to weaken beliefs instead. Quite remarkably, the weakening ingredients that we propose coincide with the nonmonotonic ones. Actually, beliefs that are inconsistent with respect to the newly introduced ones are interpreted as exceptional cases that do not actually

occur. Weakening instead of throwing out opens new perspectives with respect to iterated belief revision considerations.

- Bad worst case complexity results for belief revision should not be misinterpreted. They do not prevent computational efficiency from being reached in many actual large-size applications.

- On the contrary, techniques based on local search techniques offer efficient heuristic tools to implement belief revisions techniques for very large propositional knowledge bases. The power of these techniques is currently unmatched, and offers interesting anytime properties that can be essential when time-critical belief revision processes are under consideration.

APPENDIX: BASIC ELEMENTS OF COMPUTATIONAL COMPLEXITY

One of the main purposes of complexity theory[5] is to classify problems according to their worst case requirements on computational resources depending on the size of the input. In this framework, a problem is a generic question, i.e., a set of specific instances. Specifically, a *decision problem* is one that has only "yes" and "no" as possible answers. Formally, a decision problem can be considered as a formal language consisting of the set of all its "yes" instances.

It is usually acknowledged that an *efficient algorithm* is one that runs in time polynomial in the size of the input (assuming a standard model of computation, e.g., a sequential deterministic Turing machine). Accordingly, a decision problem is considered *efficiently solvable* iff there exists an algorithm that can classify every instance of it in a number of computational steps that is polynomially bounded.

The class of all decision problems that are solvable in polynomial time is denoted by P. All the remaining ones require exponential time and are considered not efficiently solvable. In order to determine whether a problem is efficiently solvable or not, it is sufficient to point out an efficient algorithm to solve it, or to prove that such an algorithm cannot exist.

Unfortunately, for many problems, no polynomial time algorithms are known but it seems to be impossible to prove that super-polynomial time is actually required. For all these problems, the separation efficiently solvable vs. non-efficiently solvable is not fine-grained enough to classify them in a computationally valuable way: there is a need for more refined classes for problems that are not known in P.

[5] See e.g. [33] for details.

The notion of non-deterministic Turing machine is an important tool to achieve this goal. Thus, the class of all languages (encoding decision problems) that can be recognized in polynomial time by such a non-deterministic machine is denoted by NP. Because a deterministic Turing machine can be considered as a non-deterministic one, the inclusion $\mathsf{P} \subseteq \mathsf{NP}$ is established; however, the converse is the famous open problem: $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ (that is conjectured false). Among all the problems in NP, the hardest ones are those from which every problem in NP can be *polynomially many-one reduced*: such problems are referred to as NP-*complete*. If any of them has a polynomial algorithm, then $\mathsf{P} = \mathsf{NP}$ holds. Accordingly, it is believed that it is impossible to solve NP-complete problems in polynomial time. SAT, the problem of determining whether a propositional formula in conjunctive normal form is satisfiable, is the prototypical NP-complete problem [6]. Its complementary problem $UNSAT$ (consisting of determining whether a propositional formula in conjunctive normal form is unsatisfiable) is not necessarily in NP (in contrast to P, NP is not known to be closed under complementation). It is assigned to the class coNP that contains the complementary problems to problems of NP. It is conjectured that $\mathsf{NP} \neq \mathsf{coNP}$. In the following, let us note $\bar{L}$ the complementary problem of L.

To go further into the classification of non-efficiently solvable problems, another important tool is the notion of Turing machine (deterministic or non-deterministic) *with oracle*. Let X be a class of decision problems. P^X (resp. NP^X) is the class of all decision problems that can be solved in polynomial time using a deterministic (resp. non-deterministic) Turing machine that can use an oracle for deciding a problem $Q \in X$ for "free" (i.e., within a constant, unit time). On this basis, the classes Δ_k^p, Σ_k^p and Π_k^p are defined by:

- $\Delta_0^p = \Sigma_0^p = \Pi_0^p = \mathsf{P}$,
- $\Delta_{k+1}^p = \mathsf{P}^{\Sigma_k^p}$,
- $\Sigma_{k+1}^p = \mathsf{NP}^{\Sigma_k^p}$,
- $\Pi_{k+1}^p = \mathsf{co}\Sigma_{k+1}^p$.

Thus, $\Sigma_1^p = \mathsf{NP}$ and $\Pi_1^p = \mathsf{coNP}$. The *polynomial hierarchy* PH is the union of all Σ_k^p (for k non-negative integer). While it is easy to check that both $\Delta_k^p \subseteq \Sigma_{k+1}^p$, $\Delta_k^p \subseteq \Pi_{k+1}^p$, $\Sigma_k^p \subseteq \Delta_{k+1}^p$, and $\Pi_k^p \subseteq \Delta_{k+1}^p$ hold for every k, it is unknown whether the inclusions are proper or not (but it is conjectured that it is the case). Thus, it is strongly believed that the polynomial hierarchy does not collapse, i.e., is a truly infinite hierarchy.

In order to discriminate further among the problems from Δ_2^p, one can focus on the number of calls to an NP-oracle that are used. Thus, the classes BH_k and coBH_k are defined in the following way:

- $\mathsf{BH}_0 = \mathsf{P}$,
- $\mathsf{BH}_1 = \mathsf{NP}$,
- $\mathsf{BH}_2 = \{L_1 \cap \bar{L}_2 | L_1 \in \mathsf{NP}, L_2 \in \mathsf{NP}\}$,
- $\mathsf{BH}_{2k+1} = \{L_1 \cup L_2 | L_1 \in \mathsf{BH}_{2k}, L_2 \in \mathsf{NP}\}$ with $k \geq 1$,
- $\mathsf{BH}_{2k+2} = \{L_1 \cap \bar{L}_2 | L_1 \in \mathsf{BH}_{2k+1}, L_2 \in \mathsf{NP}\}$, with $k \geq 1$,
- $\mathsf{coBH}_k = \{L | \bar{L} \in \mathsf{BH}_k\}$.

The *boolean hierarchy* BH (over the complexity class NP) is the union of all BH_k (for k non-negative integer). It turns out that BH is equivalent to the class of problems that can be solved in deterministic polynomial time using a constant number of calls to an NP-oracle.

Acknowledgments
This paper has been supported in part by a *"Contrat d'Objectifs de la Région Nord/Pas-de-Calais"* and by the IUT de Lens. We are indebted to Salem Benferhat for so many interesting comments about a preliminary version of this paper and to Bertrand Mazure for his help with respect to the implementation side.

REFERENCES

[1] C.E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] C.E. Alchourrón and D. Makinson. On the logic of theory change: safe contractions. *Studia logica*, 44:405–422, 1985.

[3] S. Benferhat, D. Dubois, and H. Prade. How to infer from inconsistent beliefs without revising. In *IJCAI'95*, pages 1449–1455, 1995.

[4] B. Bessant, É. Grégoire, P. Marquis, and L. Saïs. Combining nonmonotonic reasoning and belief revision: a practical approach. In *AIMSA'98*, volume 1480 of *Lecture Notes in Artificial Intelligence*, pages 115–128. Springer-Verlag, 1998.

[5] M. Cadoli, F. Donini, P. Liberatore, and M. Schaerf. The size of a revised knowledge base. In *PODS'95*, pages 151–162, 1995.

[6] S.A. Cook. The complexity of theorem-proving procedures. In 3^{rd} *Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[7] A. Darwiche and J. Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89:1–30, 1997.

[8] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Journal of the Association for Computing Machinery*, 5:394–397, 1962.

[9] A. del Val. On the relation between the coherence and foundations theories of belief revision. In *AAAI'94*, pages 909–914, 1984.

[10] DIMACS, 1993. Second SAT challenge organized by the Center for Discrete Mathematics and Computer Science of Rutgers University, `http://dimacs.rutgers.edu/Challenges`.

[11] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactual. *Artificial Intelligence*, 57:227–270, 1992.

[12] R. Fagin, J.D. Ullman, and M.Y. Vardi. On the semantics of updates in databases. In *PODS'83*, pages 352–365, 1983.
[13] N. Friedman and J.Y. Halpern. Belief revision: a critique. In *KR'96*, pages 429–431, 1996.
[14] P. Gärdenfors. Belief revision and nonmonotonic logic: Two sides of the same coin? In *ECAI'92*, pages 768–773, 1992.
[15] É. Grégoire. Handling inconsistency efficiently in the incremental construction of stratified belief bases. Technical report, CRIL, 1999.
[16] S.O. Hansson. Reversing the Levi identity. *Journal of Phiolosophical Logic*, 22:637–669, 1993.
[17] H. Kautz and B. Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *AAAI'96*, pages 1194–1201, 1996.
[18] D. Lehman. Belief revision revisited. In *IJCAI'95*, pages 1534–1540, 1995.
[19] P. Liberatore and M. Schaerf. Relating belief revision and circumscription. In *IJCAI'95*, pages 1557–1553, 1995.
[20] P. Liberatore and M. Schaerf. The complexity of model checking for belief revision and update. In *AAAI'96*, pages 556–561, 1996.
[21] D. Makinson and P. Gärdenfors. Relations between the logic of theory change and nonmonotonic logic. In Fuhrmann and Moreau, editors, *Workshop on the logic of theory change*, pages 185–205. LNCS 465, Springer, 1991.
[22] B. Mazure, L. Saïs, and É. Grégoire. Tabu search for sat. In *AAAI'97*, pages 281–285, 1997.
[23] B. Mazure, L. Saïs, and É. Grégoire. Boosting complete techniques thanks to local search. *Annals of Mathematics and Artificial Intelligence*, 22:319–322, 1998.
[24] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.
[25] Y. Moinard. Revision and nonmonotonicity. *International Journal of Intelligent Systems*, 9, 1994.
[26] P. Morris. The breakout method for escaping from local minima. In *AAAI'93*, pages 40–45, 1993.
[27] B. Nebel. A knowledge level analysis of belief revision. In *KR'89*, pages 301–311, 1989.
[28] B. Nebel. *Reasoning and revision in hybrid representation systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1990.
[29] B. Nebel. *Belief Revision*, chapter Syntax-based approaches to belief revision, pages 53–88. Cambridge University Press, 1992.
[30] B. Nebel. Base revision operations and schemes: semantics, representation and complexity. In *ECAI'94*, pages 341–345, 1994.
[31] B. Nebel. *How hard is it to revise a belief base?*, volume 3 of *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, chapter Belief Change, pages 77–145. Kluwer Academic, 1998.
[32] A. Newell. The knowledge level. *Artificial Intelligence*, 18:87–127, 1982.
[33] C.H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
[34] O. Papini. Revision in propositional calculus. In *ECSQAU'91*, volume 548 of *Lecture Notes in Artificial Intelligence*, pages 272–276. Springer-Verlag, 1991.
[35] B. Selman and H. Kautz. An empirical study of greedy local search for satisfiability testing. In *AAAI'93*, 1993.
[36] B. Selman, H. Kautz, and D. McAllester. Computational challenges in propositional reasoning and search. In *IJCAI'97*, pages 50–54, 1997.
[37] B. Selman, H.A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability*, 1993.
[38] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *AAAI'92*, pages 440–446, 1992.
[39] Y. Shoham. *Readings in Non-Monotonic Reasoning*, chapter A semantical approach to non-monotonic logics. Morgan Kaufmann, 1987.
[40] M.A. Williams. Iterated theory base change: a computational model. In *IJCAI'95*, pages 1541–1550, 1995.
[41] M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.