Approches exactes pour le problème de satisfiabilité diversifiée

Zhifei Zheng, Sami Cherif, Rui Sá Shibasaki, Chu-Min Li
Laboratoire MIS UR 4290, Université de Picardie Jules Verne, Amiens, France

{zhifei.zheng;sami.cherif;rui.sa.shibasaki;chu-min.li}@u-picardie.fr

Résumé

La satisfiabilité diversifiée est un problème d'optimisation combinatoire qui cherche à identifier k modèles satisfaisant une formule CNF donnée tout en maximisant la somme des distances entre chaque paire de modèles. Cet article présente deux nouvelles méthodes exactes de résolution. La première approche s'appuie sur la programmation quadratique en nombres entiers, tandis que la seconde est basée sur une formulation pseudo-booléenne linéaire, pouvant également être adaptée à la satisfiabilité maximum (MaxSAT). Notre évaluation expérimentale approfondie démontre l'efficacité et la performance des deux méthodes proposées sur diverses instances. De plus, pour certaines familles d'instances, les approches proposées parviennent à obtenir des solutions optimales.

Mots-clés

Satisfiabilité diversifiée, Programmation en Nombres Entiers, Satisfiabilité Maximum, Modélisation

Abstract

Diverse Satisfiability is a combinatorial optimization problem that seeks to identify k models satisfying a given CNF formula while maximizing the sum of pairwise Hamming distances between them. This paper introduces two novel exact solution methods. The first approach relies on quadratic integer programming, while the second is based on a linear Pseudo-Boolean formulation, which can also be adapted to Maximum Satisfiability (MaxSAT). Our comprehensive experimental evaluation demonstrates the efficiency and effectiveness of both proposed methods across various benchmark instances. Furthermore, for some instance families, the proposed approaches manage to obtain optimal solutions.

Keywords

Diverse Satisfiability, Integer Programming, Maximum Satisfiability, Modeling

1 Introduction

La satisfiabilité propositionnelle (SAT) est un problème de décision consistant à déterminer si une formule en forme normale conjonctive (CNF) donnée peut être satisfaite par une affectation de variables [6], également appelée modèle de la formule. SAT est le premier à avoir été démontré NP-complet [8], et constitue un paradigme largement utilisé pour la modélisation de nombreux problèmes du monde réel, notamment en vérification matérielle et logicielle [1, 7], en planification et ordonnancement [24, 25], ou encore en bio-informatique [16, 17]. La version classique du problème SAT vise à trouver un unique modèle pour une formule CNF donnée, mais de nombreuses applications réelles exigent l'identification de plusieurs modèles présentant des propriétés structurelles diverses. Dans ce cas, le fait de s'arrêter au premier modèle ne permet pas de mener à bien l'analyse souhaitée ou risque l'omission d'informations cruciales. Un exemple représentatif qu'on peut citer est celui de la vérification bornée de modèles (Bounded Model Checking, BMC) [1], où chaque modèle de la formule correspond à un chemin de vérification possible. Toutefois, se limiter à une unique solution peut accroître le risque d'omettre des erreurs spécifiques, dans la mesure où celle-ci ne permet pas de capturer l'ensemble des comportements potentiels du système vérifié. Cette limitation met en évidence la nécessité de générer plusieurs solutions présentant une forte diversité structurelle, en vue d'accroître la robustesse des processus de débogage et de vérification.

Dans cette optique, Nadel a introduit le problème de la satisfiabilité diversifiée (Diverse Satisfiability) dans [20]. Étant donné une formule CNF, ce problème consiste à identifier k modèles de la formule maximisant la diversité structurelle. Celle-ci est quantifiée à l'aide de la somme des distances de Hamming entre toutes les paires de modèles sélectionnés, assurant ainsi une dispersion maximale dans l'espace des solutions. Dans ce même travail, l'auteur propose une méthode incomplète s'appuyant sur une heuristique de décision dédiée. Un problème connexe introduit dans la littérature est celui de la distance de Hamming maximale pour les problèmes de satisfaction de contraintes (CSP), initialement formulé par Crescenzi et Rossi dans [9]. Néanmoins, ce problème, bien qu'applicable à des instances à domaines finis, se limite à la recherche de deux modèles distincts. Dès lors, dans le cas binaire, il devient essentiellement équivalent à une instance de la satisfiabilité diversifiée avec $k = 2$.

Étant donné que le problème SAT est déjà NP-complet, le problème de la satisfiabilité diversifiée, qui requiert l'identification de k modèles satisfaisants distincts, représente un défi computationnel encore plus complexe. En particulier, pour un paramètre donné p, Misra et al. [19] démontrent que, même dans le cas restreint où $k = 2$, déterminer deux modèles qui diffèrent exactement (ou au moins) sur p variables demeure un problème NP-difficile. Dans [2], An-

gelsmark et Thapper établissent une borne supérieure de complexité en $O(1.7338^n)$ pour la résolution du problème de distance de hamming maximale sur les formules 2-SAT (c'est-à-dire la satisfiabilité diversifiée avec $k = 2$, où l'instance ne contient que des clauses de longueur 2). Plus généralement, ils montrent que, pour une formule l-SAT où l désigne la longueur maximale des clauses, une borne supérieure en $O((2a)^n)$ peut être obtenue, sous l'hypothèse que l'instance sous-jacente de l-SAT peut être résolue en $O(a^n)$ pour une certaine constante a.

Dans cet article, nous nous focalisons sur l'introduction d'approches exactes pour le problème de satisfiabilité diversifiée, lesquelles, à notre connaissance, n'ont pas encore été explorées dans la littérature. Nous proposons deux approches génériques, l'une reposant sur la programmation entière quadratique (QIP) et l'autre sur la programmation pseudo-booléenne linéaire, cette dernière pouvant être aisément transformée en une instance de satisfiabilité maximum (MaxSAT), l'extension naturelle de SAT en problème d'optimisation. Notre approche QIP permet de formuler la fonction objectif de manière naturelle sous forme quadratique, tandis que les approches linéaires reformulent l'objectif comme une combinaison linéaire d'une série de variables booléennes, s'appuyant sur deux fonctions de pondération : la pondération directe (DW) et la pondération incrémentale (IW). L'efficacité des approches proposées est ensuite évaluée au travers d'expérimentations approfondies.

La suite de cet article est structurée comme suit. La section 2 présente les définitions formelles, les notations ainsi que les fondements théoriques du problème de satisfiabilité diversifiée, en plus d'introduire les paradigmes standards de la programmation entière quadratique, la programmation pseudo-booléenne linéaire, et la satisfiabilité maximum. La section 3 expose nos approches exactes, incluant la formulation quadratique ainsi que les formulations pseudo-booléennes linéaires et leur transformation en modèles MaxSAT. La section 4 fournit une analyse détaillée des résultats expérimentaux obtenus. Enfin, la section 5 récapitule nos contributions et discute les perspectives de nos travaux.

2 Préliminaires

2.1 Satisfiabilité diversifiée

Soit X un ensemble de variables booléennes prenant des valeurs dans $\{Vrai, Faux\}$ (ou $\{0, 1\}$). Un littéral l est soit une variable $x \in X$, soit sa négation $\overline{x}$. Le littéral positif $l = x$ est dit satisfait lorsque x est affectée à *Vrai*, tandis que le littéral négatif $l = \overline{x}$ est satisfait lorsque x est affectée à *Faux*. Une clause C est une disjonction ($\vee$) de littéraux, et peut également être représentée comme un ensemble de littéraux. Plus précisément, on peut écrire un clause sous la forme $C = C^+ \cup C^-$, où C^+ est l'ensemble des littéraux positifs et C^- l'ensemble des littéraux négatifs. De plus, une clause est satisfaite dès lors que l'un de ses littéraux est satisfait. Une formule en forme normale conjonctive (CNF) est une conjonction ($\wedge$) de clauses, et elle est satisfaite si toutes ses clauses sont satisfaites. Une formule CNF ϕ composée de m clauses peut être représentée comme un ensemble de clauses, c'est-à-dire $\phi = \{C_1, \ldots, C_m\}$. Étant donnée une formule CNF ϕ définie sur les variables de X, le problème de satisfiabilité propositionnelle (SAT) consiste à déterminer s'il existe une affectation $\alpha : X \to \{Vrai, Faux\}$ qui satisfait ϕ. Lorsqu'une telle affectation existe, on l'appelle modèle de ϕ et on dit que la formule est satisfiable.

Nous introduisons ci-après de manière formelle la notion de distance de Hamming entre deux affectations, qui correspond au nombre de valeurs différentes attribuées aux variables booléennes dans une paire d'affectations. Nous définissons ensuite une notion essentielle à la formulation du problème de satisfiabilité diversifiée, à savoir la mesure de *diversité* pour des ensembles d'affectations.

Définition 1 (Distance de Hamming entre affectations)**.** *Étant donné un ensemble de variables booléennes X et deux affectations α_1, α_2 des variables de X, la distance de Hamming entre α_1 et α_2, notée $Dis(\alpha_1, \alpha_2)$, est définie comme suit :*

$$\begin{aligned} Dis(\alpha_1, \alpha_2) &= |\{x \in X \mid \alpha_1(x) \neq \alpha_2(x)\}| \\ &= \sum_{x \in X} |\alpha_1(x) - \alpha_2(x)| \end{aligned}$$

Définition 2 (Diversité)**.** *Étant donné un ensemble de variables booléennes X et un ensemble $A = \{\alpha_1, \ldots, \alpha_k\}$ de k affectations des variables dans X, la diversité de A, notée $Div(A)$, est définie comme suit :*

$$Div(A) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} Dis(\alpha_i, \alpha_j) \tag{1}$$

Étant donnée une formule CNF ϕ et un entier $k \geq 2$, le problème de la satisfiabilité diversifiée (Diverse satisfiability, ou Diverse SAT) consiste à trouver un ensemble de k modèles de ϕ avec une diversité maximale. Plus formellement, nous cherchons :

$$A = argmax_{\mathcal{A} \in \{0,1\}^{k \times |X|}} Div(\mathcal{A})$$

Dans [20], Nadel évoque la notion de diversité pour une variable, c'est-à-dire le nombre de ses assignations à *Vrai* multiplié par le nombre de ses assignations à *Faux*, telle que définie formellement ci-dessous. Cette notion lui a servi de critère pour évaluer la contribution d'une seule variable à la diversité, dans l'heuristique proposée par l'auteur. Nous formalisons la correspondance entre la diversité classique et celles des variables dans Proposition 1.

Définition 3 (Diversité des Variables)**.** *Étant donné un ensemble de variables booléennes X et un ensemble $A = \{\alpha_1, \ldots, \alpha_k\}$ de k affectations des variables de X, la diversité de la variable $x \in X$ dans A, notée $Div(x, A)$, est définie comme suit :*

$$Div(x, A) = T(x, A) \cdot F(x, A)$$

où :

— $T(x, A) = |\{\alpha \in A \mid \alpha(x) = 1\}|$,
— $F(x, A) = |\{\alpha \in A \mid \alpha(x) = 0\}|$.

Proposition 1. *Étant donné un ensemble de variables booléennes X et un ensemble $A = \{\alpha_1, \ldots, \alpha_k\}$ de k affectations des variables dans X, on a :*

$$Div(A) = \sum_{x \in X} Div(x, A) \tag{2}$$

Preuve.

$$\begin{aligned} Div(A) &= \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} Dis(\alpha_i, \alpha_j) \\ &= \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} |\{x \in X \mid \alpha_i(x) \neq \alpha_j(x)\}| \\ &= \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{\substack{x \in X \\ \alpha_i(x) \neq \alpha_j(x)}} 1 \\ &= \sum_{x \in X} \sum_{i=1}^{k-1} \sum_{\substack{j=i+1 \\ \alpha_i(x) \neq \alpha_j(x)}}^{k} 1 \\ &= \sum_{x \in X} T(x, A) \cdot F(x, A) = \sum_{x \in X} Div(x, A) \end{aligned}$$

□

Dans le reste de cet article, nous calculons la diversité des modèles à l'aide de l'équation 2 puisqu'elle permet de représenter plus naturellement le problème dans nos formulations.

2.2 Programmation quadratique en nombres entiers

La programmation quadratique en nombres entiers (Quadratic Integer Programming, QIP) est un paradigme de programmation mathématique avec des variables entières, une fonction objectif quadratique et des contraintes linéaires. Comme défini dans [22], la forme classique d'un modèle QIP est la suivante :

$$\begin{aligned} (QIP) \quad & \max_x \; x^\top H x + c^\top x \\ & \text{s.t. } x \in \mathcal{P} \\ & x \in \mathbb{Z}^n \end{aligned}$$

où $H \in \mathbb{Q}^{n \times n}$ est une matrice symétrique, $c \in \mathbb{Q}^n$ et $\mathcal{P}$ est un polyèdre, représenté par $\mathcal{P} = \{x : Ax \leq b\}$ où $A \in \mathbb{Q}^{m \times n}$ et $b \in \mathbb{Q}^m$.
La programmation pseudo-booléenne linéaire (Linear Pseudo-Boolean Programming, LPB) est définie dans le domaine des variables booléennes. Selon [10], elle peut être formulée sous le format suivant :

$$\begin{aligned} (LPB) \quad & \max f(x_1, \ldots, x_n) \\ & \text{s.t. } f_j(x_1, \ldots, x_n) \geqq 0, \quad j \in \{1, \ldots, m\} \\ & x \in \{0, 1\}^n \end{aligned}$$

où $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} w_i \cdot x_i$, $w_i \in \mathbb{Z}$.

2.3 Satisfiabilité maximum

La satisfiabilité maximum (MaxSAT) est l'extension naturelle de SAT en problème d'optimisation. Dans cet article, nous utilisons sa version la plus générique, à savoir MaxSAT partiel pondéré [4, 15]. Ce problème prend en entrée une formule bipartite pondérée $\phi = H \cup S$ contenant des variables booléennes dans X. H est l'ensemble des clauses dures qui doivent être satisfaites, comme dans SAT. S est l'ensemble des clauses souples, composé de clauses pondérées (C, W_C), où W_C est un poids entier positif associé à la clause C. L'objectif de MaxSAT partiel pondéré est d'obtenir une affectation α qui maximise (resp. minimise) la somme des poids des clauses souples satisfaites (resp. falsifiées) tout en satisfaisant toutes les clauses dures. Formellement, soit $cost_\alpha(\phi)$ la somme des poids des clauses souples de ϕ falsifiées par l'affectation α, le problème MaxSAT partiel pondéré consiste ainsi à déterminer $optimum(\phi) = \min_\alpha cost_\alpha(\phi)$.
Dans le reste de l'article, nous utilisons MaxSAT pour faire référence à sa variante partielle pondérée. De plus, la littérature introduit des contraintes pseudo-booléennes (PB) qui peuvent être efficacement encodées sous forme clausale [23]. Ces contraintes prennent la forme suivante :

$$\sum_j a_j \cdot l_j \rhd b$$

où $a_j \in \mathbb{N}$, $b \in \mathbb{N}$, l_j est un littéral et $\rhd \in \{=, \geq, \leq\}$. Un cas particulier de contrainte PB survient lorsque tous les coefficients des littéraux sont égaux à un et peuvent donc être omis [23]. Ces contraintes sont couramment appelées contraintes de cardinalité et prennent donc la forme la suivante :

$$\sum_j l_j \bowtie b$$

où l_j est un littéral, $b \in \mathbb{N}$ et $\bowtie \in \{=, \leq, \geq\}$.

3 Formulations pour la satisfiabilité diversifiée

Dans cette section, nous présentons trois formulations distinctes du problème de satisfiabilité diversifiée, à savoir une formulation basée sur un modèle de programmation quadratique en nombres entiers (*QP*) et deux formulations linéaires pseudo-booléennes (*DW* et *IW*). Nous commençons par la première formulation *QP*, présentée en 3.1, qui transforme naturellement le problème d'optimisation initial en une fonction objectif quadratique. En 3.2, nous exposons l'idée générale permettant de faire le lien entre une formulation avec variables entières et objectif quadratique, et une formulation avec variables booléennes et objectif linéaire, conduisant ainsi aux deux modèles linéaires proposés en 3.3. Enfin, la section 3.4 présente la transformation des formulations linéaires pseudo-booléennes vers des formulations MaxSAT.

3.1 Formulation quadratique

Dans cette sous-section, nous présentons une formulation en programmation quadratique en nombres entiers (QIP)

$$\max \sum_{j \in N} O_j \cdot (k - O_j) \qquad \text{(QP-0)}$$

s.c

$$\sum_{l_j \in C_h^-} (1 - V_{i,j}) + \sum_{l_j \in C_h^+} V_{i,j} \geq 1 \quad \forall (i,h) \in K \times M \qquad \text{(QP-1)}$$

$$O_j = \sum_{i=\{1,\dots,K\}} V_{i,j} \qquad \forall j \in N \qquad \text{(QP-2)}$$

$$O_j \in \mathbb{N}^+, V_{i,j} \in \{0,1\} \qquad \forall (i,j) \in K \times N \qquad \text{(QP-3)}$$

FIGURE 1 – Modèle quadratique pour la satisfiabilité diversifiée

pour le problème de satisfiabilité diversifiée. Cette formulation s'appuie naturellement sur la Proposition 1, qui exprime la diversité comme la somme des diversités individuelles des variables. Étant donnée une formule CNF ϕ contenant n variables et m clauses, notre objectif est d'obtenir k modèles maximisant la diversité. Pour cela, nous définissons un modèle QIP à l'aide des deux séries de variables suivantes, où $K = \{1, \dots, k\}$, $N = \{1, \dots, n\}$ et $M = \{1, \dots, m\}$:

- **Variables entières** $O_j \quad \forall j \in N$, qui représentent le nombre d'affectations à *Vrai* de la $j^{\text{ème}}$ variable dans l'ensemble A des k affectations, c'est à dire $T(x_j, A)$.
- **Variables binaire** $V_{i,j} \quad \forall (i,j) \in K \times N$, qui représentent la valeur booléenne de la $j^{\text{ème}}$ variable dans la $i^{\text{ème}}$ affectation $\alpha_i \in A$, c'est à dire $\alpha_i(x_j)$.

Nous obtenons la formulation quadratique (QP) illustrée dans la Figure 1. L'équation QP-0 est une reformulation de Proposition 1, indiquant que l'objectif de maximisation, à savoir la diversité, est égal à la somme des diversités individuelles des variables. L'équation QP-1 garantit que l'ensemble A des k affectations construites constitue bien des modèles de la formule CNF ϕ donnée en entrée. L'équation QP-2 établit le lien sémantique entre les variables booléennes $V_{i,j}$ et les variables entières O_j. Cette formulation requiert $O(k \cdot n)$ variables et $O(k \cdot m + n)$ contraintes.

3.2 De la formulation quadratique aux formulations linéaires

Dans la sous-section précédente, la formulation proposée utilise des variables entières ainsi qu'une fonction objectif quadratique, conformément à la relation établie dans Proposition 1. Toutefois, la présence d'un terme quadratique dans la fonction objectif peut constituer un obstacle important à la résolution efficace du problème. Par conséquent, nous proposons une approche alternative consistant à transformer la formulation QIP en une formulation linéaire pseudo-booléenne. Autrement dit, il s'agit de transformer la fonction objectif quadratique initiale en une combinaison linéaire de variables booléennes.

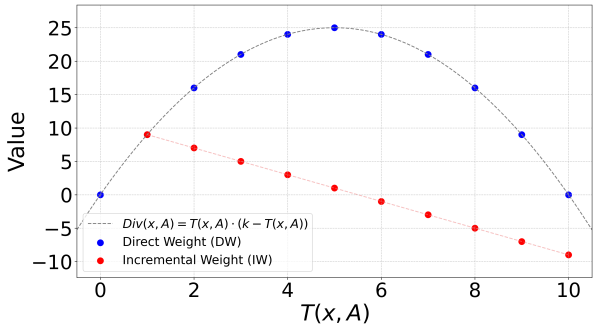

FIGURE 2 – Les fonctions de poids *DW* et *IW* pour un scénario avec $k = 10$

Pour chaque variable booléenne d'origine $x_j \in X$ dans la formule CNF ϕ donnée, nous introduisons d'abord une série de variables booléennes auxiliaires destinées à représenter le nombre d'affectations à *Vrai* de x_j à travers les k modèles, soit $T(x_j, A)$. Il convient de souligner que la relation de correspondance entre ces variables auxiliaires et l'entier $T(x_j, A)$ doit être cohérente avec les fonctions de pondération. Nous développons ensuite deux fonctions de pondération distinctes, appelées respectivement *pondération directe (DW)* et *pondération incrémentale (IW)*, permettant de reformuler le terme quadratique original $Div(x_j, A) = T(x_j, A) \cdot (k - T(x_j, A))$.

Une illustration de ces deux fonctions de pondération dans un scénario comportant $k = 10$ modèles est présentée dans la Figure 2. La première fonction, *DW*, associe directement à chaque valeur possible de $T(x_j, A)$ sa contribution correspondante à la diversité. Les points bleus répartis sur la courbe parabolique indiquent que le maximum de diversité est atteint lorsque x_j est assignée à *Vrai* dans exactement la moitié des modèles. La seconde fonction, *IW*, adopte une approche incrémentale en exprimant $Div(x_j, A)$ comme la somme de contributions marginales. Les points rouges de la Figure 2 illustrent la variation de la valeur de diversité lorsqu'on incrémente $T(x_j, A)$ d'une unité. Il est à noter que ces valeurs incrémentales deviennent négatives au-delà du point médian, traduisant les rendements décroissants associés à une répartition déséquilibrée des affectations à *Vrai*.

Formellement, soit $r = T(x_j, A)$, nous définissons la fonction de *pondération directe (DW)* comme suit :

$$\mathcal{D}_r = r \cdot (k - r) \quad \forall r \in \{0, \dots, k\}$$

et la fonction de *pondération incrémentale (IW)* par :

$$\mathcal{I}_r = D_r - D_{r-1} = -2r + k + 1 \quad \forall r \in \{1, \dots, k\}$$

On peut remarquer que pour tout $r \in \{1, \dots, k\}$, on a $\mathcal{D}r = \sum_{r'=1}^{r} \mathcal{I}_{r'}$. Ces deux fonctions de pondération distinctes offrent donc des approches différentes mais mathématiquement équivalentes pour reformuler la fonction objectif d'origine, comme cela sera détaillé dans la suite.

$$\max \sum_{j=1}^{n} \sum_{r=0}^{k} \mathcal{D}_r \cdot U_{j,r} \qquad \text{(DW-0)}$$

s.c

$$\sum_{l_j \in C_h^-} (1 - V_{i,j}) + \sum_{l_j \in C_h^+} V_{i,j} \geq 1 \qquad \forall (i,h) \in K \times M \qquad \text{(DW-1)}$$

$$\sum_{r \in (\{0\} \cup K)} U_{j,r} = 1 \qquad \forall j \in N \qquad \text{(DW-2)}$$

$$\sum_{r \in (\{0\} \cup K)} r U_{j,r} = \sum_{i \in K} V_{i,j} \qquad \forall j \in N \qquad \text{(DW-3)}$$

$$V_{i,j} \in \{0,1\}, U_{j,r} \in \{0,1\} \qquad \forall (i,j) \in K \times N \quad \forall (r,j) \in (\{0\} \cup K) \times N \qquad \text{(DW-4)}$$

FIGURE 3 – Modèle linéaire pour la satisfiabilité diversifiée avec le pondération directe *DW*.

3.3 Formulations linéaires

3.3.1 Formulation par pondération directe

Nous présentons d'abord la formulation pour le problème de satisfiabilité diversifiée avec la fonction de pondération *DW*, qui repose sur les variables suivantes, où $K = \{1, \ldots, k\}$ et $N = \{1, \ldots, n\}$:

— **Variables binaires** $U_{j,r} \quad \forall (r,j) \in (\{0\} \cup K) \times N$, qui représentent le fait que le nombre d'affectations à *Vrai* de la $j^{\text{ème}}$ variable dans l'ensemble des k modèles soit égal à r, c'est à dire $T(x_j, A) = r$.

— **Variables binaire** $V_{i,j} \quad \forall (i,j) \in K \times N$, qui représentent la valeur booléenne de la $j^{\text{ème}}$ variable dans la $i^{\text{ème}}$ affectation $\alpha_i \in A$, c'est à dire $\alpha_i(x_j)$.

Ainsi, la formulation avec la fonction de pondération *DW*, $\mathcal{D}_r = r \cdot (k - r)$, peut être obtenue comme indiqué dans la Figure 3, où $M = \{1, \ldots, m\}$. L'équation DW-0 formule la fonction objectif, où chaque terme $D_r \cdot U_{j,r}$ indique qu'il y a une contribution à la diversité $D_r = r \cdot (k - r)$ lorsque la variable x_j est assignée à *Vrai* exactement dans r des k modèles. L'équation DW-1 garantit que chaque modèle α_i satisfait la formule CNF originale ϕ, c'est-à-dire que pour toute clause C_h de ϕ et chaque modèle α_i, au moins un littéral de la clause doit être satisfait selon les assignations de variables dans α_i. L'équation DW-2 impose qu'il y ait un seul nombre d'assignations à *Vrai* de la variable j parmi les k modèles. L'équation DW-3 établit la relation de correspondance entre les variables $U_{j,r}$ et les assignations réelles des modèles $V_{i,j}$. Il y a $O(k \cdot n)$ variables et $O(k \cdot m + n)$ contraintes dans la formulation *DW*.

3.3.2 Formulation par pondération incrémentale

Dans cette sous-section, nous développons une autre formulation pour le problème de satisfiabilité diversifiée en utilisant la fonction *IW*. Les variables utilisées dans cette formulation sont les suivantes :

— **Variables binaires** $U_{j,r} \quad \forall (r,j) \in (\{0\} \cup K) \times N$, qui représentent le fait que le nombre d'affectations à *Vrai* de la $j^{\text{ème}}$ variable dans l'ensemble des k affectations A soit supérieure ou égale à r, c'est à dire $T(x_j, A) >= r$.

— **Variables binaire** $V_{i,j} \quad \forall (i,j) \in K \times N$, qui représentent la valeur booléenne de la $j^{\text{ème}}$ variable dans la $i^{\text{ème}}$ affectation $\alpha_i \in A$, c'est à dire $\alpha_i(x_j)$.

La formulation linéaire avec *IW*, définie par $\mathcal{I}_r = -2r + k + 1$, est obtenue comme illustré dans la Figure 4. La structure de *IW* reste similaire à celle de *DW*. L'équation IW-0 exprime la contribution incrémentale de l'affectation *Vrai* de la variable x_j à la diversité globale. L'équation IW-1 impose la satisfiabilité de l'ensemble des k affectations obtenues. L'équation IW-2 assure l'ordre dans l'utilisation des variables $U_{j,r}$ ce qui, en conjonction avec IW-3, permet de capturer l'information sémantique de $U_{j,r}$. Les complexités en nombre de variables et de contraintes dans la formulation *IW* sont respectivement de $O(n \cdot k)$ et $O(k \cdot (m + n))$.

$$\max \sum_{j=1}^{n} \sum_{r=0}^{k} \mathcal{I}_r \cdot U_{j,r} \qquad \text{(IW-0)}$$

s.c

$$\sum_{l_j \in C_h^-} (1 - V_{i,j}) + \sum_{l_j \in C_h^+} V_{i,j} \geq 1 \qquad \forall (i,h) \in K \times M \qquad \text{(IW-1)}$$

$$U_{j,r} \leq U_{j,r-1} \qquad \forall (r,j) \in (K \setminus \{1\}) \times N \qquad \text{(IW-2)}$$

$$\sum_{r \in K} U_{j,r} = \sum_{i \in K} V_{i,j} \qquad \forall j \in N \qquad \text{(IW-3)}$$

$$V_{i,j} \in \{0,1\}, U_{j,r} \in \{0,1\} \quad \forall (i,j) \in K \times N \quad \forall (r,j) \in K \times N \qquad \text{(IW-4)}$$

FIGURE 4 – Modèle linéaire pour la satisfiabilité diversifiée avec le pondération incrémentale *IW*.

3.4 Formulations MaxSAT

Lorsqu'on se focalise sur l'exigence intrinsèque du problème de satisfiabilité diversifiée, on ne peut pas ignorer la tâche centrale consistant à obtenir des modèles en logique propositionnelle, une tâche pour laquelle MaxSAT semble intuitivement être mieux adapté. Dans cette sous-section, nous présentons la transformation des formulations linéaires précédentes vers MaxSAT. Nous reformulons les modèles sous forme clausale, accompagnée de contraintes pseudo-booléennes (PB) ou de cardinalité, comme illustré dans la Figure 5. De plus, nous signalons que l'équation IW-0* peut contenir des poids négatifs, ce qui n'est pas conforme au standard original de MaxSAT. Elle est donc transformée à l'aide de l'équation suivante :

$$\begin{cases} (U_{j,i}, \mathcal{I}_r) & \mathcal{I}_r \geq 0, \\ (\overline{U}_{j,i}, -\mathcal{I}_r) & \mathcal{I}_r < 0, \end{cases} \qquad \forall (r,j) \in K \times N \qquad \text{(IW-0)}$$

Un exemple illustratif est présenté dans l'exemple suivant. Cette technique de normalisation permet de représenter correctement la fonction de poids incrémentale dans le cadre standard de MaxSAT, qui exige des poids non négatifs pour toutes les clauses souples.

Exemple 1. *Si on prend $k = 4$ and $n = 3$, selon la définition $\mathcal{I}_r = -2r + k + 1$, on a :*

$$\mathcal{I}_1 = 3, \mathcal{I}_2 = 1, \mathcal{I}_3 = -1, \mathcal{I}_4 = -3$$

Pour chaque variable x_j où $j \in \{1, 2, 3\}$, avant normalisation, les clause souples présentent des poids négatifs selon IW-0 :*

$$(U_{j,1}, 3), (U_{j,2}, 1), (U_{j,3}, -1), (U_{j,4}, -3)$$

La contribution des x_j peut être formulée comme suit :

$$obj_{original} = 3U_{j,1} + U_{j,2} - U_{j,3} - 3U_{j,4}$$

Après normalisation de IW-0, on obtient :*

$$(U_{j,1}, 3), (U_{j,2}, 1), (\overline{U}_{j,3}, 1), (\overline{U}_{j,4}, 3)$$

avec la contribution suivante :

$$obj_{normalized} = 3U_{j,1} + U_{j,2} + \overline{U}_{j,3} + 3\overline{U}_{j,4}$$

Puisque $\overline{U}_{j,r} = 1 - U_{j,r}$, on peut réecrire les objectifs comme suit :

$$\begin{aligned} obj_{normalized} &= 3U_{j,1} + U_{j,2} + (1 - U_{j,3}) + 3(1 - U_{j,4}) \\ &= 3U_{j,1} + U_{j,2} - U_{j,3} - 3U_{j,4} + 4 \\ &= obj_{original} + 4 \end{aligned}$$

L'ajout d'une constante à toutes les valeurs de la fonction objectif ne modifie pas la solution optimale d'une instance MaxSAT donc la formulation normalisée permet bien d'identifier les mêmes ensembles optimaux de modèles diversifiés.

4 Résultats expérimentaux

Dans cette section, nous présentons les résultats expérimentaux des approches proposées. Le protocole expérimental, incluant l'environnement de test, les instances traitées, ainsi que les solveurs utilisés, est détaillé en 4.1. La section 4.2 présente les performances globales, tandis que la section 4.3 compare les formulations basées sur les deux fonctions de pondération distinctes.

4.1 Protocole expérimental

Tous les tests ont été menés sur la plateforme MatriCS [1], équipée d'un système CentOS 8.6 et d'un processeur Intel Xeon E5-2680 v4 fonctionnant à une fréquence de 2,40 GHz, avec une capacité Turbo Boost allant jusqu'à 3,30 GHz. Le temps limite fixé pour tous les solveurs est de 7200 secondes. Les instances utilisées dans nos expériences proviennent de deux sources : 20 instances issues de la vérification de matériel, telles que proposées et utilisées dans [20], ainsi que 88 instances provenant de SATLIB [2], pour un total de 108 instances réparties en 8 familles distinctes. La taille de ces instances varie de 61 variables et 300 clauses

1. https://www.matrics.u-picardie.fr
2. https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

Soft Clauses

$(U_{j,r}, r \cdot (k - r))$ $\quad \forall (r, j) \in (\{0\} \cup K) \times N$ (DW-0)

Hard Clauses

$$(\bigvee_{l_j \in C_h^-} \overline{V}_{i,j}) \vee (\bigvee_{l_j \in C_h^+} V_{i,j}) \quad \forall (i, h) \in K \times M \quad \text{(DW-1)}$$

$$\sum_{r \in (\{0\} \cup K)} U_{j,r} = 1 \quad \forall j \in N \quad \text{(DW-2)}$$

$$\sum_{r \in (\{0\} \cup K)} r U_{j,r} = \sum_{i \in K} V_{i,j} \quad \forall j \in N \quad \text{(DW-3)}$$

(a) MaxSAT-DW

Soft Clauses

$(U_{j,r}, \mathcal{I}_r)$ $\quad \forall (r, j) \in K \times N$ (IW-0*)

Hard Clauses

$$(\bigvee_{l_j \in C_h^-} \overline{V}_{i,j}) \vee (\bigvee_{l_j \in C_h^+} V_{i,j}) \quad \forall (i, h) \in K \times M \quad \text{(IW-1)}$$

$$U_{j,r} \rightarrow U_{j,r-1} \quad \forall (r, j) \in (K \setminus \{1\}) \times N \quad \text{(IW-2)}$$

$$\sum_{r \in K} U_{j,r} = \sum_{i \in K} V_{i,j} \quad \forall j \in N \quad \text{(IW-3)}$$

(b) MaxSAT-IW

FIGURE 5 – Formulations MaxSAT avec *DW* (a) et *IW* (b).

à 83908 variables et 276106 clauses. Toutes les instances garantissent l'existence d'au moins 10 modèles distincts.
Les solveurs de l'état de l'art suivants ont été utilisés dans nos expériences afin d'évaluer les méthodologies proposées : CPLEX [11] pour la formulation QIP proposée en 3.1 ainsi que pour les deux formulations pseudo-booléennes linéaires introduites en 3.3 ; MaxHS [21], WMaxCDCL [14], et OpenWBO [18] pour les deux formulations MaxSAT décrites en 3.4. Nous notons que ces trois solveurs MaxSAT représentent respectivement des approches basées sur la programmation linéaire en nombres entiers (ILP), sur la séparation et évaluation (Branch-and-Bound), et sur les appels aux oracles SAT. Pour réaliser les encodages CNF des contraintes de cardinalité et pseudo-booléennes, nous avons utilisé la bibliothèque PySAT [3] [12, 13], avec les contraintes de cardinalité encodées à l'aide de l'encodage *Cardinality Network* [3], qui permet d'encoder une contrainte de cardinalité de borne $\mathcal{K}$ sur $\mathcal{N}$ littéraux avec une complexité $O(\mathcal{N} \log^2 \mathcal{K})$ en termes de clauses et de variables auxiliaires. Concernant les contraintes pseudo-booléennes, nous avons utilisé le mode "Best" proposé par PySAT, qui sélectionne automatiquement la méthode d'encodage la plus adaptée.
Pour justifier la non-trivialité des solutions obtenues, nous avons également mis en place une approche naïve par énumération, basée sur le solveur SAT CaDiCaL [5]. Le solveur est exécuté de manière itérative k fois. Après chaque

3. https://pysathq.github.io/

itération, une nouvelle clause est ajoutée à la formule pour bloquer l'assignation satisfaisante nouvellement trouvée. Une fois les k itérations terminées, le programme est suspendu et la diversité est calculée à partir des k modèles obtenus.

4.2 Performances globales

Les analyses de performance des différentes approches testées à travers l'ensemble des familles, pour les valeurs de k égales à 2, 5 et 10, sont présentées dans les Tableaux 1, 2 et 3. Ces tableaux présentent les performances sous deux aspects : la capacité des approches proposées à résoudre le problème à l'optimalité et la diversité obtenue par celles-ci, comparée à l'énumérateur naïf CaDiCaL. À partir de ces tableaux, il ressort que toutes les formulations proposées parviennent à obtenir des solutions optimales dans certaines familles, notamment pour des valeurs plus petites de k, comme celle de 2. De manière générale, les formulations linéaires avec les fonctions de poids *DW* et *IW*, obtiennent le plus grand nombre de solutions optimales pour toutes les valeurs de k. L'écart se renforce lorsque la valeur de k augmente jusqu'à 10, illustrant ainsi une bonne scalabilité de l'approche linéaire pour la résolution du problème.

Pour les solveurs MaxSAT, OpenWBO coopérant avec *DW* obtient le plus grand nombre d'instances résolues à l'optimalité, soit 89 sur 108, lorsque $k = 2$, tandis que MaxHS coopérant avec *IW* bascule en première position lorsque $k = 5$ et $k = 10$, résolvant respectivement 67 et 33 instances. WMaxCDCL présente des performances comparables à celles de MaxHS et OpenWBO lorsque $k = 2$ et $k = 10$, bien que ses performances diffèrent des deux autres lorsque $k = 10$.

De plus, on note que, dans plusieurs cas, bien que la formulation QIP avec CPLEX parvienne à atteindre la valeur optimale prouvée par certains autres solveurs, elle échoue à prouver l'optimalité, avec respectivement 7, 21, et 15 instances non prouvées lorsqu'on prend $k = 2$, $k = 5$ et $k = 10$. Dans l'ensemble, ces résultats démontrent que, malgré le fait que toutes les approches proposées sont capables de résoudre de manière optimale le problème de satisfiabilité diversifiée, le choix du solveur et de la fonction de poids a un impact significatif sur les performances, certaines combinaisons montrant des résultats supérieurs en fonction des cas spécifiques.

En termes de diversité, pour $k = 2, 5, 10$, les meilleurs résultats obtenus par un solveur exact est respectivement de 3.58, 2.50, et 3.34 fois meilleure que celle de la solution obtenue par l'énumération naïve avec CaDiCaL. Lorsqu'on restreint l'analyse aux cas avec optimalité prouvée, ces facteurs deviennent respectivement de 113, 63, et 23. L'écart de performance substantiel entre l'énumérateur naïf et les approches exactes proposées démontre clairement la difficulté inhérente du problème, et confirme la nécessité de développer des approches exactes performantes.

On peut également observer qu'avec l'augmentation de la valeur de k, les approches exactes proposées échouent parfois à obtenir des solutions faisables dans le temps imparti. Cela est particulièrement le cas pour WMaxCDCL et CPLEX et pourrait être dû au fait que l'objectif interne des solveurs exacts est d'obtenir des solutions optimales, de sorte que les résultats intermédiaires incomplets ne reflètent pas pleinement le processus de résolution. Par exemple, l'approche linéaire avec *IW* résolue avec le solveur CPLEX ne parvient à obtenir que 4 résultats faisables pour la famille difficile "hardware" lorsque $k = 5$, mais tous ces résultats sont optimaux. Malgré cette difficulté, les approches proposées offrent toujours une qualité satisfaisante pour les solutions sous-optimales, ce qui démontre une grande robustesse.

En se focalisant sur la comparaison de la capacité de résolution optimale de toutes les approches exactes proposées, les Figures 6, 7, 8 offrent une vision plus complète du nombre cumulé d'instances résolues par chaque approche dans le temps imparti.

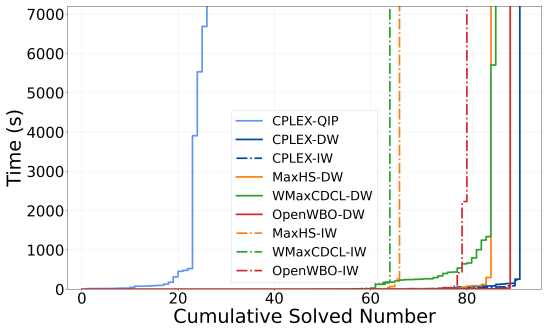

FIGURE 6 – Nombre cumulé d'instances résolues pour $k = 2$ en fonction du temps de résolution en secondes.

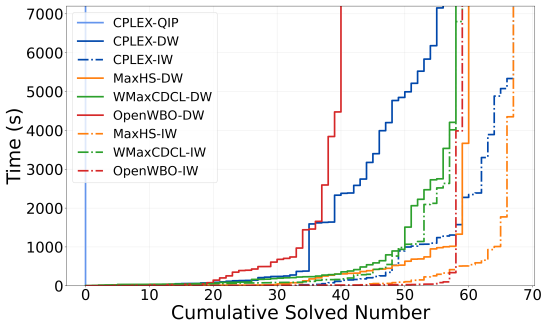

FIGURE 7 – Nombre cumulé d'instances résolues pour $k = 5$ en fonction du temps de résolution en secondes.

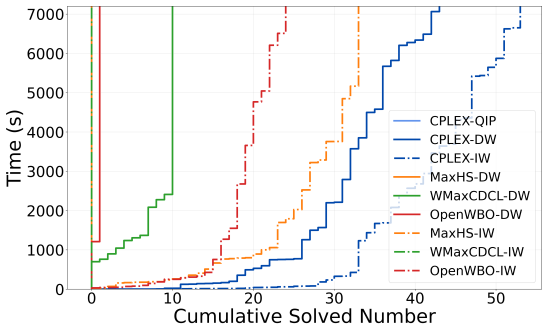

FIGURE 8 – Nombre cumulé d'instances résolues pour $k = 10$ en fonction du temps de résolution en secondes.

Famille	#I	#Var	#CL	Énum.	CPLEX			MaxSAT (DW)			MaxSAT (IW)		
				CaDi.	QIP	DW	IW	MaxHS	WMaxCDCL	OpenWBO	MaxHS	WMaxCDCL	OpenWBO
flat100-	30	155	562	[2]	23(6686)[73]	30(11)[103]	30(8)[103]	30(2)[103]	30(15)[103]	**30(2)[103]**	30(5)[103]	30(16)[103]	30(6)[103]
flat100+	30	425	1584	[2]	1(6685)[63]	30(1047)[283]	30(730)[283]	30(486)[283]	30(357)[283]	**30(50)[283]**	30(81)[283]	30(435)[283]	25(51)[235]
ais	4	155	2730	[18]	1(4)[6]	4(196)[34]	4(60)[34]	3(56)[34]	4(5831)[34]	**4(35)[34]**	3(69)[34]	3(171)[22]	3(425)[22]
morphed	20	500	3100	[2]	0(-)[0]	20(65)[200]	**20(25)[200]**	19(166)[200]	20(9836)[200]	19(21)[199]	1(8)[200]	0(-)[0]	18(46)[197]
logistics	4	1881	11682	[3]	1(5531)[216]	**3(58)[365]**	3(108)[368]	3(286)[364]	2(443)[218]	2(13)[293]	2(304)[366]	1(28)[218]	2(2302)[285]
hardware	20	49391	161548	[5483]	0(-)[596]	4(249)[1177]	4(195)[1180]	0(-)[18835]	0(-)[0]	**4(66)[10052]**	0(-)[18381]	0(-)[0]	2(74)[9149]
All	108	9476	31620	[1018]	26(18905)[156]	91(1626)[377]	**91(1126)[378]**	85(996)[3647]	86(16482)[154]	89(187)[2018]	66(466)[3563]	64(651)[116]	80(2904)[1836]

TABLE 1 – Résultats pour $k = 2$. Pour chaque famille d'instances, le nombre d'instances ($\#I$), le nombre moyen de variables ($\#Var$) et le nombre moyen de clauses ($\#CL$) sont indiqués. Ensuite, la diversité des résultats obtenue par l'approche d'énumération naïve utilisant CaDiCaL (CaDi.) est rapportée. Les performances de l'approche QIP et de l'approche linéaire avec CPLEX ainsi que les trois solveurs MaxSAT, à savoir MaxHS, WMaxCDCL et OpenWBO, sur les formulations avec deux fonctions de poids distinctes (*DW*, *IW*) sont présentées sous la forme de $\#OPT(tot\_time_{opt})[Div_{all}]$, où $\#OPT$ représente le nombre de solutions optimales obtenues dans le temps limite, tot_time_{opt} est la somme des temps de résolution pour ces solutions optimales et Div_{all} est la diversité moyenne calculée en fonction des meilleures solutions faisables (ou optimales) obtenues pour pour toutes les instances dans la famille concernée. Dans le cas où aucune solution faisable n'est trouvée dans le temps limite, la diversité est considérée comme étant nulle (égale à 0 comme pénalité) sinon la diversité de la meilleure solution faisable est considérée. Parmi toutes nos approches, celle ayant résolu de manière optimale le plus grand nombre d'instances dans le temps le plus court est indiquée en gras. De plus, la meilleure diversité globale Div_{all} parmi toutes les approches, y compris l'énumération naïve avec CaDiCaL, est soulignée.

Famille	#I	#Var	#CL	Énum.	CPLEX			MaxSAT (DW)			MaxSAT (IW)		
				CaDi.	QIP	DW	IW	MaxHS	WMaxCDCL	OpenWBO	MaxHS	WMaxCDCL	OpenWBO
flat100-	30	155	562	[50]	0(-)[595]	30(2169)[827]	**30(81)[827]**	30(3558)[827]	30(5311)[827]	20(7597)[817]	30(609)[827]	30(4472)[827]	28(403)[823]
flat100+	30	425	1584	[32]	0(-)[59]	22(56712)[1493]	29(26379)[2160]	29(13603)[2249]	27(25914)[1973]	20(15181)[2148]	30(3023)[2267]	28(25118)[2067]	**30(796)[2267]**
ais	4	155	2730	[280]	0(-)[50]	1(5834)[119]	1(2348)[124]	**1(429)[314]**	1(645)[124]	0(-)[304]	1(506)[327]	1(765)[124]	1(3987)[306]
morphed	20	500	3100	[30]	0(-)[87]	0(-)[1977]	3(10458)[1985]	0(-)[1855]	0(-)[0]	0(-)[1197]	**6(9311)[2000]**	0(-)[0]	0(-)[2000]
logistics	4	1881	11682	[38]	0(-)[0]	0(-)[995]	0(-)[1608]	0(-)[2604]	0(-)[0]	0(-)[1682]	0(-)[2652]	0(-)[0]	0(-)[1782]
hardware	20	49391	161548	[42853]	0(-)[1919]	3(16991)[3850]	**4(4427)[3850]**	0(-)[82752]	0(-)[0]	0(-)[87145]	0(-)[100503]	0(-)[0]	0(-)[8613]
All	108	9476	31620	[7976]	0(-)[555]	56(81706)[1765]	67(43693)[1974]	60(17589)[16630]	58(31870)[782]	40(22779)[17257]	**67(13449)[19952]**	59(30354)[808]	59(5185)[2901]

TABLE 2 – Résultats pour $k = 5$. Le format des données est aligné avec Tableau 1.

Famille	#I	#Var	#CL	Énum.	CPLEX			MaxSAT (DW)			MaxSAT (IW)		
				CaDi.	QIP	DW	IW	MaxHS	WMaxCDCL	OpenWBO	MaxHS	WMaxCDCL	OpenWBO
flat100-	30	155	562	[337]	0(-)[1501]	30(14651)[3410]	**30(1253)[3410]**	0(-)[3224]	10(14085)[1815]	1(1209)[2897]	27(21543)[3410]	0(-)[2144]	17(31749)[3410]
flat100+	30	425	1584	[179]	0(-)[20]	5(18333)[1487]	**16(43420)[3905]**	0(-)[8323]	0(-)[0]	0(-)[8286]	6(20519)[9350]	0(-)[1531]	7(3270)[9350]
ais	4	155	2730	[1226]	0(-)[0]	0(-)[211]	0(-)[211]	0(-)[1272]	0(-)[0]	0(-)[1230]	0(-)[1366]	0(-)[0]	0(-)[1278]
morphed	20	500	3100	[168]	0(-)[0]	**8(45029)[7988]**	7(31263)[7993]	0(-)[6548]	0(-)[0]	0(-)[5106]	0(-)[6371]	0(-)[0]	0(-)[8000]
logistics	4	1881	11682	[208]	0(-)[0]	0(-)[0]	0(-)[1942]	0(-)[9230]	0(-)[0]	0(-)[7772]	0(-)[9975]	0(-)[0]	0(-)[7760]
hardware	20	49391	161548	[105280]	0(-)[0]	0(-)[14790]	0(-)[15886]	0(-)[330172]	0(-)[0]	0(-)[34762]	0(-)[42646]	0(-)[0]	0(-)[21896]
All	108	9476	31620	[19724]	0(-)[423]	43(78013)[5586]	**53(75936)[6534]**	0(-)[65952]	10(14085)[504]	1(1209)[10823]	33(42062)[13042]	0(-)[1021]	24(35019)[9416]

TABLE 3 – Résultats pour $k = 10$. Le format des données est aligné avec Tableau 1.

Lorsque $k = 2$, *DW* surpasse généralement *IW*, à l'exception du seul cas où CPLEX-*DW* et CPLEX-*IW* parviennent à résoudre rapidement toutes les instances. À mesure que k augmente à 5 et 10, *IW* montre généralement une meilleure performance que le poids direct. Par ailleurs, on observe que la formulation quadratique (QIP) montre des performances plus faibles par rapport aux approches linéaires. De plus, l'approche linéaire avec CPLEX se comporte de manière comparable aux solveurs MaxSAT uniquement lorsque $k = 5$, mais elle présente un avantage dans tous les autres cas. Ces résultats confirment que nos approches exactes peuvent résoudre efficacement les problèmes de satisfiabilité diversifiée de taille raisonnable, les approches linéaires montrant des résultats particulièrement prometteurs.

4.3 Poids direct vs poids incrémental

Pour analyser les différences entre les deux fonctions de poids proposées, *DW* et *IW*, nous menons une comparaison détaillée de leurs performances à travers différentes familles d'instances et configurations de solveurs. Les Figures 9 et 10 présentent les comparaisons des temps d'exécution des deux fonctions de poids pour $k = 2$ et $k = 5$. Chaque point dans les diagrammes de dispersion correspond à une instance, avec ses coordonnées déterminées par le temps d'exécution pour les deux fonctions de poids. Lorsqu'une instance n'a pas été résolue dans le temps imparti, la limite du temps d'exécution est reportée. Les points situés en dessous de la diagonale indiquent les instances où *IW* réalise de meilleures performances, tandis que les points au-dessus de la diagonale sont en faveur de *DW*.

Nous observons que les distributions des points suivent des tendances distinctes lorsque la valeur de k varie. Lorsque $k = 2$, *DW* montre généralement des performances supérieures, avec 33.33%, 62.96%, 95.37%, 92.59% des points situés au-dessus ou sur la ligne diagonale respectivement pour les solveurs CPLEX, MaxHS, WMaxCDCL, OpenWBO. À mesure que k augmente à 5, nous observons une tendance où *IW* surpasse *DW*, avec des pourcentages de points de 84.26%, 89.81%, 81.48%, 72.22% situés sous ou sur la ligne diagonale respectivement pour les solveurs CPLEX, MaxHS, WMaxCDCL et OpenWBO. Dans le cas de $k = 10$, nous omettons les figures correspondantes car toutes les approches, à l'exception des modèles linéaires avec CPLEX, n'ont pas réussi à résoudre de nombreuses instances, mais nous observons clairement dans le Tableau 3 que *IW* obtient les meilleurs résultats. Enfin, sur la base des résultats obtenus pour toutes les valeurs de k, nous constatons l'avantage de *IW* par rapport à *DW* à mesure que k augmente.

5 Conclusion

Dans cette étude, nous avons proposé des formulations exactes pour le problème de la satisfiabilité diversifiée, en utilisant la programmation entière quadratique (QIP) ainsi que des modèles de programmation linéaire pseudo-booléene, pouvant être adaptées en formulations MaxSAT. Ces approches permettent de formuler l'objectif de la satisfiabilité diversifiée sous l'angle de la diversité des variables. L'approche QIP présente une formulation naturelle de l'objectif sous forme quadratique, tandis que les formulations linéaires et MaxSAT reposent sur deux fonctions de poids distinctes, exploitant pleinement les capacités des solveurs modernes à résoudre des problèmes d'optimisation linéaire complexes. Les résultats expérimentaux, menés sur un large éventail d'instances avec différentes valeurs du paramètre k, représentant le nombre de modèles, démontrent l'efficacité et la robustesse des méthodologies proposées.

Dans le cadre de nos travaux futurs, nous envisageons de concevoir des algorithmes dédiés pour la satisfiabilité diversifiée qui dépassent les limitations des schémas de résolution globaux tout en conservant la structure et la représentation de l'objectif proposées dans nos approches actuelles. Par ailleurs, à mesure que nous élargissons nos expériences à un éventail d'instances plus large, nous espérons établir de nouvelles solutions optimales, afin de soutenir les chercheurs intéressés par ce domaine. Ces repères élargis faciliteront également la conception et l'évaluation d'algorithmes heuristiques et d'approximation pour le problème de la satisfiabilité diversifiée.

Remerciements

Ce travail est partiellement soutenu par le projet ANR-24-CE23-6126 (BforSAT) et par la chaire IA ANR-19-CHIA-0013-01 (MASSAL'IA) co-financée par l'agence nationale de recherche et le gestionnaire du réseau de distribution d'électricité Enedis. Il a bénéficié d'un accès aux ressources HPC de la « Plateforme MatriCS » de l'Université de Picardie Jules Verne qui est cofinancée par l'Union Européenne avec le Fonds Européen de Développement Régional (FEDER) et le Conseil Régional des Hauts-De-France entre autres.

Références

[1] S. Agbaria, D. Carmi, O. Cohen, D. Korchemny, M. Lifshits, and A. Nadel. Sat-based semiformal verification of hardware. In R. Bloem and N. Sharygina, editors, *FMCAD 2010*, pages 25–32. IEEE, 2010.

[2] O. Angelsmark and J. Thapper. Algorithms for the maximum hamming distance problem. In B. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *CSCLP 2004*, volume 3419 of *LNCS*, pages 128–141. Springer, 2004.

[3] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks and their applications. In O. Kullmann, editor, *SAT 2009*, volume 5584 of *LNCS*, pages 167–180. Springer, 2009.

[4] F. Bacchus, M. Järvisalo, and R. Martins. Maximum satisfiabiliy. In *Handbook of satisfiability*, pages 929–991. IOS Press, 2021.

[5] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

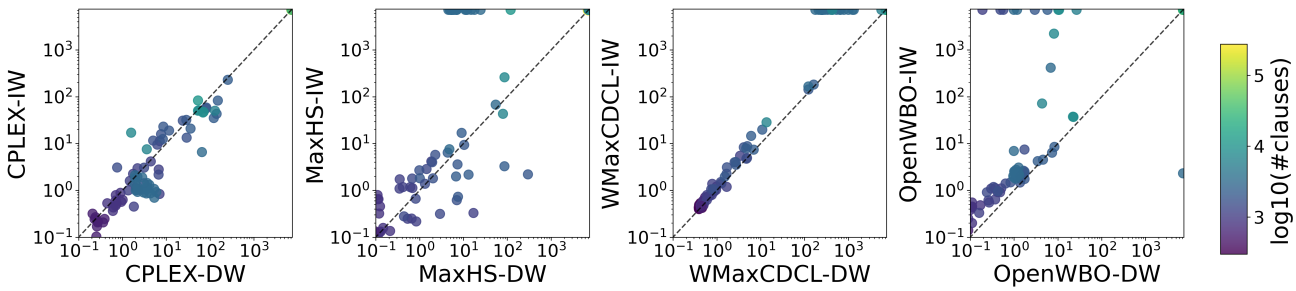

FIGURE 9 – Comparaison des temps d'exécution en échelle logarithmique des deux fonctions de poids pour $k = 2$.

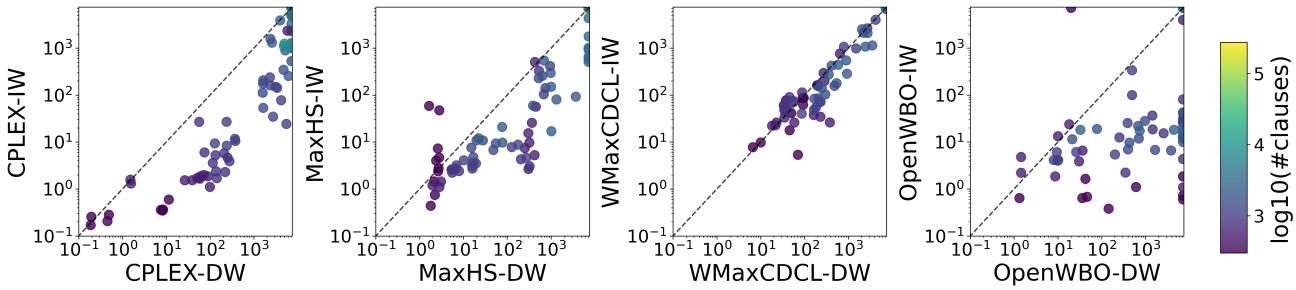

FIGURE 10 – Comparaison des temps d'exécution en échelle logarithmique des deux fonctions de poids pour $k = 5$.

[6] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021.

[7] E. M. Clarke, D. Kroening, and F. Lerda. A tool for checking ANSI-C programs. In K. Jensen and A. Podelski, editors, *TACAS 2004*, volume 2988 of *LNCS*, pages 168–176. Springer, 2004.

[8] S. A. Cook. The complexity of theorem-proving procedures. In M. A. Harrison, R. B. Banerji, and J. D. Ullman, editors, *STOC 1971*, pages 151–158. ACM, 1971.

[9] P. Crescenzi and G. Rossi. On the hamming distance of constraint satisfaction problems. *Theor. Comput. Sci.*, 288(1) :85–100, 2002.

[10] P. Hammer Ivànescu and S. Rudeanu. Boolean methods in operations research and related areas. *Oekonometrie und Unternehmensforschung (*, 7, 1968.

[11] IBM Corporation. *IBM ILOG CPLEX Optimization Studio*, 2022.

[12] A. Ignatiev, A. Morgado, and J. Marques-Silva. PySAT : A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018.

[13] A. Ignatiev, Z. L. Tan, and C. Karamanos. Towards universally accessible SAT technology. In *SAT*, pages 4 :1–4 :11, 2024.

[14] C. Li, Z. Xu, J. Coll, F. Manyà, D. Habet, and K. He. Boosting branch-and-bound maxsat solvers with clause learning. *AI Commun.*, 35(2) :131–151, 2022.

[15] C. M. Li and F. Manyà. MaxSAT, Hard and Soft Constraints. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 903–927. IOS Press, 2021.

[16] I. Lynce and J. Marques-Silva. SAT in bioinformatics : Making the case with haplotype inference. In A. Biere and C. P. Gomes, editors, *SAT 2006*, volume 4121 of *LNCS*, pages 136–141. Springer, 2006.

[17] P. Manolios, M. G. Oms, and S. O. Valls. Checking pedigree consistency with PCS. In O. Grumberg and M. Huth, editors, *TACAS 2007*, volume 4424 of *LNCS*, pages 339–342. Springer, 2007.

[18] R. Martins, V. Manquinho, and I. Lynce. Open-wbo : A modular maxsat solver,. In C. Sinz and U. Egly, editors, *SAT 2014*, volume 8561 of *LNCS*, pages 438–445. Springer, 2014.

[19] N. Misra, H. Mittal, and A. Rai. On the parameterized complexity of diverse sat. *arXiv preprint arXiv :2412.09717*, 2024.

[20] A. Nadel. Generating diverse solutions in SAT. In K. A. Sakallah and L. Simon, editors, *SAT 2011*, volume 6695 of *LNCS*, pages 287–301. Springer, 2011.

[21] A. Niskanen, J. Berg, and M. Järvisalo. Maxhs in maxsat evaluation 2022. *MaxSAT Evaluation 2022*, page 35, 2022.

[22] A. D. Pia, S. S. Dey, and M. Molinaro. Mixed-integer quadratic programming is in np. *Mathematical Programming*, 162 :225–240, 2017.

[23] O. Roussel and V. Manquinho. Pseudo-boolean and cardinality constraints. In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1087–1129. IOS Press, 2021.

[24] P. Surynek. Lazy compilation of variants of multi-robot path planning with satisfiability modulo theory (SMT) approach. In *IROS 2019*, pages 3282–3287. IEEE, 2019.

[25] H. Xu, R. A. Rutenbar, and K. A. Sakallah. sub-sat : a formulation for relaxed boolean satisfiability with applications in routing. In S. S. Sapatnekar and M. Pedram, editors, *ISPD 2002*, pages 182–187. ACM, 2002.