Prétraitement PLNE pour la simplification des instances MaxSAT

Jialu Zhang[1], Chu-Min Li[1], Sami Cherif[1], Shuolin Li[2], Zhifei Zheng[1]

[1] Laboratoire MIS UR 4290, Université de Picardie Jules Verne, Amiens, France
[2] Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

{jialu.zhang;chu-min.li;sami.cherif;zhifei.zheng}@u-picardie.fr;shuolin.li@lis-lab.fr

Résumé

Le problème de satisfiabilité maximum (MaxSAT) constitue un défi d'optimisation majeur avec de nombreuses applications du monde réel. Cet article propose une approche qui intègre des techniques de prétraitement issues de la programmation linéaire en nombres entiers (PLNE) dans les solveurs MaxSAT. Les résultats expérimentaux montrent que ce prétraitement améliore l'efficacité des solveurs MaxSAT modernes sur les récents benchmarks de l'évaluation MaxSAT. Une analyse approfondie révèle que cette méthode peut réduire le nombre de variables d'environ 40% et le nombre de clauses d'environ 60% pour certaines instances MaxSAT. En particulier, elle est particulièrement efficace sur la famille d'instances « extension enforcement », permettant au solveur WMaxCDCL de réduire le temps CPU de plus de 65% lors de la résolution de ces cas.

Mots-clés

Satisfiabilité Maximum, PLNE, prétraitement

Abstract

The Maximum Satisfiability problem (MaxSAT) is a major optimization challenge with numerous practical applications. This paper proposes an approach that integrates preprocessing techniques from Integer Linear Programming (ILP) into MaxSAT solvers. Experimental results show that this preprocessing enhances the efficiency of modern MaxSAT solvers on recent MaxSAT evaluation benchmarks. A detailed analysis reveals that this method can reduce the number of variables by approximately 40% and the number of clauses by around 60% for certain MaxSAT instances. Notably, it proves particularly effective for the "extension enforcement" instance family, enabling the WMaxCDCL solver to reduce CPU time by more than 65% when solving these instances.

Keywords

Maximum Satisfiability, ILP, Preprocessing.

1 Introduction

La satisfiabilité maximum (MaxSAT) constitue une extension naturelle de la satisfiabilité propositionnelle (SAT) en problème d'optimisation [4]. Pour une formule donnée en forme normale conjonctive (CNF), SAT consiste à vérifier s'il existe une interprétation des variables qui satisfait toutes les contraintes clausales de la formule, Dans MaxSAT, l'objectif est de trouver le nombre maximum de clauses pouvant être satisfaites dans la formule. De nombreux problèmes d'optimisation du monde réel peuvent être modélisés sous forme d'instances MaxSAT, notamment dans les domaines de la planification et de l'ordonnancement [5, 6, 15] ou encore de la vérification matérielle et logicielle [12], entre autres. Malheureusement, MaxSAT est plus difficile à résoudre que SAT, tant en théorie qu'en pratique, car il est plus complexe de trouver et de prouver l'optimalité des solutions [7].

Les algorithmes de résolution du problème MaxSAT peuvent être globalement classés en deux catégories : les algorithmes exacts et les algorithmes heuristiques. Les algorithmes exacts, tels que ceux de type séparation (BnB) et évaluation et la programmation linéaire en nombres entiers (PLNE), visent à trouver la solution optimale tout en démontrant son optimalité. En revanche, les algorithmes heuristiques, comme la recherche locale ou le recuit simulé, sont généralement plus rapides, mais ne garantissent pas l'optimalité [7, 8]. Il est reconnu que les solveurs PLNE, bien qu'efficaces sur certaines familles d'instances, ne sont pas compétitifs pour la majorité des instances MaxSAT (industrielles ou aléatoires) [2]. Ainsi, la pratique courante observée dans les évaluations MaxSAT[1] récentes, en particulier pour les solveurs les plus performants, consiste à combiner les solveurs PLNE dans un portfolio avec d'autres types de solveurs.

À titre d'exemple, dans la dernière évaluation MaxSAT [3], où la limite de temps totale est de 3600 secondes par instance, le solveur EvalMaxSAT exécute d'abord le solveur PLNE SCIP [1] pendant 400 secondes, puis un solveur basé sur l'algorithme OLL [9] pendant 3200 secondes. De même, UWrMaxSat [11] alterne l'exécution de SCIP et de son propre algorithme MaxSAT avec des limites de temps possiblement différentes, en comparant les bornes supérieures et inférieures obtenues pour les améliorer mutuellement. Ainsi, dans ces solveurs hybrides tirant parti de PLNE, le solveur PLNE est généralement utilisé de manière indépendante dans un cadre de portfolio, nécessitant un réglage heuristique minutieux, notamment sur la gestion des durées d'exécution.

[1] https://maxsat-evaluations.github.io/

Dans cet article, contrairement aux méthodes existantes, nous proposons une approche unifiée dans laquelle la PLNE est entièrement intégrée au processus de résolution en tant qu'étape de prétraitement. Notre approche consiste d'abord à lire la formule CNF pour la convertir en un ensemble de contraintes linéaires entières avec une fonction objectif. Ensuite, un solveur PLNE est appelé pour simplifier le problème, avant que les contraintes linéaires simplifiées ne soient retransformées en forme clausale afin d'être résolues par un solveur MaxSAT. Les résultats expérimentaux montrent que cette approche permet de résoudre un plus grand nombre d'instances que les solveurs MaxSAT de l'état de l'art. De plus, notre méthode ne nécessite pas de définir des limites de temps heuristiques pour les solveurs PLNE, qui continuent à s'exécuter jusqu'à la fin du prétraitement, contrairement à d'autres approches telles que EvalMaxSAT ou UWrMaxSat.

La suite de ce papier est organisée comme suit. La Section 2 introduit formellement le problème MaxSAT et présente brièvement les techniques de prétraitement en PLNE. La Section 3 décrit notre méthode d'intégration des techniques de prétraitement PLNE dans un solveur MaxSAT. La Section 4 présente les résultats expérimentaux. Enfin, on conlut et on discute les perceptives de nos travaux dans la Section 5.

2 Préliminaires

2.1 Satisfiabilité Maximum

Étant donné un ensemble de variables propositionnelles $V = \{x_1, x_2, \ldots, x_n\}$, un littéral l est soit une variable x, soit sa négation $\neg x$. Une clause c est une disjonction de k littéraux $l_1 \vee \ldots \vee l_k$, et peut être représentée comme un ensemble de littéraux. Une formule F en forme normale conjonctive (CNF) est une conjonction de m clauses $c_1 \wedge \ldots \wedge c_m$. Une variable x est dite affectée si elle prend la valeur 0 (*false*) ou 1 (*true*). Un littéral x est évalué à vrai (faux) si la variable x est affectée à 1 (0). Pour $\neg x$, c'est l'inverse. Une clause c est satisfaite si l'un de ses k littéraux est évalué à vrai. Une formule F est satisfaite uniquement lorsque toutes ses clauses le sont. Une affectation est dite *complète* si toutes les variables sont affectées, sinon elle est dite *partielle*. Le problème de satisfiabilité propositionnelle (SAT) consiste à trouver une affectation complète qui satisfait une formule CNF donnée [4].

MaxSAT est une extension naturelle du problème SAT en problème d'optimisation, et consiste à trouver une affectation qui maximise le nombre de clause satisfaites dans une formule CNF donnée. Dans la suite, on emploie le terme *MaxSAT* pour désigner la version la plus générique de ce problème: MaxSAT partiel pondéré [7]. MaxSAT partiel divise les clauses en deux catégories : les clauses *dures* et les clauses *souples*, c'est-à-dire $F = H \cup S$. L'objectif est de trouver une affectation qui satisfait toutes les clauses dures H tout en maximisant le nombre de clauses souples satisfaites dans S. Dans MaxSAT partiel pondéré, chaque clause souple $c \in S$ est associée à un poids entier positif w_c. L'objectif consiste alors à trouver une affectation optimale qui maximise la somme des poids des clauses souples satisfaites dans S, tout en satisfaisant l'ensemble des clauses dures dans H.

2.2 Techniques de prétraitement

Les solveurs PLNE utilisent des techniques de prétraitement afin de transformer un modèle donné en un modèle équivalent, mais potentiellement plus facile à résoudre. Ces techniques incluent la fixation de variables, l'agrégation de variables, l'élimination de contraintes redondantes, ainsi que d'autres mécanismes d'inférence avancés [13]. La technique de fixation de variables repose sur un algorithme de *probing* qui attribue temporairement la valeur 0 ou 1 à une variable binaire, puis propage les implications qui en découlent. L'agrégation de variables exploite les équations et les relations de contraintes présentes dans le modèle, ainsi que des algorithmes de détection de clusters, afin de regrouper plusieurs variables en une seule. Par ailleurs, l'élimination des contraintes redondantes permet de supprimer les contraintes qui sont implicitement satisfaites par d'autres, simplifiant ainsi le modèle [1].

Des techniques de prétraitement sont également utilisées dans les solveurs SAT et MaxSAT, telles que la propagation unitaire, la détection de *failed literals*, la résolution auto-subsomante, ainsi que d'autres méthodes avancées [8]. Bien que ces techniques partagent des similarités conceptuelles avec celles employées en PLNE, leur implémentation diffère. Afin de déterminer si les solveurs MaxSAT peuvent bénéficier des techniques de prétraitement issues de PLNE, il est nécessaire de convertir un problème MaxSAT en un problème PLNE. Un tel modèle pour MaxSAT partiel pondéré est donné par l'équation (1). Soit c une clause dans S (resp. H) de la forme $S_c^- \vee S_c^+$ (resp. $H_c^- \vee H_c^+$), où S_c^- (resp. S_c^+) est une disjonction de littéraux négatifs (resp. positifs) tels que $\neg x$ (resp. x). De nouvelles variables binaires y_x sont introduites pour chaque variable propositionnelle x, et z_c pour chaque clause souple c de la formule.

$$\textbf{Objectif:} \quad \text{Maximiser} \sum_{c \in S} w_c \cdot z_c \tag{1a}$$

Sous contraintes:

$$\sum_{x \in H_c^+} y_x + \sum_{x \in H_c^-} (1 - y_x) \geq 1, \quad \forall c \in H \tag{1b}$$

$$z_c \leq \sum_{x \in S_c^+} y_x + \sum_{x \in S_c^-} (1 - y_x), \quad \forall c \in S \tag{1c}$$

$$z_c \in \{0, 1\}, \quad \forall c \in S \tag{1d}$$

$$y_x \in \{0, 1\}, \quad \forall x \in V \tag{1e}$$

3 Prétraitement PLNE pour MaxSAT

Cette section présente notre méthodologie visant à intégrer les techniques de prétraitement PLNE dans les solveurs MaxSAT, ainsi que les encodages des variables et des contraintes.

3.1 Méthodologie

Pour intégrer les techniques de prétraitement de la PLNE dans un solveur MaxSAT, nous proposons la méthodologie suivante :

1. À partir d'une instance MaxSAT ($originInst$), construire le modèle PLNE ($originModel$) en utilisant les équations (1a)–(1e).

2. Appliquer des techniques de prétraitement à $originModel$ à l'aide d'un solveur PLNE, puis extraire le modèle prétraité ($preModel$).

3. Convertir $preModel$ en une instance MaxSAT simplifiée ($simpInst$). Si la conversion échoue, interrompre le prétraitement et revenir à la résolution de $originInst$.

4. Résoudre $simpInst$ à l'aide d'un solveur MaxSAT afin d'obtenir la solution de l'instance simplifiée ($simpSol$).

5. Transformer $simpSol$ en une solution de l'instance initiale ($orgSol$) et retourner le résultat.

L'aspect clé de notre méthodologie réside dans la conversion de $preModel$ en $simpInst$. Nous commençons par examiner les types de variables et de contraintes présents dans $preModel$, puis on procède à leur encodage au format CNF pondéré. Cet encodage consiste à associer les variables de $preModel$ à celles de $simpInst$, à encoder les contraintes sous forme de clauses dures, et à représenter la fonction objectif sous forme de clauses souples. Les détails de cette procédure sont décrits dans les sous-sections suivantes.

3.2 Encodage des variables

Les variables dans $preModel$ peuvent être affectées, agrégées ou libres. Nous sauvegardons les valeurs des variables affectées afin de faciliter la reconstruction de la solution initiale. Pour les variables non affectées (sans valeur fixée) de $preModel$, nous appliquons l'algorithme 1 afin de déterminer leurs littéraux correspondants dans $simpInst$. Pour simplifier, nous considérons uniquement deux types d'opérations d'agrégation : $x = y$ et $x = \neg y$. Si l'agrégation implique plus de deux variables, nous retournons un échec et interrompons le processus de prétraitement. La structure de données d'entrée, $vpool$, maintient la correspondance entre les variables de $preModel$ et leurs équivalents dans $simpInst$, garantissant ainsi une reconstruction correcte de la solution de l'instance originale.

Example 1. On suppose qu'on a trois variables dans $preModel$ avec les relations d'agrégation suivantes : $(x = \neg y)$ et $(y = z)$. Il suffit alors de créer une nouvelle variable v_1 dans $simpInst$ pour remplacer x, y et z. La correspondance finale est la suivante : $\{x \to \neg v_1,\ y \to v_1,\ z \to v_1\}$, ce qui garantit les équivalences $(x = \neg y)$ et $(y = z)$.

Input : Variable x, table de correspondance $vpool$
Output : Littéral lit ou 'échec'

if *x n'est pas agrégée* **then**
  if *x n'est pas dans $vpool$* **then**
    $vpool[x] \leftarrow$ nouvelle variable dans $simpInst$;
  return $vpool[x]$;
else if *x est agrégée à y* **then**
  $lit \leftarrow GetVarLit(y, vpool)$;
  if $x = \neg y$ **then**
    return $\neg lit$;
  else
    return lit;
else
  return'échec';

Algorithme 1 : $GetVarLit(x, vpool)$

3.3 Encodage des contraintes

Nous utilisons le solveur SCIP pour effectuer le prétraitement de $originModel$, car il s'agit d'un solveur open-source de programmation en nombres entiers mixtes largement utilisé dans les évaluations MaxSAT. SCIP propose une interface permettant d'extraire des informations détaillées à partir de $preModel$. Nous prenons en compte les types de contraintes listés ci-dessous, générés par SCIP dans $preModel$. Ces contraintes — en particulier la contrainte non linéaire *Logical AND* — proviennent principalement de techniques d'extraction de portes logiques [10]. D'autres types de contraintes, telles que les contraintes pseudo-booléennes [4], seront prises en charge dans le cadre de travaux futurs.

- **Contrainte *Logical OR*:** $\sum_{i=1}^{n} x_i \geq 1$, encodée par la clause $(x_1 \vee x_2 \vee ... \vee x_n)$.
- **Contrainte *Logical AND*:** $\prod_{i=1}^{n} x_i = y$, encodée par les clauses $(y \vee \neg x_1 \vee \neg x_2 \vee ... \vee \neg x_n), (\neg y \vee x_i)$ $\forall 1 \leq i \leq n$.
- **Contrainte *Setppc packing*:** $\sum_{i=1}^{n} x_i \leq 1$, encodée par les clauses $(\neg x_i \vee \neg x_j)$ $\forall 1 \leq i < j \leq n$.
- **Contrainte *Setppc partitioning*:** $\sum_{i=1}^{n} x_i = 1$, encodée par les clauses $(x_1 \vee x_2 \vee ... \vee x_n), (\neg x_i \vee \neg x_j)$ $\forall 1 \leq i < j \leq n$.

L'algorithme parcourt l'ensemble des contraintes dans $preModel$. Si une contrainte correspond à un type reconnu, elle est encodée sous forme d'une ou plusieurs clauses dures et ajoutée à $simpInst$. Dans le cas contraire, le prétraitement est interrompu et l'instance MaxSAT originale est transmise au solveur MaxSAT. Concernant la fonction objectif, les variables de décision sont encodées sous forme de clauses souples, en leur attribuant des poids correspondant aux coefficients des variables dans le modèle PLNE. Afin de limiter le temps consommé durant le prétraitement, la taille de l'instance d'entrée est d'abord vérifiée. Si elle dépasse un seuil prédéfini, le prétraitement est ignoré. De

MSE19-24	SCIP	CPLEX
1843	809	850

Table 1: Comparaison du nombre d'instances résolues par les solveurs PLNE.

Solveur	Avec prétraitement	Sans prétraitement
RC2	1220	1238
WMaxCDCL	1426	1436
EvalMaxSAT	1433	1435
UWrMaxSat	1443	1444

Table 2: Comparaison des solveurs MaxSAT, avec et sans prétraitement, en fonction du nombre d'instances résolues.

plus, comme la taille de $simpInst$ n'est pas nécessairement inférieure à celle de l'instance initiale, nous réutilisons l'instance d'origine dans le cas où $simpInst$ s'avère plus volumineuse.

4 Évaluation expérimentale

Nous utilisons des solveurs PLNE et MaxSAT de l'état de l'art pour nos expériences. Plus précisément, les solveurs PLNE utilisés sont SCIP (version 9.1.1) et CPLEX (version 22.1.1.0). Parmi les solveurs MaxSAT, le solveur RC2 a remporté l'évaluation MaxSAT de 2019, tandis que les solveurs WMaxCDCL, UWrMaxSat et EvalMaxSAT sont classés parmi les trois meilleurs dans la catégorie non pondérée de l'édition 2024. Les instances MaxSAT proviennent de la catégorie non pondérée des évaluations MaxSAT de 2019 à 2024 (MSE19-24). Afin de garantir l'équité des tests, les doublons ont été supprimés. Les calculs ont été réalisés sur la plateforme MatriCS[2], équipée d'un processeur AMD EPYC 7502 à 64 cœurs (2,5 GHz) et de 1000 Go de mémoire vive. Chaque instance de test est exécutée sur un cœur, avec 31 Go de RAM et une limite de temps fixée à 3600 secondes.

Nous avons comparé notre méthodologie aux solveurs PLNE ainsi qu'aux solveurs MaxSAT. La Table 1, la Table 2 et la Figure 1 présentent, pour chaque solveur, le nombre d'instances résolues ainsi que les temps de calcul correspondants. D'après les résultats, bien que les solveurs PLNE obtiennent des performances modestes sur l'ensemble du benchmark, leurs techniques de prétraitement permettent de simplifier les instances originales. Cette simplification a permis aux solveurs WMaxCDCL et RC2 de résoudre respectivement 10 et 18 instances supplémentaires. En revanche, les techniques de prétraitement n'améliorent pas de manière significative les performances des solveurs EvalMaxSAT et UWrMaxSat, probablement en raison de l'intégration déjà existante de SCIP dans ces deux solveurs [3].

Nous avons ensuite réalisé des expériences supplémentaires sur le solveur WMaxCDCL. La Table 3 présente le nombre d'instances affectées par le processus de prétraitement. Les

[2] https://www.matrics.u-picardie.fr

Famille	Instances	+Petites	+Grandes	Échouées	Ignorées
MS19only	427	84	148	129	66
MS20only	401	54	90	124	133
MS21only	254	42	63	83	66
MS22only	254	33	110	65	46
MS23only	260	37	73	100	50
MS24only	247	19	62	114	52
Total	1843	269	546	615	413

Table 3: Nombre d'instances plus petites (+Petites), plus grandes (+Grandes), échouées et Ignorées pour chaque famille du benchmark. Les instances ignorés sont celles contenant plus de 200 000 variables ou 1 000 000 de clauses.

Métriques (moy)	+Petites	+Grandes
temps de prétraitement	2.88s	19.71s
Taux de variables affectées	34%	6%
Taux de variables agrégées	36%	23%

Table 4: Temps moyen de prétraitement et taux de variables affectées et agrégées.

résultats démontrent que le prétraitement permet de simplifier avec succès 269 instances. En revanche, 615 instances n'ont pas pu être converties en raison de contraintes non prises en charge ou d'agrégations complexes, et 546 instances ont été prétraitées en devenant plus volumineuses que les originales. Cela montre que les techniques de prétraitement PLNE ne sont pas toujours pertinentes pour les instances MaxSAT.

Nous avons également comparé diverses métriques des instances simplifiées avec succès et des instances plus grandes dans la Table 4. D'après les résultats, nous avons observé un contraste marqué entre les deux groupes. Certaines instances sont plus facilement simplifiées par les techniques de prétraitement, tandis que d'autres ne le sont pas. Le taux de variables affectées est nettement plus élevé pour les instances simplifiées avec succès, ce qui indique que la technique de fixation des variables joue un rôle important dans la simplification des instances.

Pour évaluer les bénéfices que WMaxCDCL tire des instances simplifiées, nous avons augmenté la limite de temps à 10 000 secondes et effectué des tests sur les instances de la famille "extension enforcement", qui modélise un problème dans le domaine de l'argumentation abstraite, inclu-

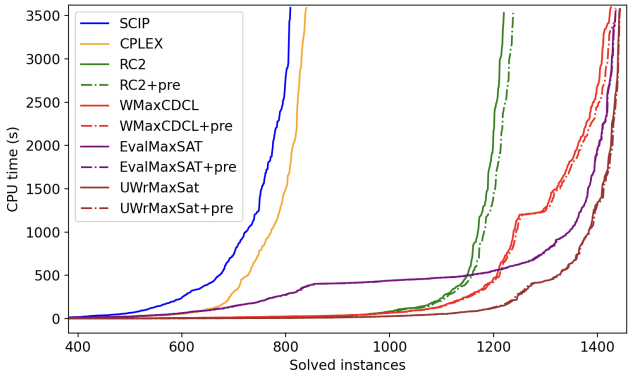

Figure 1: Nombre d'instances résolues vs. temps CPU

	Temps (s)			Taux (%)	
Inst	Origin	Simp	Prétraitement	Vars	Clauses
1	670	54	2.7	30.3	57.1
2	1284	379	2.8	21.3	51.5
3	200	106	3.0	33.6	59.1
4	985	270	2.4	43.5	64.6
5	2814	1549	16.0	42.7	64.6
6	3645	1012	15.1	34.4	59.6
7	>10000	5614	16.9	32.3	58.3
8	>10000	4177	18.3	35.9	60.5
9	4786	1252	16.0	38.5	62.1
10	>10000	3136	14.5	34.2	59.5
11	3170	296	14.5	44.4	65.7
12	589	734	16.8	32.3	58.3
13	5181	804	15.7	37.3	61.4
14	2992	1314	15.7	36.8	61.0
15	5000	1328	15.2	37.3	61.3
16	1774	440	10.8	56.2	72.5
17	1115	193	9.3	56.7	72.8

Table 5: Temps CPU et taux de réduction des variables et des clauses sur les instances de la famille "extension enforcement" (no-strict). "Origin" et "Simp" représentent respectivement le temps CPU pris par WMaxCDCL pour résoudre une instance originale et une instance simplifiée.

ant les variantes $strict$ et $nonstrict$ [14]. Nous avons choisi la variante $nonstrict$ de cette famille, car WMaxCDCL a résolu 5 instances supplémentaires grâce aux techniques de prétraitement proposées en 3600 secondes. Les résultats sont présentés dans la Table 5. Le taux de réduction du nombre de variables est calculé selon la formule $(Origin_{vars} - Simp_{vars})/Origin_{vars} * 100\%$, la même formule étant appliquée au nombre de clauses. D'après les résultats, nous observons que le prétraitement a permis de réduire environ 40% des variables et 60% des clauses, ce qui a permis à WMaxCDCL d'économiser plus de 65% du temps CPU lors de la résolution de ces instances.

5 Conclusion

Dans cet article, nous avons proposé une méthodologie pour intégrer les techniques de prétraitement PLNE dans les solveurs MaxSAT. Les résultats expérimentaux montrent que ces techniques de prétraitement permettent à WMaxCDCL, le vainqueur de l'évaluation MaxSAT 2024 dans la catégorie non pondérée, de résoudre 10 instances supplémentaires dans cette catégorie et de réduire la consommation de temps CPU de plus de 65% pour la résolution des instances de la famille "extension enforcement". Cependant, le prétraitement n'est pas toujours bénéfique pour les solveurs MaxSAT, car il peut parfois rendre les instances simplifiées plus grandes que les originales. De plus, les techniques de prétraitement n'apportent pas d'amélioration substantielle pour les solveurs MaxSAT qui utilisent un solveur PLNE dans un portfolio. Nos travaux futurs viseront à étendre le types de contraintes supportées ainsi que l'agrégation complexe de variables dans $preModel$. Il serait également pertinent de réaliser une analyse approfondie des techniques de prétraitement dans les solveurs PLNE afin d'en améliorer la compatibilité avec les solveurs MaxSAT.

Remerciements

Ce travail est partiellement soutenu 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, et par le projet ANR-24-CE23-6126 (BforSAT). 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.

References

[1] T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, July 2009.

[2] C. Ansótegui and J. Gabàs. Solving (weighted) partial maxsat with ILP. In *Integration of AI and OR Techniques in Constraint Programming*, 2013.

[3] J. Berg, M. Järvisalo, R. Martins, A. Niskanen, and T. Paxian, editors. *MaxSAT Evaluation 2024: Solver and Benchmark Descriptions*. Department of Computer Science Series of Publications B. Department of Computer Science, University of Helsinki, Finland, 2024.

[4] A. Biere, M. Heule, and H. van Maaren. *Handbook of Satisfiability: Second Edition*. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021.

[5] S. Cherif, H. Sattoutah, C. Li, C. Lucet, and L. B. Devendeville. Minimizing working-group conflicts in conference session scheduling through maximum satisfiability (short paper). In P. Shaw, editor, *30th International Conference on Principles and Practice of Constraint Programming, CP 2024, September 2-6, 2024, Girona, Spain*, volume 307 of *LIPIcs*, pages 34:1–34:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[6] E. Demirović and N. Musliu. Maxsat-based large neighborhood search for high school timetabling. *Computers & Operations Research*, 78:172–180, 2017.

[7] C. M. Li and F. Manyà. Chapter 23. MaxSAT, Hard and Soft Constraints. In *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications. IOS Press, Feb. 2021.

[8] C.-M. Li, Z. Xu, J. Coll, F. Manyà, D. Habet, and K. He. Combining Clause Learning and Branch and Bound for MaxSAT. *LIPIcs, Volume 210, CP 2021*, 210:38:1–38:18, 2021.

[9] A. Morgado, C. Dodaro, and J. Marques-Silva. Core-guided maxsat with soft cardinality constraints. In B. O'Sullivan, editor, *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2014.

[10] R. Ostrowski, É. Grégoire, B. Mazure, and L. Sais. Recovering and exploiting structural knowledge from CNF formulas. In P. V. Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, volume 2470 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2002.

[11] M. Piotrow. UWrMaxSat: Efficient Solver for MaxSAT and Pseudo-Boolean Problems. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 132–136, Baltimore, MD, USA, Nov. 2020. IEEE.

[12] S. Safarpour, H. Mangassarian, A. Veneris, M. H. Liffiton, and K. A. Sakallah. Improved design debugging using maximum satisfiability. In *Formal Methods in Computer Aided Design (FMCAD'07)*, pages 13–19, 2007.

[13] M. W. P. Savelsbergh. Preprocessing and Probing Techniques for Mixed Integer Programming Problems. *ORSA Journal on Computing*, 6(4):445–454, Nov. 1994.

[14] J. P. Wallner, A. Niskanen, and M. Järvisalo. Complexity results and algorithms for extension enforcement in abstract argumentation. *J. Artif. Int. Res.*, 60(1):1–40, Sept. 2017.

[15] Z. Zheng, S. Cherif, and R. S. Shibasaki. Optimizing power peaks in simple assembly line balancing through maximum satisfiability. In *36th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2024, Herndon, VA, USA, October 28-30, 2024*, pages 363–370. IEEE, 2024.