Approche générique pour l'acquisition de contraintes qualitatives

Mohamed-Bachir Belaid, Nassim Belmecheri, Arnaud Gotlieb, Nadjib Lazaar, Helge Spieker

Approche générique pour l'acquisition de contraintes qualitatives *

Mohamed-Bachir Belaid,[1] Nassim Belmecheri,[2] Arnaud Gotlieb,[2] Nadjib Lazaar[3] and Helge Spieker[2]

[1]NILU, Norwegian Institute for Air Research, Kjeller, Norway.
[2]Simula Research Laboratory, Oslo, Norvége
[3] LIRMM, Université de Montpellier, CNRS, Montpellier, France

Abstract

Many planning, scheduling or multi-dimensional packing problems involve the design of subtle logical combinations of temporal or spatial constraints. On the one hand, the precise modelling of these constraints, which are formulated in various relation algebras, entails a number of possible logical combinations and requires expertise in constraint-based modelling. On the other hand, active constraint acquisition (CA) has been used successfully to support non-experienced users in learning conjunctive constraint networks through the generation of a sequence of queries. In this paper, we propose GEQCA*, which stands for Generic Qualitative Constraint Acquisition, an active CA method that learns qualitative constraints via the concept of qualitative queries.* GEQCA *combines qualitative queries with time-bounded path consistency (PC) and background knowledge propagation to acquire the qualitative constraints of any scheduling or packing problem. We prove soundness, completeness and termination of* GEQCA *by exploiting the jointly exhaustive and pairwise disjoint property of qualitative calculus and we give an experimental evaluation that shows (i) the efficiency of our approach in learning temporal constraints and, (ii) the use of* GEQCA *on real scheduling instances.*

Résumé

De nombreux problèmes de planification et d'ordonnancement impliquent la création de combinaisons subtiles de contraintes temporelles ou spatiales. La modélisation précise de ces contraintes, qui sont formulées dans diverses algèbres de relations, nécessite une expertise en modélisation basée sur les contraintes et implique un grand nombre de combinaisons logiques possibles. L'acquisition active de contraintes (AC) a été utilisée avec succès pour aider les utilisateurs non expérimentés à apprendre les réseaux de contraintes conjonctives par la génération d'une séquence de requêtes. Dans cet article, nous proposons une méthode d'AC appelée GEQCA *pour Generic Qualitative Constraint Acquisition, qui permet d'apprendre les contraintes qualitatives en utilisant des requêtes qualitatives.* GEQCA *combine les requêtes qualitatives avec la cohérence de chemin limitée dans le temps (PC pour Path Consistency) et la propagation des connaissances de base pour acquérir les contraintes qualitatives. Nous prouvons la correction, la complétude et la terminaison de* GEQCA*. Nous présentons également une évaluation expérimentale qui montre l'efficacité de notre approche dans l'apprentissage des contraintes temporelles ainsi que l'utilisation de* GEQCA *sur des instances réelles d'ordonnancement.*

Introduction

Le raisonnement sur le temps et l'espace est essentiel pour résoudre de nombreux problèmes pratiques, tels que la planification automatisée [4] et l'ordonnancement [2]. Dans ce contexte, le calcul qualitatif fournit un cadre algébrique qui établit des relations entre des paires d'entités à l'aide d'un langage qui est *exhaustif et disjoint par paires*. Des exemples de calculs qualitatifs incluent (sans s'y limiter) l'algèbre des points [6] ou l'algèbre des intervalles d'Allen [1] pour raisonner sur les tâches temporelles, et le calcul des connexions de régions (RCC) [5] pour raisonner sur les relations topologiques entre les régions spatiales. Dans ce contexte, les techniques de satisfaction de contraintes et de programmation par contraintes (CP) sont des cadres pratiques pour modéliser et résoudre des réseaux de contraintes qualitatives. Pour faciliter la modélisation des problèmes de programmation par contraintes, Bessiere et al. ont introduit un cadre permettant d'apprendre les modèles de contraintes par un apprentissage passif à partir d'un ensemble d'exemples d'affectations étiquetées ou par un apprentissage actif avec des requêtes spécifiques permettant de classer les affectations complètes.

Cet article présente le concept de l'acquisition générique de contraintes qualitatives (GEQCA), un nouvel algorithme d'acquisition active de contraintes pour apprendre tout type de contraintes qualitatives entre entités. L'algorithme GEQCA combine des requêtes qualitatives, une cohérence de chemin limitée dans le temps, une heuristique dédiée et une propagation des connaissances de base pour acquérir des contraintes. L'algorithme est conçu pour répondre aux limitations des algorithmes d'acquisition de contraintes existants, telles que l'incapacité de traiter les disjonctions, le contrôle du nombre de requêtes et la connaissance limitée du contexte. L'objectif de GEQCA est de faciliter la modélisation et la résolution de réseaux de contraintes complexes dans des situations pratiques telles que les problèmes d'ordonnancement.

*Cette présentation se base sur des résultats publiés à AAAI 2022 [3].

GEQCA : Acquisition de contraintes via des requêtes qualitatives

GEQCA est un algorithme générique conçu pour apprendre des contraintes qualitatives. Il est basé sur un nouveau concept appelé "requête qualitative", où l'utilisateur doit confirmer si une relation atomique est valide entre une paire de variables d'entité données. GEQCA prend en entrée un vocabulaire de variables d'entités, un langage de relations atomiques, des connaissances de base et un timeout comme paramètre. L'algorithme commence avec un réseau contenant uniquement des contraintes universelles entre entités, puis itère sur les paires d'entités pour réduire l'ensemble des relations possibles à un ensemble localement consistant avec ce que l'utilisateur a en tête. GEQCA utilise une procédure de propagation pour réduire automatiquement les relations incompatibles avec l'état courant de l'apprentissage sans avoir besoin de passer par l'utilisateur.

Experiments

Notre évaluation expérimentale de GEQCA porte sur l'algèbre d'Allen appliquée à des entités temporelles. Le langage Γ utilisé contient les 13 relations atomiques connues de cette algèbre.

Le tableau 1 présente l'effort fourni par l'utilisateur pour résoudre 5 instances de planification en utilisant GEQCA, avec une heuristique de sélection des paires que nous avons introduite dans ce travail. Nous avons utilisé les instances RCPSP3, disponibles publiquement, en considérant la structure du problème incluant la durée des tâches, les exigences en ressources et les capacités des sources, noté K_1. De plus, certaines contraintes peuvent déjà être connues de l'utilisateur, telles que la contrainte globale cumulative et la contrainte de délai. Nous appelons K_2 le background knowledge incluant la contrainte cumulative et la contrainte de délai. Nous avons également utilisé une limite de temps (`cutoff`) d'une heure et noté T_{max} le temps d'attente maximum entre deux requêtes. Chaque instance de planification est caractérisée par le nombre de tâches (par exemple, `sch_30_1` fait référence à l'instance numéro 1 avec 30 `TIs`).

La première observation est que l'utilisation de GEQCA avec des connaissances sur la structure du problème K_1 permet de réduire considérablement l'effort de l'utilisateur (en moyenne une réduction de 38%). La deuxième observation est que l'effort de l'utilisateur est également réduit lorsqu'il utilise des connaissances qui portent sur des contraintes connues telles que la contrainte *cumulative* et les contraintes de délai. Nous observons une réduction de 41% en utilisant K_1 pour la propagation. L'utilisation de $K_1 \wedge K_2$ apporte une amélioration faible mais non significative (en moyenne une réduction de 41% au lieu de 38%). En outre, en termes de temps CPU, le temps d'attente entre deux requêtes peut atteindre le seuil d'une heure sous $\mathcal{K}= K_1 \wedge K_2$. Cela s'explique par la procédure *solve*, qui peut prendre plus d'une heure pour essayer de prouver la cohérence d'une relation avec le réseau appris.

Instance	$\mathcal{K}$					
	$\varnothing$		K_1		$K_1 \wedge K_2$	
	`eF`	T_{max}	`eF`	T_{max}	`eF`	T_{max}
`sch_30_1`	95%	0.79	55%	0.91	53%	1.18
`sch_30_2`	98%	0.62	52%	0.90	48%	393.08
`sch_60_1`	99%	4.80	65%	8.51	62%	13.00
`sch_60_2`	99%	7.72	64%	8.08	61%	12.55
`sch_60_3`	98%	5.94	59%	9.66	57%	3,600

TABLE 1 – Effort de l'utilisateur `eF` avec GEQCA agissant sur les instances RCPSP (avec `cutoff` $= 3,600s$, T_{max} en secondes).

Conclusion

Cet article, publié à AAAI 2022, présente un nouvel algorithme d'apprentissage actif appelé GEQCA pour apprendre des réseaux qualitatifs via des requêtes qualitative. L'algorithme utilise la propriété JEPD du calcul qualitatif et la cohérence du chemin sur les contraintes temporelles pour prouver la convergence et minimiser le nombre de requêtes nécessaires. Les résultats montrent que GEQCA est une approche appropriée et efficace pour des applications pratiques.

Remerciements

Ce travail a reçu un financement du projet T-LARGO et AutoCSP du Conseil norvégien de la recherche, accord de subvention n° 274786 et n° 324674. Ainsi que du programme de recherche et d'innovation Horizon 2020 de l'Union européenne (projet TAILOR).

Références

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11) :832–843, November 1983.

[2] Roman Barták, Miguel Salido, and Francesca Rossi. Constraint satisfaction techniques in planning and scheduling. *Journal of Intelligent Manufacturing*, 21 :5–15, 02 2008.

[3] Mohamed-Bachir Belaid, Nassim Belmecheri, Arnaud Gotlieb, Nadjib Lazaar, and Helge Spieker. Geqca : Generic qualitative constraint acquisition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3690–3697, 2022.

[4] Said Belhadji and Amar Isli. Temporal Constraint Satisfaction Techniques in Job Shop Scheduling Problem Solving. *Constraints*, 3(2) :203–211, June 1998.

[5] David A Randell, Zhan Cui, and Anthony G Cohn. A spatial logic based on regions and connection. *KR*, 92 :165–176, 1992.

[6] Marc B Vilain and Henry A Kautz. Constraint propagation algorithms for temporal reasoning. In *AAAI*, volume 86, pages 377–382, 1986.

Utilisation de SAT pour résoudre le problème SALBP avec minimisation du pic de consommation

Matthieu Py, Arnauld Tuyaba

Université Clermont Auvergne, Mines Saint-Etienne, CNRS, LIMOS, F-63000 Clermont-Ferrand, France

{matthieu.py, arnauld.tuyaba}@uca.fr

Résumé

Dans cet article, on s'intéresse au problème SALB3PM (Simple Assembly Line Balancing Problem with Power Peak Minimization). On propose une résolution de ce problème en utilisant une séquence de problèmes SAT à résoudre, avec appels à un solveur SAT pour résoudre chaque sous-problème. La pertinence de cette approche est vérifiée expérimentalement par comparaison avec les résultats existants sur quelques instances issues de la littérature. Ces travaux sont une extension de [4].

Mots-clés

Ordonnancement, SALB3PM, Modélisation, Programmation par contraintes, SAT, contraintes énergétiques

1 Présentation du problème SALB3PM

Le problème SALB3PM (Simple Assembly Line Balancing Problem with Power Peak Minimization) est un problème d'optimisation qui consiste, étant donné un ensemble de tâches et un ensemble de machines, à construire l'ordonnancement des tâches sur les différentes machines de manière à minimiser le pic d'énergie consommé par l'ordonnancement obtenu. Les données du problème SALB3PM sont les suivantes :

- Un ensemble O de n tâches, où chaque tâche $j \in O$ a une durée de traitement t_j et une consommation énergétique W_j.
- Un ensemble de machines M numérotées de 1 à m.
- Un temps de cycle c : après c unités de temps, les machines sont arrêtées. L'horizon temporel est découpé en périodes et est noté $T = \{0, ..., c-1\}$. On note l'ensemble $T^j = \{0, ..., c-t_j\}$ l'ensemble des périodes où peut commencer la tâche $j \in O$ pour pouvoir terminer avant la fin du temps de cycle.
- Un ensemble de précédences P : si une tâche $i \in O$ précède une tâche $j \in O$, ce qui est noté $i \prec j$, alors :
  - soit la tâche i est affectée à une machine de plus petit numéro que la tâche j,
  - soit les tâches i et j sont affectées à la même machine mais la tâche i est ordonnancée avant la tâche j.

L'objectif du problème SALB3PM est de décider d'affecter chaque tâche sur une machine et d'ordonnancer les tâches affectées sur les mêmes machines (c'est à dire déterminer leur période de lancement), en minimisant le pic d'énergie de la solution obtenue, c'est à dire en minimisant la consommation maximale énergétique des tâches ordonnancées simultanément.

2 Modélisation linéaire existante

Dans l'article [3], le problème SALB3PM est modélisé sous la forme du programme linéaire en nombres entiers présenté dans la figure 1. Les variables de décision sont les suivantes :

- Les variables d'affectation $X_{j,k}$. Étant donnée une tâche $j \in O$ et une machine $k \in M$, la variable $X_{j,k}$ vaut 1 si et seulement si la tâche j est affectée à la machine k.
- Les variables d'ordonnancement $S_{j,t}$. Étant donnée une tâche $j \in O$ et une période $t \in T$, la variable $S_{j,t}$ vaut 1 si et seulement si la tâche j débute à la période t.
- La variable W_{max} est une borne supérieure sur le pic de consommation énergétique.

Les lignes du programme linéaire en nombres entiers modélisent la fonction objectif et les contraintes du problème :

1. On doit minimiser le pic de consommation énergétique.
2. Chaque tâche doit être affectée à exactement une machine.
3. La somme des durées des tâches affectées à chaque machine ne peut pas dépasser le temps de cycle.
4. Si une tâche i précède une tâche j ($i \prec j$), alors la tâche i ne peut pas être affectée à une machine de plus grand numéro que la tâche j.
5. Chaque tâche doit débuter à une et une seule période.
6. Si une tâche i précède une tâche j ($i \prec j$) et que les deux tâches sont affectées à la même machine, alors la tâches j débute forcément après la tâche i.
7. Il est impossible d'avoir deux tâches en cours d'exécution sur la même machine à une période donnée.

$$min \quad W_{max} \tag{1}$$

$$s.t. \quad \sum_{k \in M} X_{j,k} = 1 \qquad \forall j \in O \tag{2}$$

$$\sum_{j \in O} t_j \times X_{j,k} \leq c \qquad \forall k \in M \tag{3}$$

$$X_{j,k} \leq \sum_{h \in M : h \leq k} X_{i,h} \qquad \forall i, j \in O : i \prec j, k \in M \tag{4}$$

$$\sum_{t \in T^j} S_{j,t} = 1 \qquad \forall j \in O \tag{5}$$

$$S_{j,t} \leq \sum_{\tau=0}^{t-t_i} S_{i,\tau} + 2 - X_{i,k} - X_{j,k} \qquad \forall i, j \in O : i \prec j, k \in M, t \in T^j \tag{6}$$

$$X_{i,k} + X_{j,k} + \sum_{\tau=t-t_i+1}^{t} S_{i,\tau} + \sum_{\tau=t-t_j+1}^{t} S_{j,\tau} \leq 3 \qquad \forall i, j \in O, k \in M, t \in T \tag{7}$$

$$\sum_{j \in O} W_j \times (\sum_{\tau=t-t_j+1}^{t} S_{j,\tau}) \leq W_{max} \qquad \forall t \in T \tag{8}$$

$$X_{i,k}, \;\; S_{i,t} \in \{0,1\}, \;\; W_{max} \in \mathbb{Z}^+ \qquad \forall i \in O, k \in M, t \in T$$

FIGURE 1 – Modélisation de SALB3PM sous forme de PLNE [3]

8. Le pic de consommation énergétique est plus grand que la consommation maximale énergétique des tâches ordonnancées sur une même période.

3 Approche SAT

On modélise le problème SALB3PM sous forme d'un problème de satisfaisabilité booléenne (problème SAT). Comme le problème SAT est un problème de décision, et non un problème d'optimisation, on modélise initialement le problème de faisabilité qui consiste à trouver une solution qui respecte toutes les contraintes du problème et qui est présenté dans la figure 2. Les variables de décisions sont les suivantes, où les deux premiers jeux de variables de décision sont les mêmes que pour la modélisation linéaire :

— Les variables d'affectation $X_{j,k}$. Étant donnée une tâche $j \in O$ et une machine $k \in M$, la variable $X_{j,k}$ vaut 1 si et seulement si la tâche j est affectée à la machine k.

— Les variables d'ordonnancement $S_{j,t}$. Étant donnée une tâche $j \in O$ et une période $t \in T$, la variable $S_{j,t}$ vaut 1 si et seulement si la tâche j débute à la période t.

— Les variables d'activité $A_{j,t}$. Étant donnée une tâche $j \in O$ et une période $t \in T$, la variable $A_{j,t}$ vaut 1 si et seulement si la tâche j est active à la période t.

Les premières lignes de la formule propositionnelle modélisent les contraintes du problème :

1. Chaque tâche doit être affectée à au moins une machine.
2. Chaque tâche doit être affectée à au plus une machine.
3. Si une tâche i précède une tâche j ($i \prec j$), alors la tâche i ne peut pas être affectée à une machine de plus grand numéro que la tâche j.
4. Chaque tâche doit débuter à au moins une période.
5. Chaque tâche doit débuter à au plus une période.
6. Une tâche ne peut pas débuter si elle ne peut pas terminer avant la fin du temps de cycle.
7. Il est impossible d'avoir deux tâches en cours d'exécution sur la même machine à une période donnée.
8. Si une tâche débute à une période donnée, alors elle est forcément active ensuite pendant la durée de son exécution.
9. Si une tâche i précède une tâche j ($i \prec j$), alors la tâche j ne peut pas être affectée à la même machine i et commencer après le début de la tâche i.

On peut constater, par exemple en comparant la ligne (2) du programme linéaire et les lignes (2) et (3) de la formule propositionnelle, que le modèle SAT est beaucoup plus gros que le modèle linéaire. Pour cette raison, on effectue un prétraitement pour calculer un ensemble de décisions *évidentes* à cause de la liste des précédences et de la durée des tâches. Ces prétraitements servent à rajouter de nouvelles contraintes permettent de réduire la taille de l'espace de recherche mais elles servent surtout à éliminer des contraintes du modèle initial (si une des contraintes (2) à (9) est en contradiction avec une des contraintes (10) à (12), alors elle n'est pas intégrée au modèle) :

10 Si une chaîne de précédences empêche une tâche j d'être affectée sur une machine k, on interdit cette affectation (on le note $ip(j,k) = 1$)

$$\bigvee_{k \in M} X_{j,k} \qquad \forall j \in O \quad (1)$$

$$\overline{X_{j,k_1}} \vee \overline{X_{j,k_2}} \qquad \forall j \in O, k_1, k_2 \in M : k_1 \neq k_2 \quad (2)$$

$$\overline{X_{j,k}} \vee \overline{X_{i,h}} \qquad \forall i, j \in O : i \prec j, k, h \in M : k < h \quad (3)$$

$$\bigvee_{t \in T^j} S_{j,t} \qquad \forall j \in O \quad (4)$$

$$\overline{S_{j,t_1}} \vee \overline{S_{j,t_2}} \qquad \forall j \in O, t_1, t_2 \in T^j : t_1 \neq t_2 \quad (5)$$

$$\overline{S_{j,t}} \qquad \forall j \in O, t \in T : t \notin T^j \quad (6)$$

$$\overline{X_{i,k}} \vee \overline{X_{j,k}} \vee \overline{A_{j,t}} \vee \overline{A_{j,t}} \qquad \forall i, j \in O : i \neq j, k \in M, t \in T \quad (7)$$

$$\overline{S_{j,t}} \vee A_{j,t+\epsilon} \qquad \forall j \in O, t \in T^j, \epsilon \in [0, t_j - 1] \quad (8)$$

$$\overline{X_{i,k}} \vee \overline{X_{j,k}} \vee \overline{S_{i,t_1}} \vee \overline{S_{j,t_2}} \qquad \forall i, j \in O : i \prec j, k \in M, \; t_1 \in T^i, t_2 \in T^j : t_1 > t_2 \quad (9)$$

$$\overline{Xj,k} \qquad \forall j \in O, k \in M : ip(j,k) \quad (10)$$

$$\overline{Xj,k} \vee \overline{S_{j,t}} \qquad \forall j \in O, k \in M : p(j,k), t \in T^j : ip(j,k,t) \quad (11)$$

$$A_{j,t} \qquad \forall t \in [c - t_i, t_i - 1] \quad (12)$$

$$\bigvee_{j \in C} \overline{A_{j,t}} \qquad \forall C \in \mathbb{C}, t \in T \quad (13)$$

FIGURE 2 – Modélisation des contraintes de SALB3PM sous forme de formule propositionnelle

11 Si une chaîne de précédences empêche une tâche j d'être affectée sur une machine k à une période t, on interdit d'avoir simultanément cette affectation et cet ordonnancement (on le note $ip(j,k,t) = 1$)

12 Si une tâche a une durée qui dépasse la moitié du temps de cycle, elle est forcément active aux dates médianes

Une fois que l'on a écrit le modèle mathématique précédent, sa résolution par un algorithme de résolution du problème SAT permet de trouver un ordonnancement faisable.

Pour ensuite déterminer l'ordonnancement avec un pic d'énergie minimum, on relance, autant de fois que possible, notre algorithme en intégrant au modèle des contraintes supplémentaires. Ces contraintes correspondent à l'interdiction d'exécuter en même temps un ensemble de tâches qui l'étaient dans la dernière solution calculée et dont la somme des énergies est supérieure au pic d'énergie de la meilleure solution rencontrée depuis le début de l'exécution de l'algorithme. Ces contraintes correspondent à la ligne (13) du modèle et l'ensemble $\mathbb{C}$ contient en permanence l'ensemble des ensembles de tâches que l'on ne souhaite plus voir ordonnancées en même temps. Après ajout de ces contraintes, on relance la résolution du problème de faisabilité.

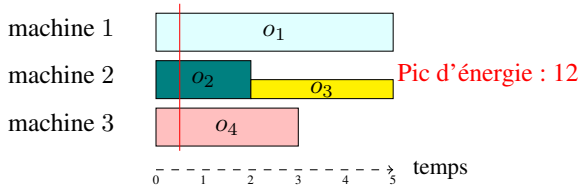
Lorsqu'il n'est plus possible de trouver une solution pour le problème de faisabilité, la meilleure solution rencontrée jusqu'à cet instant est une solution optimale pour le problème SALB3PM.

4 Exemple illustratif

On considère 4 tâches, 3 machines, un temps de cycle $c = 5$ et des précédences lexicographiques $o_1 \prec o_2 \prec o_3 \prec o_4$:

Tâche	o_1	o_2	o_3	o_4
Durée	5	2	3	3
Energie	4	4	2	4

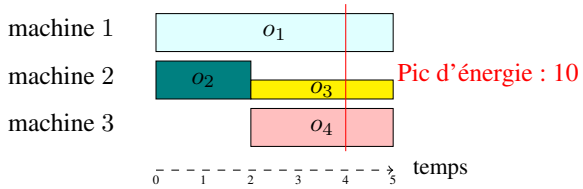
Tout d'abord, on crée le modèle initial et, à la première itération, le solveur SAT trouve la solution suivante :



La solution proposée a un pic d'énergie de 12, causé par l'ordonnancement des tâches o_1, o_2 et o_4. On ajoute donc les contraintes suivantes au modèle, pour empêcher désormais ces trois tâches d'être actives à une même période :

$$\overline{A_{o_1,t}} \vee \overline{A_{o_2,t}} \vee \overline{A_{o_4,t}} \qquad \forall t \in T$$

On relance ensuite le solveur SAT et, à la deuxième itération, le solveur SAT trouve la solution suivante :



Instance	Données			Résultat PLNE			Résultat SAT		
	n	m	c	Pic	Temps	Statut	Pic	Temps	Statut
mertens-1	7	6	6	**104**	**0.01 s**	Optimal	**104**	0.227 s	Optimal
mertens-2	7	2	18	**49**	**0.09 s**	Optimal	**49**	0.238 s	Optimal
bowman-1	8	5	20	**164**	**0.05 s**	Optimal	**164**	0.226 s	Optimal
jaeschke-1	9	8	6	**248**	**0.02 s**	Optimal	**248**	0.21 s	Optimal
jaeschke-2	9	3	18	**78**	**0.16 s**	Optimal	**78**	0.279 s	Optimal
jackson-1	11	8	7	**179**	**0.08 s**	Optimal	**179**	0.254 s	Optimal
jackson-2	11	2	94	**65**	1.05 s	Optimal	**65**	**0.502 s**	Optimal
mansoor-1	11	4	48	**145**	0.69 s	Optimal	**145**	**0.379 s**	Optimal
mansoor-2	11	2	94	**77**	5.03 s	Optimal	**77**	**0.563 s**	Optimal
mitchell-1	21	8	14	**221**	1.44 s	Optimal	**221**	**0.6 s**	Optimal
mitchell-2	21	3	39	**85**	427.94 s	Optimal	**85**	**9.884 s**	Optimal
roszieg-1	25	10	14	**254**	104.36 s	Optimal	**254**	**7.8 s**	Optimal
roszieg-2	25	4	32	**117**	163.59 s	Optimal	**117**	**5.653 s**	Optimal
heskiaoff-1	28	8	138	≤ 290	$\geq$ 3600 s		**251**	**196.693 s**	Optimal
heskiaoff-2	28	3	342		$\geq$ **3600 s**		$\leq$ **107**	$\geq$ **3600 s**	
buxey-1	29	14	25	$\leq$ **292**	$\geq$ **3600 s**		$\leq$ **292**	$\geq$ **3600 s**	
buxey-2	29	7	47	≤ 350	$\geq$ 3600 s		**172**	**137.46 s**	Optimal
sawyer-1	30	14	25	**395**	157 s	Optimal	**395**	**2.044 s**	Optimal
sawyer-2	30	7	47		$\geq$ 3600 s		**214**	**257.916 s**	Optimal
gunther-1	35	14	40	**394**	2297 s	Optimal	**394**	**2.131 s**	Optimal
gunther-2	35	9	54		$\geq$ 3600 s		**295**	**6.081 s**	Optimal

TABLE 1 – Comparaison des approches PLNE et SAT sur quelques instances

La solution proposée a un pic d'énergie de 10, causé par l'ordonnancement des tâches o_1, o_3 et o_4. On ajoute donc les contraintes suivantes au modèle, pour empêcher désormais ces trois tâches d'être actives à une même période :

$$\overline{A_{o_1,t}} \vee \overline{A_{o_3,t}} \vee \overline{A_{o_4,t}} \qquad \forall t \in T$$

On relance ensuite le solveur SAT et, à la troisième itération, le solveur SAT trouve que le problème ne possède pas de solution. La meilleure solution proposée jusqu'à maintenant, qui est ici la solution proposée à la deuxième itération, est donc la solution optimale, avec un pic d'énergie de 10.

5 Expérimentations

Nous avons expérimenté notre algorithme sur plusieurs instances de la littérature, avec pour chaque instance les résultats obtenus en implémentant le modèle existant sur CPLEX [3] et les résultats obtenus avec notre algorithme. Le problème SAT sous-jacent est résolu grâce au solveur Sat4j [1]. On donne, pour chaque instance, le nombre de tâches n, de machines m, le temps de cycle c et, pour chaque méthode, le pic d'énergie de la meilleure solution calculée, la durée d'exécution (au plus 1 heure) et son statut (optimal ou non). On constate dans la table 1 que l'approche proposée permet de trouver la solution optimale sur plus d'instances (19 instances sur 21) qu'avec l'approche ILP connue (15 sur 21).

6 Conclusion

Dans cet article, on a étudié le problème SALB3PM. On a proposé une approche qui fait appel itérativement au problème SAT pour résoudre ce problème. Cette approche a été comparée avec une approche par programmation linéaire en nombres entiers existante, et donne des résultats préliminaires prometteurs. Ces résultats préliminaires doivent être confirmés en offrant le même niveau de finesse de modélisation pour les deux approches et en comparant avec les récentes améliorations du modèle linéaire [2]. Il faudra réfléchir également à comment réduire la taille du modèle SAT initial et comment mieux gérer la fonction objectif. Enfin, il serait intéressant d'essayer d'hybrider ces approches entre elles ou avec d'autres approches heuristiques [5].

Références

[1] Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7 :59–64, 2010.

[2] Paolo Gianessi and Xavier Delorme. A new ILP Model for a Line Balancing Problem with Minimization of Power Peak. In *31st European Conference on Operational Research (EURO-2021)*, 2021.

[3] Paolo Gianessi, Xavier Delorme, and Oussama Masmoudi. Simple Assembly Line Balancing Problem with Power Peak Minimization. In *IFIP International Conference on Advances in Production Management Systems (APMS)*, 2019.

[4] Matthieu Py and Arnauld Tuyaba. Résolution du problème SALB3PM grâce à une approche SAT. In *24ème édition du congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*, 2023.

[5] Arnauld Tuyaba, Laurent Deroussi, Nathalie Grangeon, and Sylvie Norre. Prise en compte de la consommation énergétique dans l'équilibrage de lignes d'assemblage. In *24ème édition du congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*, 2023.