Recherche incomplète aidée par une base de conflits

Trong-Hieu Tran[1], Cédric Pralet[2], Hélène Fargier[1]

[1] IRIT-CNRS, Université de Toulouse, Toulouse, France
{trong-hieu.tran,helene.fargier}@irit.fr
[2] ONERA/DTIS, Université de Toulouse, F-31055 Toulouse, France
cedric.pralet@onera.fr

Mots-clés : *recherche incomplète, explications, compilation de connaissances, OPTW*

1 Motivations

Dans ce travail, nous considérons la résolution de problèmes d'optimisation combinatoire avec des méthodes de recherche incomplète aidées par une base de conflits. Pour fixer les idées, nous prenons l'exemple du problème OPTW (*Orienteering Problem with Time Windows* [3]), dans lequel on considère un ensemble de N clients pouvant être visités pendant certaines fenêtres temporelles et un véhicule nécessitant un certain temps de transition entre deux clients successifs. L'objectif est alors de sélectionner un sous-ensemble de clients et de choisir un ordre de visite pour ces clients de telle sorte que cet ordre soit faisable au sens des fenêtres temporelles autorisées, et de manière à maximiser la somme des récompenses R_i associées aux clients i sélectionnés. Pour les OPTW, une méthode de référence est la recherche locale itérée (ILS) qui consiste à itérer des phases de recherche locale (mouvements de type *insert*, *swap*, *2-opt*, *replace*) et de perturbation de la solution courante [2]. Ces itérations sont généralement effectuées de façon totalement indépendante, c'est-à-dire qu'une itération donnée n'exploite pas du tout des connaissances qu'il aurait été possible d'acquérir lors des itérations précédentes. Partant de ce constat, nous proposons de mémoriser des connaissances acquises au fur et à mesure des itérations, soit pour éviter de réexplorer des configurations non admissibles, soit pour identifier les zones les plus prometteuses dans l'espace de recherche. Par rapport à des mécanismes de recherche tabou, l'enjeu est d'obtenir une base de conflits plus pérenne.

2 Recherche incomplète aidée par les conflits

Pour expérimenter cette approche sur les OPTW, nous nous concentrons sur l'amélioration de la phase d'insertion de la méthode ILS. Plus précisément, nous proposons d'enregistrer des *conflits* représentant des combinaisons de clients ne pouvant pas tous être sélectionnés. Chaque conflit correspond à un sous-ensemble de clients $c = \{i_1, \ldots, i_k\} \subseteq [1..N]$. Des conflits portant sur un nombre restreint de clients peuvent par exemple être identifiés par des méthodes de programmation dynamique qui explorent efficacement tous les ordres de visite possibles.

Partant de là, après chaque phase de perturbation dans ILS, nous utilisons les conflits trouvés lors des itérations précédentes pour éviter de tester certains mouvements d'ajouts ou de remplacement de clients. Nous les utilisons aussi potentiellement pour déterminer un ensemble de clients à $S \subseteq [1..N]$ tel que la somme des récompenses des clients dans S soit maximale et tel qu'il n'existe pas de conflit interdisant la sélection simultanée des clients dans S. En introduisant des variables $x_i \in \{0, 1\}$ représentant la sélection des clients i, cela revient à résoudre le problème de maximisation de la fonction objectif $\sum_{i \in [1..N]} R_i\, x_i$ sous la contrainte $x_{i_1} + \ldots + x_{i_k} \leq k - 1$ pour tout conflit $c = \{i_1, \ldots, i_k\}$ de la base. Une fois l'ensemble de clients S trouvé, on considère d'abord l'insertion de tous les clients dans S (exploitation de la vue globale de la base de conflits), et dans un second temps l'insertion des clients j restants dans $[1..N] \setminus S$, pour autant que la base de conflits n'interdise par leur sélection.

3 Méthodes de gestion de la base de conflits

Dans l'algorithme décrit précédemment, deux questions fondamentales sont : (1) sous quelle forme mémoriser les conflits et (2) comment utiliser la base de conflits pour extraire efficacement une sélection S. Pour répondre à ces questions, trois méthodes ont été explorées.

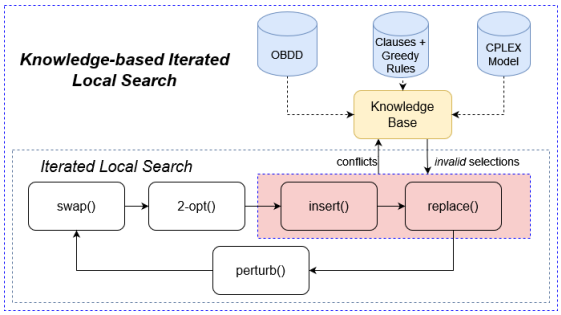

FIG. 1 – Architecture proposée pour aider la recherche locale pour le problème OPTW

La première utilise un diagramme de décision binaire ordonné (OBDD [1]) pour stocker les conflits. Les bonnes propriétés des OBDD permettent d'extraire une sélection optimale de clients S en temps linéaire en la taille de l'OBDD, cependant cette taille peut croître très vite lors des ajouts successifs de conflits.

La deuxième méthode mémorise les conflits sous forme d'une simple liste et extrait une sélection optimale S à chaque étape à l'aide d'un outil de programmation linéaire 0/1. L'inconvénient est que lorsque le nombre de conflits augmente trop, extraire une sélection optimale devient un problème dur.

La troisième méthode mémorise toujours une simple liste de conflits mais utilise une recherche incomplète pour extraire très rapidement une sélection S de bonne qualité. Nous avons testé ici un mécanisme très simple de recherche gloutonne qui tente de sélectionner à chaque étape un client compatible avec les sélections précédentes et dont la récompense associée est maximale.

4 Analyses et travaux futurs

Les tests réalisés sur des instances d'OPTW ont montré que l'utilisation de la base de conflits permettait d'éviter un nombre potentiellement conséquent d'essais d'insertion de clients qui sont incompatibles avec des clients déjà présents dans la solution courante. Pour l'instant, l'utilisation de la base de conflits détériore par contre les performances car elle diminue fortement le nombre d'itérations réalisées dans un temps de calcul donné. Une des pistes de travaux futurs consiste à travailler sur cet aspect pour limiter le temps dédié à l'identification des conflits et à la gestion de la base de conflits. Nous comptons également ajouter une phase de prétraitement pour peupler très rapidement la base de conflits avec des combinaisons de sélections incohérentes. Nous envisageons enfin étendre l'approche à un contexte de décision dans l'incertain, par exemple en cas de modification des récompenses associées aux clients.

Solveur	ILS	ILS-OBDD	ILS-GreedyRules	ILS-CPLEX
Ecart	**2.37**	11.12	10.07	10.24
#itérations par seconde	**42.15**	0.51	10.47	0.19

TAB. 1 – Comparaison entre les différentes approches avec des instances c10x de Solomon (temps limite = 60s). L'écart est la distance moyenne avec les meilleures valeurs connues

Références

[1] R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8) :677–691, 1986.

[2] A. Gunawan, H. C. Lau, P. Vansteenwegen, and K. Lu. Well-tuned algorithms for the team orienteering problem with time windows. *Journal of the Operational Research Society*, 68(8) :861–876, 2017.

[3] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden. The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, 47(1) :53–63, 2013.