Optimisation du placement de trains sur des voies de garage

Léa Blaise[1,2]

[1] DPF Solutions, SNCF Réseau, France
[2] LocalSolver, Paris, France
lblaise@localsolver.com

Mots-clés : *Optimisation, recherche locale, garage de trains*

1 Introduction - le *Track Assignment Problem*

La programmation des opérations en gare ferroviaire (nettoyage, accouplement des rames...) est cruciale à la qualité de service. Ces opérations imposent de réaliser des manœuvres et nécessitent de garer des rames. On souhaite alors placer les rames sur les voies de façon à effectuer le moins de mouvements techniques possible : il s'agit du *Track Assignment Problem*.

Le problème se présente sous la forme suivante : à partir d'un ensemble de trains arrivant et partant d'une gare à des horaires donnés, il s'agit d'affecter un maximum de trains aux voies de la gare (les autres étant affectés à une voie fictive), de sorte qu'ils ne se gênent pas les uns les autres au moment de quitter la gare. On considèrera que toutes les voies sont en impasse : les trains entrent et sortent toujours par le même côté (ces voies sont donc gérées comme des piles). Lorsque les trains sont typés, on peut ajouter un objectif secondaire au problème : regrouper les trains de même type sur les voies, afin de faciliter leur éventuel accouplement.

2 Maximisation du nombre de trains affectés à des voies

La méthode proposée pour la résolution du *Track Assignment Problem* (premier objectif seulement : maximisation du nombre de trains affectés à des voies) est constituée de deux grandes étapes : obtention d'une solution réalisable au moyen d'un algorithme glouton, puis amélioration de cette solution par un algorithme de recherche locale.

L'algorithme glouton consiste, pour chaque train, à choisir une voie capable de l'accueillir. Il se base d'une part sur un algorithme glouton proposé dans [1], résolvant exactement le problème dans le cas où les voies sont de longueur infinie, et d'autre part sur les algorithmes gloutons classiques de résolution du problème *Bin Packing* (*First Fit* et *Best Fit* : [2]).

La solution renvoyée par l'algorithme glouton est ensuite améliorée par un algorithme de recherche locale. Différents types de transformations élémentaires sont envisagés : affectation à une voie d'un train non affecté, échange de trains entre des voies différentes, et suppression de l'affectation d'un train. A chaque itération de l'algorithme, chacune de ces transformations a une certaine probabilité d'être choisie, d'autant plus élevée que la transformation améliore la valeur de la solution.

3 Minimisation de la présence simultanée de trains de types différents sur une même voie

Ayant obtenu une solution satisfaisante du point de vue du nombre de trains non affectés grâce aux deux algorithmes précédents, on cherche désormais à réduire la présence simultanée de trains de types différents sur une même voie, sans dégrader la valeur de l'objectif principal. Pour cela, on définit trois spécifications de cet objectif secondaire :

— minimisation du nombre de voies contenant simultanément des trains de types différents
— minimisation de la durée totale pendant laquelle des trains de types différents sont simultanément présents sur une même voie
— minimisation du nombre de trains placés derrière un train de type différent.

La méthode proposée pour l'optimisation de cet objectif secondaire se base sur un algorithme de recuit simulé. Les transformations élémentaires envisagées ne détériorent pas la valeur de l'objectif principal : elles consistent en des déplacements et échanges de trains quelconques ou possédant des propriétés particulières. A chaque itération de l'algorithme, chacune de ces transformations a une certaine probabilité d'être choisie, d'autant plus élevée que la transformation a de chances d'être améliorante.

4 Résultats

Nous avons évalué les performances de notre méthode sur des instances comprenant entre 15 et 100 trains. Sur les instances correspondant à des cas réels (gare de Metz au mois de mai 2018, jusqu'à 40 trains), nous obtenons en moins d'une seconde une solution dans laquelle chaque train est affecté à une voie, et deux trains de types différents ne sont jamais présents simultanément sur une même voie. Sur les instances générées, de plus grande taille et plus difficiles à résoudre, nous obtenons en moyenne, après trente secondes de recuit simulé, une solution de bien meilleure qualité qu'en dix minutes de calcul par programmation linéaire en nombres entiers.

Références

[1] Gabriele Di Stefano and Magnus Love Koci : A Graph Theoretical Approach to the Shunting Problem. *Electronic Notes in Theoretical Computer Science*, 92 :16–33, 2004.

[2] David Johnson : Fast Algorithms for Bin-Packing. *Journal of Computer and System Sciences*, 8 :272–314, 1974.