Ordonnancement de véhicules: une approche par recherche locale à grand voisinage

Bertrand Estellon, Frédéric Gardi, Karim Nouioua

Ordonnancement de véhicules: une approche par recherche locale à grand voisinage

Bertrand Estellon[1] **Frédéric Gardi**[1,2] **Karim Nouioua**[1]

[1] Laboratoire d'Informatique Fondamentale
Parc Scientifique et Technologique de Luminy, Marseille

[2] Prologia - Groupe Air Liquide
Parc Scientifique et Technologique de Luminy, Marseille

estellon@lif.univ-mrs.fr gardi@prologia.fr nouioua@lif.univ-mrs.fr

Résumé

Le problème $\mathcal{NP}$-difficile de l'ordonnancement de véhicules a fait l'objet d'une attention particulière ces dernières années. Alors qu'une approche directe par programmation linéaire en nombres entiers ou programmation par contraintes s'avère très souvent inefficace, plusieurs méthodes d'optimisation locale ont au contraire démontré leur efficacité. Nous décrivons ici une nouvelle approche par recherche locale à voisinage large basée sur la programmation linéaire en nombres entiers. Cette approche a notamment été éprouvée lors du Challenge ROADEF'2005 sur un problème d'ordonnancement de véhicules proposé par le constructeur RENAULT, plus complexe que le problème habituellement traité dans la littérature. Pour comparaison, nous décrivons aussi une méthode de recherche locale à petit voisinage, en l'occurrence la méthode qui nous a permis de remporter le Challenge ROADEF'2005.

Abstract

The $\mathcal{NP}$-hard problem of car sequencing has received a lot of attention these last years. Whereas a direct approach based on integer linear programming or constraint programming is generally inefficient, several local optimization methods have on contrary demonstrated their efficiency. Here a new approach by large neighborhood local search is described based on integer linear programming. This approach was notably tested at the time of the Challenge ROADEF'2005 on a car sequencing problem proposed by the manufacturer RENAULT, more complex than the problem generally addressed in the literature. For comparison, we also describe a local search method with small neighborhood, in fact the method which enabled us to gain the Challenge ROADEF'2005.

1 Introduction

Les usines modernes d'assemblage de véhicules sont composées de trois grands ateliers : l'atelier tôlerie dans lequel est assemblé le corps du véhicule, l'atelier peinture dans lequel celui-ci est peint, et enfin l'atelier montage dans lequel sont posés les équipements et les options du véhicule. Le *problème de l'ordonnancement de véhicules* consiste à déterminer l'ordre dans lequel un ensemble de véhicules doit parcourir ces trois ateliers, de façon à en faciliter la fabrication. La séquence de véhicules ainsi formée est appelée film.

Chaque atelier est sujet à des contraintes spécifiques, qui tendent à être en conflit les unes avec les autres. Afin de limiter la complexité du problème, les différents travaux parus sur le sujet [4, 5, 14, 12, 6, 8, 7] ne considèrent que les contraintes et objectifs liés à l'atelier montage. Dans le cadre du Challenge ROADEF'2005, un problème plus complexe était proposé par le constructeur RENAULT, intégrant non seulement les besoins de l'atelier montage mais aussi ceux de l'atelier peinture.

Le problème chez RENAULT. Pour l'atelier peinture, l'objectif est de minimiser la consommation de solvant, lequel est utilisé pour purger les pistolets de peinture à chaque changement de couleur dans le film. De façon plus explicite, le but est de minimiser le nombre de purges ou encore le nombre de changements de couleur dans le film, ce qui revient à grouper dans le film les véhicules ayant des couleurs identiques. Cependant, une séquence de véhicules d'une même couleur

ne doit pas dépasser une longueur maximale, notée PAINT-LIMIT (il faut périodiquement changer de couleur pour permettre un meilleur contrôle visuel de la qualité). Cette contrainte est la seule contrainte dure du problème.

Afin de lisser la charge de travail sur les différents postes composant la chaîne de montage, il est nécessaire d'espacer dans le film les véhicules dont la pose des options nécessitent des opérations lourdes. En d'autres termes, le but est de limiter la densité des véhicules qui demandent beaucoup de travail au montage, afin de ne pas surcharger les postes de travail concernés par l'assemblage de ces véhicules.

Ce besoin d'espacer les véhicules est formalisé par une contrainte de ratio pour chaque option. Par exemple, pour une option à laquelle est associée un ratio 3/7, on ne devra pas trouver plus de 3 véhicules concernés par l'option dans toute séquence contiguë du film, appelée fenêtre, comportant 7 véhicules. Comme il n'est pas possible de savoir a priori si toutes les contraintes de ratio pourront être respectées, celles-ci sont définies comme des contraintes molles. Par conséquent, l'objectif sera de minimiser le nombre de violations pour l'ensemble des contraintes de ratio. Pour l'exemple précédent, si 5 véhicules possèdent l'option en question sur une fenêtre de 7, alors 2 violations sont comptées. Afin de distinguer certaines options dites prioritaires, les violations peuvent être pondérées par une constante.

Motivations. Le problème de l'ordonnancement de véhicules est fortement $\mathcal{NP}$-difficile, même lorsque ne sont considérées que les contraintes liées à l'atelier montage [4, 8], et une approche exacte par programmation linéaire en nombres entiers ou programmation par contraintes atteint ses limites lorsque le nombre de véhicules à ordonnancer dépasse quelques dizaines. Ainsi, plusieurs heuristiques ont été proposées dans la littérature pour résoudre le problème, notamment des algorithmes à base de fourmis [14, 6, 7] et des algorithmes de recherche locale [12, 6]. De la même manière, les méthodes qui ont été présentées par les différents participants au challenge ROADEF'2005 [3] sont basées sur des techniques de recherche locale de type recuit simulé ou recherche tabou (voir [2] pour plus de détails sur ces techniques). La principale caractéristique de toutes ces méthodes est la *petite taille du voisinage exploré* à chaque itération de l'algorithme.

Dans cette note, nous présentons une approche du problème par *recherche locale à grand voisinage* [1]. Tout d'abord, nous proposons une formulation originale du problème par un programme linéaire en nombres entiers. Nous montrons ensuite comment utiliser la *programmation linéaire en nombres entiers* pour explorer de très grand voisinage dans une méthode de recherche locale de type k-échange [2, 1]. Pour comparaison, nous décrivons aussi une méthode de recherche locale à petit voisinage, en l'occurrence la méthode qui nous a permis de remporter le Challenge ROADEF'2005. Les deux méthodes ont été implantées sur le principe de la descente [2] (l'objectif n'est jamais détérioré au cours de l'algorithme) ; en effet, aucun optimum local n'a été rencontré en pratique (sur les instances proposées par RENAULT). En conclusion, nous discuterons des résultats obtenus par chacune de ces deux méthodes et de quelques directions futures de recherche.

2 Recherche locale à grand voisinage

Dans cette section, nous décrivons en détail la méthode de recherche locale à grand voisinage que nous avons implantée lors de la phase de qualification du Challenge ROADEF'2005. L'approche que nous proposons peut être qualifiée d'*hybride*, en ce sens qu'elle mêle *recherche locale* et *résolution exacte par programmation linéaire en nombres entiers*. Nous avons récemment appris que ce type d'approche, formalisée par Mautor et Michelon [10], avait déjà été appliquée avec succès à la résolution d'un problème d'ordonnancement de projet [11].

2.1 Programmation linéaire en nombres entiers

Voici une formulation du problème d'ordonnancement de véhicules (chez RENAULT) par un programme linéaire en nombres entiers. Afin de limiter le nombre de variables de celui-ci, les véhicules identiques sont groupés par classe (deux véhicules sont identiques s'ils ont les mêmes options et la même couleur). Le nombre de véhicules à ordonnancer et le nombre de classes sont respectivement notés NPOS et NCL. Le nombre de véhicules de la classe i est noté N_i. Enfin, $\mathsf{P}_i/\mathsf{Q}_i$ dénote le ratio associé à l'option i.

À chaque couple classe i/position j, nous associons une variable binaire $cl_{i,j}$ dont la valeur est égale à 1 si le véhicule à la position j appartient à la classe i et à 0 dans le cas contraire. Les contraintes (1) et (2), décrites ci-dessous, assurent que tous les véhicules d'une même classe sont assignés à une position dans le film et qu'une position ne sera occupée que par un et un seul véhicule d'une même classe :

$$\forall i \in \{1, \ldots, \mathsf{NCL}\}, \qquad \sum_{j=1}^{\mathsf{NPOS}} cl_{i,j} = \mathsf{N}_i \tag{1}$$

$$\forall j \in \{1, \ldots, \mathsf{NPOS}\}, \qquad \sum_{i=1}^{\mathsf{NCL}} cl_{i,j} = 1 \tag{2}$$

Pour chaque couple classe i/option j, la constante $\mathsf{OP}_{i,j}$ est égale à 1 si les véhicules de la classe i possèdent l'option j et à 0 dans le cas contraire. Ensuite, à chaque couple option i/position j est associée une variable binaire $o_{i,j}$ dont la valeur est égale à 1 si le véhicule en position j possède l'option i et à 0 dans le cas contraire. Les contraintes (3) permettent alors d'exprimer les variables $o_{i,j}$ en fonction des variables $cl_{i,j}$ et des constantes $\mathsf{OP}_{i,j}$:

$$\forall i \in \{1,\ldots,\mathsf{NOP}\} \text{ et } \forall j \in \{1,\ldots,\mathsf{NPOS}\},$$
$$o_{i,j} = \sum_{k=1}^{\mathsf{NCL}} (\mathsf{OP}_{k,i} * cl_{k,j}) \quad (3)$$

Enfin, notons $v_{i,j}$ la variable représentant le nombre de violations de l'option i pour la fenêtre qui commence en j. L'objectif étant de minimiser la somme de ces variables, nous posons :

$$\forall i \in \{1,\ldots,\mathsf{NOP}\} \text{ et } \forall j \in \{1,\ldots,\mathsf{NPOS}-\mathsf{Q}_i+1\},$$
$$\sum_{k=j}^{j+\mathsf{Q}_i-1} o_{i,k} - \mathsf{P}_i \leq v_{i,j} \quad (4)$$

De façon similaire, nous allons maintenant définir les variables et les contraintes concernant les couleurs. Pour chaque couple classe i/couleur j, la constante $\mathsf{CO}_{i,j}$ est égale à 1 si les véhicules de la classe i sont de couleur j et à 0 dans le cas contraire. Ensuite, à chaque couple couleur i/position j est associée une variable binaire $c_{i,j}$ dont la valeur est égale à 1 si le véhicule en position j est de couleur i. Voici les contraintes permettant d'exprimer ces variables en fonction des variables $cl_{i,j}$ et des constantes $\mathsf{CO}_{i,j}$:

$$\forall i \in \{1,\ldots,\mathsf{NCOUL}\} \text{ et } \forall j \in \{1,\ldots,\mathsf{NPOS}\} :$$
$$c_{i,j} = \sum_{k=1}^{\mathsf{NCL}} (\mathsf{CO}_{k,i} * cl_{k,j}) \quad (5)$$

À chaque position est associée une variable binaire p_j dont la valeur est égale à 1 lorsque une purge est effectuée entre les positions $j-1$ et j. L'objectif étant de minimiser le nombre de purges, nous posons :

$$\forall i \in \{1,\ldots,\mathsf{NCOUL}\} \text{ et } \forall j \in \{2,\ldots,\mathsf{NPOS}\} :$$
$$c_{i,j} - c_{i,j-1} \leq p_j \text{ et } c_{i,j-1} - c_{i,j} \leq p_j \quad (6)$$

Enfin, les contraintes dures concernant la PAINT-LIMIT s'expriment naturellement à l'aide des variables $c_{i,j}$:

$$\forall i \in \{1,\ldots,\mathsf{NCOUL}\} \text{ et}$$
$$\forall j \in \{1,\ldots,\mathsf{NPOS}-\mathsf{PAINT\text{-}LIMIT}\} :$$
$$\sum_{k=j}^{j+\mathsf{PAINT\text{-}LIMIT}} cl_{i,k} \leq \mathsf{PAINT\text{-}LIMIT} \quad (7)$$

Pour conclure, la fonction objectif du programme linéaire s'écrit comme suit, avec CV_i le coût d'une violation sur l'option i et CP le coût d'une purge :

$$\text{Minimiser} \sum_{i=1}^{\mathsf{NOP}} \sum_{j=1}^{\mathsf{NPOS}-\mathsf{Q}_i+1} \mathsf{CV}_i \times v_{i,j} + \sum_{j=2}^{\mathsf{NPOS}} \mathsf{CP} \times p_j$$

Notons que si les variables $cl_{i,j}$ sont entières, alors toute solution du programme linéaire est entière, et ce, même si les autres variables sont relâchées. De la même façon, si les variables $o_{i,j}$ et $c_{i,j}$ sont entières, alors la solution du programme l'est également. Par conséquent, le type des variables sera défini comme suit lors de la résolution du programme :

si $\mathsf{NCL} \leq \mathsf{NOP} + \mathsf{NCOUL}$ **alors**
$\forall i \in \{1,\ldots,\mathsf{NCL}\}, \forall j \in \{1,\ldots,\mathsf{NPOS}\}, cl_{i,j} \in \{0,1\}$,
et toutes les autres variables sont relâchées ;
sinon
$\forall i \in \{1,\ldots,\mathsf{NOP}\}, \forall j \in \{1,\ldots,\mathsf{NPOS}\}, o_{i,j} \in \{0,1\}$,
$\forall i \in \{1,\ldots,\mathsf{NCOUL}\}, \forall j \in \{1,\ldots,\mathsf{NPOS}\}, c_{i,j} \in \{0,1\}$,
et toutes les autres variables sont relâchées ;
fin si

Contrairement à la formulation plus classique proposée dans [7] (omettant les variables intermédiaires $o_{i,j}$ et $c_{i,j}$), le nombre de variables entières n'est pas ici déterminé par le nombre de classes mais par le nombre d'options et de couleurs. Ceci s'avère très souvent intéressant, tant pour les instances industrielles proposées par RENAULT que pour les instances de la CSPLib [5]. Cela nous a notamment permis de déterminer que l'instance 21-90 de la CSPLib ne pouvait être satisfaite (la seule dont on ne connaissait pas le statut jusqu'à présent).

2.2 Une recherche locale de type k-échange

Le programme linéaire en nombres entiers que nous venons de formuler nous a permis de résoudre des instances comportant jusqu'à une centaine de véhicules. Malheureusement, pour des instances de grande taille (de l'ordre du millier de véhicules), sa résolution exacte n'est plus possible : la relaxation linéaire du programme fournit des solutions de mauvaise qualité (très fractionnaires, de coût égal à zéro) et le *branch & bound* peine à trouver une première solution entière. Dans cette section, nous montrons comment mettre à profit la puissance de la résolution exacte par programmation linéaire en nombres entiers pour explorer de très large voisinage dans une méthode de recherche locale de type k-échange [2].

À chaque itération de l'algorithme, K véhicules du film sont sélectionnés pour être échangés (nous dirons que ces véhicules sont mobiles). Le voisinage à explorer est alors défini comme étant l'ensemble des films pouvant être obtenus par permutation de ces véhicules mobiles. Parmi les K! voisins potentiels, nous allons en

rechercher un de meilleur coût que celui de la solution courante, voire si possible le meilleur. Pour cela, nous résolvons le programme linéaire en nombres entiers (PLNE) défini comme dans la section précédente, où les variables correspondant aux véhicules non mobiles sont fixées. La résolution de ce programme se fait par un algorithme classique de *branch & bound*, dans une limite de temps fixée à T-MAX secondes. Une fois la résolution terminée ou interrompue faute de temps, la solution courante est mise à jour et les valeurs de K et de T-MAX ajustée, en vue d'effectuer une nouvelle itération. Pour résumer, voici les grandes lignes de l'algorithme.

Algorithme Grand Voisinage :
début ;
initialiser K et T-MAX ;
calculer une solution initiale ;
tant que le temps n'est pas écoulé **faire**
choisir K véhicules mobiles ;
formuler le PLNE pour les K véhicules mobiles ;
lancer la résolution du PLNE durant T-MAX secondes ;
mettre à jour la solution courante ;
ajuster K et T-MAX ;
fin faire ;
fin ;

Plusieurs algorithmes gloutons ont été proposés dans la littérature [6] ; ces algorithmes, que nous avons enrichis pour tenir compte des contraintes de l'atelier peinture, permettent d'obtenir rapidement des solutions initiales de bonne qualité (généralement en temps $O(\text{NPOS}^2)$). Ensuite, le caractère effectif de l'algorithme dépend essentiellement de deux facteurs. Tout d'abord, le choix des valeurs de K et de T-MAX est crucial. Si K est trop grand ou T-MAX trop petit, la résolution du PLNE peut échouer (aucune solution entière n'est trouvée dans le temps imparti). L'idée est de débuter l'algorithme avec une petite valeur de K que l'on accroît au fil des itérations, jusqu'à ce qu'une résolution échoue ; la valeur de K est alors stabilisée, de façon à permettre le plus grand choix de véhicules mobiles possibles pour une valeur de T-MAX fixée. Le deuxième facteur porte sur le choix des véhicules mobiles. En effet, nous avons pu constater que plus les véhicules mobiles sont éloignés les uns des autres dans le film, plus la qualité de la relaxation linéaire du programme s'améliore, abaissant ainsi le temps de recherche d'une solution entière optimale par *branch & bound* (des éléments d'analyse polyédrale [13], que nous ne détaillerons pas ici, permettent d'expliquer en partie ce phénomène). Ainsi, les véhicules mobiles sont en grande partie choisis au hazard dans le film, le reste provenant des positions où apparaissent les violations.

Cet algorithme a été implanté en langage C ANSI dans le cadre du Challenge ROADEF'2005, la librairie GLPK [9] étant utilisée pour la résolution du PLNE. Les expérimentations que nous avons menées sur les instances proposées par RENAULT[1] (qui peuvent comporter jusqu'à plus d'un millier de véhicules) nous ont conduits à fixer la valeur de T-MAX autour de dix secondes. La valeur de K, initialement fixée à une cinquantaine de véhicules, est très rapidement augmentée de façon à sélectionner de la moitié jusqu'au deux tiers des véhicules du film, permettant ainsi d'*explorer des voisinages de taille* $O(\text{NPOS}!)$. En moyenne, le temps passée à la résolution du PLNE varie de trois à quatre secondes. Notons qu'afin d'accélérer la convergence de l'algorithme, une solution réalisable de base, construite efficacement à partir de la solution courante, est passée à l'algorithme du simplexe de façon à réduire le temps de calcul d'une solution de la relaxation linéaire du programme.

3 Recherche locale à petit voisinage

Dans cette section, nous décrivons la méthode de recherche locale à petit voisinage qui nous a permis de remporter le Challenge ROADEF'2005. La méthode en question consiste à appliquer à chaque itération une transformation sur le film ne modifiant celui-ci que *très localement*. Si cette transformation est valide (c'est-à-dire respecte la contrainte dure de PAINT-LIMIT) et ne détériore pas le coût de la solution courante, alors celle-ci est effectuée. Cette méthode s'inspire largement des récents travaux de Gottlieb, Puchta et Solnon [12, 6] sur le sujet. Le travail que nous avons réalisé dans le cadre du Challenge ROADEF'2005 a consisté en une adaptation et un enrichissement des techniques proposées par ces trois auteurs afin de traiter au mieux le problème proposé par RENAULT (en particulier les objectifs et contraintes liés à l'atelier peinture).

Algorithme Petit Voisinage :
début ;
calculer une solution initiale (algorithme glouton) ;
tant que le temps n'est pas écoulé **faire**
choisir une transformation à appliquer au film.
choisir les positions auxquelles celle-ci sera appliquée.
si celle-ci est valide et ne détériore pas le coût **alors**
effectuer la transformation ;
fin si ;
fin faire ;
fin ;

Les transformations que nous utilisons sont au nombre de cinq : *échange*, *insertion avant*, *insertion*

[1] Nos tests ont été réalisés sur un PC équipé d'un processeur 1.6 GHz et d'une RAM de 256 Mo.

arrière, *miroir* et *mélange aléatoire* (voir la figure 1). Le voisinage exploré à partir d'une solution courante est ici *extrêmement réduit*. En effet, le voisinage susceptible d'être exploré à l'aide des quatre premières transformations (échange, insertion avant, insertion arrière et miroir) est au plus de taille $O(\mathsf{NPOS}^2)$. De plus, le choix du voisin n'est guidé par aucune règle sophistiquée : le premier voisin qui ne détériore pas la solution courante est sélectionné pour une nouvelle recherche. Accepter les transformations n'améliorant pas le coût de la solution courante est d'ailleurs une des clés de la réussite de cette approche : cela permet de diversifier très largement la recherche et d'éviter ainsi certains optima locaux. L'emploi du mélange aléatoire, qui ne s'applique ici que sur de petites portions du film (quelques dizaines de véhicules tout au plus), est lui aussi crucial puisqu'il semble assurer, du moins en pratique, une convergence de l'algorithme vers un optimum global.

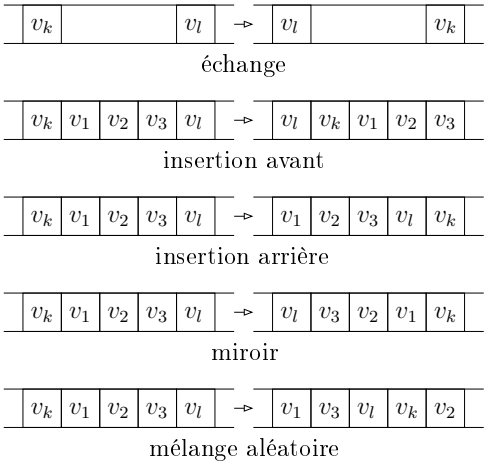

FIG. 1 – Les cinq transformations.

Quant à la fréquence de l'emploi de chacune des cinq transformations, les campagnes d'essais menées sur les jeux de données fournis par RENAULT nous ont permis de tirer les enseignements suivants. De manière générale, le nombre d'échanges doit rester prépondérant (environ 60% du nombre total de transformations appliquées). Viennent ensuite les transformations insertion avant, insertion arrière et miroir (environ 13% chacune du nombre total de transformations) et enfin le mélange aléatoire, coûteux en temps mais nécessaire à la convergence (inférieur à 1% du nombre total de transformations).

Le taux d'échec d'une transformation restant très élevé (seulement 0.3% des transformations sont effectuées en moyenne), l'efficacité de l'algorithme repose sur le très grand nombre d'itérations réalisées. Pour cela, deux critères sont à considérer attentivement : le choix des positions à partir desquelles sont appliquées les transformations et la complexité de la procédure d'évaluation.

Le choix des positions. Voici les différentes stratégies que nous avons utilisées. La première, très simple, consiste à tirer de façon aléatoire les positions v_k et v_l définies sur la figure 1. Une seconde stratégie, moins brutale, consiste à choisir les positions v_k et v_l de façon à contenir des véhicules à l'origine de violations. Toutefois, afin d'accélérer la convergence de l'algorithme, il est nécessaire d'employer des stratégies plus fines, augmentant les chances de réussite d'une transformation. Ces stratégies sont alors fonction de l'objectif et du type de transformation.

Par exemple, lorsque la priorité en terme d'objectif est donnée à la minimisation du nombre de purges, le choix de v_k et v_l correspondra aux début et fin d'une séquence de véhicules d'une même couleur, ceci empêchant l'insertion d'un véhicule d'une couleur différente dans la séquence en question. Pour la transformation du type échange, le choix de positions adjacentes dans le film limitera le risque de détérioration du coût et facilitera l'évaluation. Pour les mêmes raisons, il est utile de choisir deux véhicules partageant certaines caractéristiques au niveau des options ou de la couleur. Pour les transformations du type insertion ou miroir, le choix de v_k et v_l se fera tel que la distance $|l-k|$ soit égale au dénominateur Q_i du ratio d'une des options, ce qui limitera le risque de détérioration du coût.

L'évaluation. L'efficacité de l'évaluation repose sur la remarque suivante. Pour un ratio $\mathsf{P}_i/\mathsf{Q}_i$ donné, pas plus de $2\mathsf{Q}_i$ fenêtres associées à ce ratio ne sont modifiées lors d'une transformation de type échange, insertion ou miroir. En conservant dans une structure de données adéquate le nombre de véhicules concernés par un ratio pour chaque fenêtre du film associée à celui-ci, il est donc possible de calculer en temps $O(\mathsf{Q}_i)$ l'évolution du nombre de violations concernant le ratio en question, suite à l'application d'une de ces transformations.

L'évaluation du nombre de purges, toujours pour une transformation de type échange, insertion ou miroir, peut être quant à elle réalisée en temps $O(1)$. De plus, en stockant pour chaque position j du film les indices de début et de fin de la séquence de véhicules incluant le véhicule en j et de même couleur que ce dernier, il est possible de tester en temps $O(1)$ les dépassement de PAINT-LIMIT.

Enfin, la mise en place d'heuristiques permettant l'arrêt de l'évaluation d'une transformation au plus tôt en cas d'échec permet de réduire encore, et de manière

non négligeable, le temps d'évaluation d'une transformation en moyenne. Ainsi, l'algorithme que nous avons implanté en langage C ANSI dans le cadre du Challenge ROADEF'2005 peut effectuer jusqu'à près de 20 millions d'itérations par minute[2].

4 Résultats expérimentaux

Les tableaux 1 et 2 présentent les résultats que nous avons obtenus sur les instances de la Base A du Challenge ROADEF'2005. Les tableaux 3, 4 et 5 présentent les résultats obtenus respectivement par les algorithmes de Naddef *et al.*, Cordeau *et al.* et Gagné *et al.* [3] sur ces mêmes instances[3]. Dans ces tableaux, EP désigne le nombre de violations des contraintes de ratios prioritaires, ENP le nombre de violations de contraintes de ratios prioritaires et RAF le nombre de purges ; lorsque le nom d'une instance se termine par _EP_RAF_ENP, cela signifie que les objectifs à minimiser sont dans l'ordre et sans compensation possible EP, RAF et enfin ENP. Les tests réalisés par les organisateurs du Challenge ont privilégié la robustesse : le temps d'exécution a été limité à dix minutes sur un PC équipé d'un processeur 1.6 GHz et d'une RAM de 1 Go et les résultats ont été obtenus en effectuant la moyenne après cinq exécutions de l'algorithme.

Instances de la Base A	EP	ENP	RAF
022_3_4_EP_RAF_ENP	0	0	67
022_3_4_RAF_EP_ENP	41	2	11
024_38_3_EP_ENP_RAF	5	157	923
024_38_3_EP_RAF_ENP	10	865	607
024_38_5_EP_ENP_RAF	9	113	951
024_38_5_EP_RAF_ENP	4	477	695
025_38_1_EP_ENP_RAF	0	464	807
025_38_1_EP_RAF_ENP	0	2565	284
039_38_4_EP_RAF_ch1	28	0	373
039_38_4_RAF_EP_ch1	289	0	68
048_39_1_EP_ENP_RAF	5	105	460
048_39_1_EP_RAF_ENP	3	857	287
064_38_2_EP_RAF_ENP_ch1	0	774	180
064_38_2_EP_RAF_ENP_ch2	0	59	44
064_38_2_RAF_EP_ENP_ch1	441	790	63
064_38_2_RAF_EP_ENP_ch2	367	84	27

TAB. 1 – Résultats de l'algorithme Grand Voisinage.

Les algorithmes Grand Voisinage et Petit Voisinage ont tout deux permis d'améliorer très largement les résultats qu'avait jusqu'à présent obtenus RENAULT

[2]Nos tests ont été réalisés sur un PC équipé d'un processeur 1.6 GHz et d'une RAM de 256 Mo.

[3]Les résultats de l'ensemble des participants du Challenge ROADEF'2005 ainsi que les autres instances peuvent être retrouvés à l'adresse http://www.prism.uvsq.fr/~vdc/ROADEF/.

Instances de la Base A	EP	ENP	RAF
022_3_4_EP_RAF_ENP	0	1	31
022_3_4_RAF_EP_ENP	39	1	11
024_38_3_EP_ENP_RAF	4	0	306
024_38_3_EP_RAF_ENP	4	83	249
024_38_5_EP_ENP_RAF	4	34	309
024_38_5_EP_RAF_ENP	4	79	280
025_38_1_EP_ENP_RAF	0	99	720
025_38_1_EP_RAF_ENP	0	2452	229
039_38_4_EP_RAF_ch1	13	0	131
039_38_4_RAF_EP_ch1	161	0	68
048_39_1_EP_ENP_RAF	0	61	290
048_39_1_EP_RAF_ENP	0	615	175
064_38_2_EP_RAF_ENP_ch1	0	759	112
064_38_2_EP_RAF_ENP_ch2	0	51	34
064_38_2_RAF_EP_ENP_ch1	423	782	63
064_38_2_RAF_EP_ENP_ch2	367	52	27

TAB. 2 – Résultats de l'algorithme Petit Voisinage.

Instances de la Base A	EP	ENP	RAF
022_3_4_EP_RAF_ENP	0	1	31
022_3_4_RAF_EP_ENP	39	1	11
024_38_3_EP_ENP_RAF	4	0	302
024_38_3_EP_RAF_ENP	4	83	250
024_38_5_EP_ENP_RAF	4	34	312
024_38_5_EP_RAF_ENP	4	80	282
025_38_1_EP_ENP_RAF	0	117	702
025_38_1_EP_RAF_ENP	0	2274	229
039_38_4_EP_RAF_ch1	13	0	132
039_38_4_RAF_EP_ch1	155	0	68
048_39_1_EP_ENP_RAF	0	61	290
048_39_1_EP_RAF_ENP	0	612	174
064_38_2_EP_RAF_ENP_ch1	0	759	115
064_38_2_EP_RAF_ENP_ch2	0	52	34
064_38_2_RAF_EP_ENP_ch1	423	783	63
064_38_2_RAF_EP_ENP_ch2	367	53	27

TAB. 3 – Résultats de l'approche de Naddef *et al.* [3] par recuit simulé.

à l'aide d'un algorithme de recherche locale de type recuit simulé. Toutefois, les résultats de l'algorithme Grand Voisinage sont nettement inférieurs à ceux de l'algorithme Petit Voisinage, ainsi qu'à ceux des autres finalistes du Challenge. Il est à noter que même si ces derniers diffèrent les uns des autres par le paradigme de recherche locale employé, les algorithmes de Naddef *et al.*, Cordeau *et al.* et Gagné *et al.* [3] (comme ceux des autres finalistes), se basent tous plus ou moins sur les transformations proposées par Gottlieb, Puchta et Solnon [12, 6] et exposées dans la section précédente.

5 Conclusions et perspectives

La méthode de recherche locale à grand voisinage reste à affiner. Tout d'abord, l'écriture d'un algorithme

Instances de la Base A	EP	ENP	RAF
022_3_4_EP_RAF_ENP	0	1	31
022_3_4_RAF_EP_ENP	39	2	11
024_38_3_EP_ENP_RAF	5	35	482
024_38_3_EP_RAF_ENP	4	266	390
024_38_5_EP_ENP_RAF	4	83	422
024_38_5_EP_RAF_ENP	4	274	355
025_38_1_EP_ENP_RAF	0	231	773
025_38_1_EP_RAF_ENP	0	3443	237
039_38_4_EP_RAF_ch1	15	0	179
039_38_4_RAF_EP_ch1	226	0	68
048_39_1_EP_ENP_RAF	0	117	385
048_39_1_EP_RAF_ENP	0	986	214
064_38_2_EP_RAF_ENP_ch1	0	766	114
064_38_2_EP_RAF_ENP_ch2	0	51	34
064_38_2_RAF_EP_ENP_ch1	423	794	63
064_38_2_RAF_EP_ENP_ch2	367	52	27

TAB. 4 – Résultats de l'approche de Cordeau *et al.* [3] par recherche tabou.

Instances de la Base A	EP	ENP	RAF
022_3_4_EP_RAF_ENP	0	1	34
022_3_4_RAF_EP_ENP	39	5	11
024_38_3_EP_ENP_RAF	4	52	737
024_38_3_EP_RAF_ENP	4	508	323
024_38_5_EP_ENP_RAF	4	73	752
024_38_5_EP_RAF_ENP	4	151	392
025_38_1_EP_ENP_RAF	0	288	799
025_38_1_EP_RAF_ENP	0	3245	234
039_38_4_EP_RAF_ch1	15	0	196
039_38_4_RAF_EP_ch1	200	0	68
048_39_1_EP_ENP_RAF	0	81	406
048_39_1_EP_RAF_ENP	0	904	191
064_38_2_EP_RAF_ENP_ch1	0	789	117
064_38_2_EP_RAF_ENP_ch2	0	62	34
064_38_2_RAF_EP_ENP_ch1	423	816	63
064_38_2_RAF_EP_ENP_ch2	367	69	27

TAB. 5 – Résultats de l'approche de Gagné *et al.* [3] par colonie de fourmis.

du simplexe et d'un *branch & bound* dédié au problème (ou plus simplement l'utilisation d'une librairie d'optimisation commerciale comme CPLEX de ILOG ou XPRESS-MP de Dash Optimization) permettrait sans doute d'abaisser considérablement le temps de la résolution du PLNE et ainsi d'accélérer significativement la convergence de l'algorithme. Ensuite, l'intégration d'autres schémas de résolution exacte comme la programmation par contraintes est à étudier.

Malgré cela, nous pensons que la méthode de recherche locale à petit voisinage, correctement implantée, reste à privilégier. Celle-ci permet non seulement d'obtenir très rapidement d'excellents résultats, mais elle est aussi très robuste, ce qui est plus surprenant. Une des clés du succès de cette approche est le fait qu'aucun optimum local n'est jamais rencontré en pratique. Cette propriété est-elle liée à la structure du problème, à la structure des instances, ou bien "le hazard fait-il bien les choses" ? Il serait intéressant d'étudier ce phénomène plus en détail.

Références

[1] R.K. AHUJA, Ö. ERGUN, J.B. ORLIN et A.P. PUNNEN (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics* 123, p. 75–102.

[2] E. AARTS et J.K. LENSTRA (1997). *Local Search in Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester, Angleterre.

[3] CHALLENGE ROADEF'2005. In *Actes du 6ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*, p. 21–42. Ecole Polytechnique de l'Université de Tours, Tours, France.

[4] I.P. GENT (1998). Two results on car sequencing problem. APES Research Report 02-1998. Department of Computer Science, University of Strathclyde, Glasgow, Royaume-Uni.

[5] I.P. GENT et T. WALSH (1999). CSPLib : a benchmark library for constraints. APES Research Report 09-1999. Department of Computer Science, University of Strathclyde, Glasgow, Royaume-Uni.

[6] J. GOTTLIEB, M. PUCHTA et C. SOLNON (2003). A study of greedy, local search and ant colony optimization approaches for car sequencing problems. In *Applications of Evolutionary Computing* (sous la direction de G.R. Raidl et al.), *LNCS* 2611, p. 246–257. Springer-Verlag, Heidelberg, Allemagne.

[7] M. GRAVEL, C. GAGNÉ et W.L. PRICE (2005). Review and comparison of three methods for the solution of the car sequencing problem. (à paraître dans *Journal of the Operational Research Society*)

[8] T. KIS (2004). On the complexity of the car sequencing problem. *Operations Research Letters* 32, p. 331–335.

[9] A. MAKHORIN (2004). Librairie GLPK (GNU Linear Programming Kit, version 4.4). `http://www.gnu.org/software/glpk/`

[10] T. MAUTOR et P. MICHELON (1997). MIMAUSA : a new hybrid method combining exact solution and local search. In *Proceedings of*

the 2nd Metaheuristics International Conference. Sophia-Antipolis, France.

[11] M. Palpant, C. Artigues et P. Michelon (2004). LSSPER : solving the ressource-constrained project scheduling problem with large neighborhood search. *Annals of Operations Research* 31(1), p. 237–257.

[12] M. Puchta et J. Gottlieb (2002). Solving car sequencing problems by local optimization. In *Applications of Evolutionary Computing* (sous la direction de S. Cagnoni et al.), *LNCS* 2279, p. 132–142. Springer-Verlag, Heidelberg, Allemagne.

[13] A. Schrijver (2003). *Combinatorial Optimization : Polyhedra and Efficiency*. Algorithms and Combinatorics 24, Vol. A, Springer-Verlag, Berlin, Allemagne.

[14] C. Solnon (2000). Solving car sequencing problems with artificial ants. In *Proceedings of the 14th European Conference on Artificial Intelligence* (sous la direction de H. Werner), p. 118–122. IOS Press, Berlin, Allemagne.