Une variante de recherche adaptative dans les grands voisinages pour le problème de *job shop*

Rodolphe Piteira[1], Julien Bernard[1], Hervé Manier[2], Marie-Ange Manier[2]

[1] FEMTO-ST, CNRS, UFC, 25000 Besançon, France
piteira@free.fr,julien.bernard@femto-st.fr
[2] FEMTO-ST, CNRS, UTBM, 90010 Belfort, France
{herve.manier, marie-ange.manier}@utbm.fr

Mots-clés : *job shop, recherche adaptative, ALNS*

1 Introduction

La méthode *Adaptive Large Neighborhood Search (ALNS)* est une métaheuristique utilisée pour explorer de grands voisinages à l'aide d'opérateurs de destruction et de reconstruction de solutions [2]. Parmi les problèmes qui ont de grands voisinages figure le problème de *job shop*. Ce problème se caractérise par n *jobs* constitués de n_i opérations ($1 \leq i \leq n$) à ordonnancer sur m machines. La j^{e} opération du job i, notée O_{ij}, doit s'effectuer sur la machine M_{ij} donnée avec un temps d'exécution p_{ij} donné. On note $N = \sum n_i$ le nombre total d'opérations. On cherche à minimiser le temps de complétion maximum $C_{\max}$.

Nous proposons une variation de la recherche adaptative de [2] pour le problème de *job shop* que nous analysons à travers des expériences sur des instances classiques de la littérature.

2 Recherche adaptative pour le problème de job shop

Notre objectif est de tester la pertinence des associations entre représentations des solutions (codages) et voisinages pour la conception de méta-heuristiques plus efficaces. Ainsi, plutôt que de considérer classiquement des opérateurs de destruction et de reconstruction, notre algorithme adaptatif utilise des couples codages-voisinages.

Notre algorithme part d'une solution courante. Ensuite, il effectue les actions suivantes pendant un certain nombre d'itérations : (1) un codage-voisinage ℓ parmi les q couples est choisi aléatoirement proportionnellement à sa pondération α_ℓ, tel que $\sum \alpha_\ell = 1$. À l'initialisation, $\alpha_\ell = \frac{1}{q}$; (2) le meilleur voisin est choisi parmi un certain nombre (noté `essais`) de solutions du voisinage de la solution courante ; (3) les pondérations sont ajustées : si le voisin est meilleur, alors α_ℓ est augmenté de e, sinon il ne change pas de valeur sauf si on considère une pénalisation (`pénal` $= 1$) auquel cas on diminue α_ℓ de e ; puis toutes les pondérations sont normalisées pour que leur somme soit toujours égale à 1 ; (4) si le voisin est meilleur que la solution courante, alors il devient la solution courante.

3 Expérience et résultats

Pour nos expériences, nous considérons deux codages : le codage par *job*, noté `job`, (un vecteur de taille N qui contient n_i fois le job i) et le codage par machine, noté `mch`, (un vecteur de m listes ordonnées d'opérations affectées à chacune des machines) ; ainsi que trois opérateurs de voisinage classiques : l'échange d'adjacent (`adj`), l'échange (`swp`), l'insertion (`ins`). L'opérateur de voisinage s'applique directement sur le vecteur dans le cas du codage par *job* et sur une des m listes choisie aléatoirement dans le cas du codage par machine.

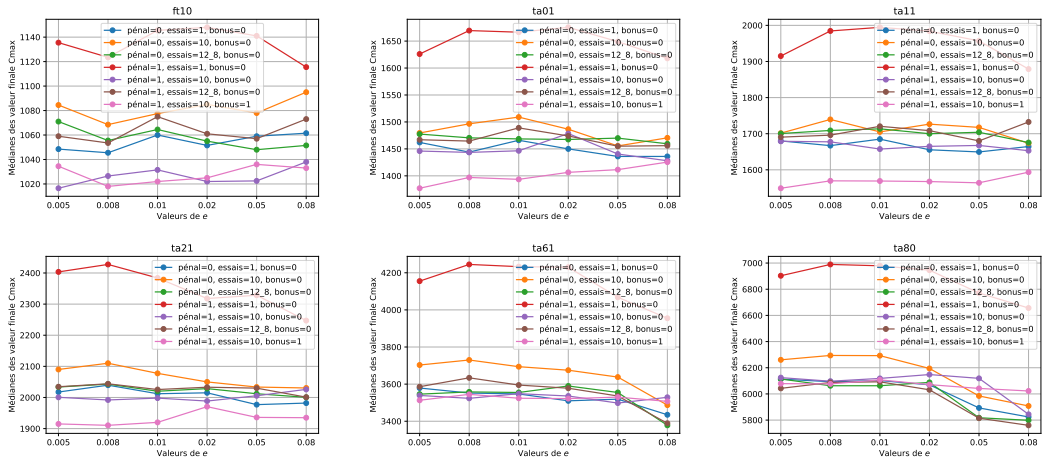

FIG. 1 – $C_{\max}$ médian en fonction de e, pour chacune des 6 instances étudiées

Les valeurs de e testées sont $0.005, 0.008, 0.01, 0.02, 0.05, 0.08$. Trois configurations de `essais` ont été testées : 1, 10 et une configuration spéciale avec 12 pour le codage par *job* et 8 pour le codage par machine pour compenser les temps de calcul des voisins. Enfin, un cas spécial a été expérimenté où le calcul de e prend en compte l'amélioration du $C_{\max}$ en comparant aux cinq dernières améliorations (`bonus` $= 1$). Enfin, nous avons choisi 6 instances de la littérature [1, 3] (n et m entre parenthèses) : `ft10` (10, 10), `ta01` (15, 15), `ta11` (20, 15), `ta21` (20, 20), `ta61` (50, 20), `ta80` (100, 20). Pour chaque lot de paramètres, entre 10 et 30 expériences sont lancées, le nombre d'itérations est fixé à $20 \times n \times m$.

La figure 1 montre les $C_{\max}$ médians pour chacune des instances en fonction de e. La pénalisation avec un seul essai (en rouge) est la pire des configurations. La configuration `essais` à 12/8 se comporte de manière similaire avec ou sans pénalisation (marron et vert) et est performante sur les grandes instances. La pénalisation est utile quand `essais` vaut 10 (orange vs violet). L'ajout du bonus (rose) est utile pour les petites instances. Quant à l'influence de e, on observe que des petites valeurs de e sont préférables pour les petites instances tandis que des grandes valeurs de e sont préférables pour les grandes instances.

Dans la configuration avec bonus, pénalisation et 10 essais (rose), c'est généralement `job-adj` qui devient prépondérant pour les petites instances, et `mch-adj` pour les grandes instances. Dans la configuration sans pénalisation et 12/8 essais, aucun codage-voisinage ne sort du lot pour les petites instances, tandis que `job-ins` est prépondérant pour les grandes instances.

4 Conclusion

La variante de recherche adaptative que nous proposons pour le problème de *job shop* fonctionne de manière satisfaisante et offre des premiers résultats intéressants sur des instances classiques, y compris par rapport à l'ALNS de [2].

Références

[1] Henry Fisher. Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, pages 225–251, 1963.

[2] David Pisinger and Stefan Ropke. Large neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, pages 399–419. Springer US, Boston, MA, 2010.

[3] Eric Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2) :278–285, 1993.