Représentation binaire pour problèmes symboliques

Christelle Bloch[1], Pascal Chatonnay[1]

[1] FEMTO-ST (UMR CNRS 6174), Université de Franche-Comté
1 Cours Leprince-Ringuet, BP 21126 F-25 201 Montbéliard Cedex, France
{christelle.bloch, pascal.chatonnay}@univ-fcomte.fr

Mots-clés *: Métaheuristiques, méthodes génétiques, représentations, VRP*

1 Introduction

Les problèmes de tournées de véhicules (VRP) figurent parmi les problèmes d'optimisation combinatoire les plus étudiés. Les auteurs utilisent généralement une représentation symbolique des solutions, sous la forme d'une ou plusieurs séquences de lieux à visiter (tournées). Ce type de représentation est également très utilisé dans de nombreux problèmes d'ordonnancement d'atelier, en particulier ceux intégrant des contraintes de transport (tels que le HSP, rencontré dans les lignes de traitement de surface). Cette représentation est construite de manière relativement directe à partir du problème sous étude, mais peut présenter toutefois des inconvénients. Ainsi, dès 2004, Prins démontrait tout l'intérêt d'utiliser des chromosomes sous forme de permutations, mais sans délimiteurs, couplés à une procédure de découpage en tournées (nommée « *split* ») [3]. Généralement, ce type de représentation est associé à des croisements de type « *order crossover* » (tels que OX), pour ne pas générer de doublons des symboles au moment du croisement. Dans la même lignée que les travaux de Prins sur ce sujet, nous proposons le passage à une représentation indirecte, dans un premier temps, puis, à terme, à une représentation binaire. Le but est d'estimer l'impact de ces modélisations sur les performances d'un algorithme génétique, comme nous l'avions fait, en 2010 pour comparer les performances de divers opérateurs [2].

2 Changer de représentation pour changer de résolution

La contribution proposée est décrite à travers un exemple présenté dans le tableau 1, et basé sur l'alphabet constitué de n=6 symboles : A,B,C,D,E et F. Elle ré-utilise dans un premier temps un concept d'alphabet, déjà décrit par Prins, pour établir une représentation indirecte (3ème colonne du tableau 1). Cette représentation reste symbolique, mais ne produit pas de répétition de symbole par croisement simple en k-points [1], contrairement au codage de permutation classique (2ème colonne du tableau 1). En effet, avec ce dernier, un croisement dont le point de coupure est situé en milieu de chromosome provoque l'apparition de doublons du symbole C (resp. E), apparaissant en gras dans Enfant1 (resp. Enfant2). C'est pour pallier ce type de fonctionnement que des croisements qui vérifient la présence d'un symbole avant de l'insérer dans une solution fille (tels que OX ou LOX) sont généralement utilisés. Dans le codage indirect, c'est la position de chaque symbole dans l'alphabet de référence (privé des symboles déjà utilisés) qui est représentée, plutôt que le symbole lui-même. Ainsi, A est en position 1 dans A,B,C,D,E,F ; C est en position 2 dans B,C,D,E,F ; et ainsi de suite… notons que le dernier chiffre est toujours égal à 1 car à la dernière étape il ne reste toujours qu'un seul symbole possible. Ce codage garantit ainsi par construction l'absence de répétition d'un même symbole dans les enfants générés par croisement 1- point.

Solutions	Codage permutation	Codage indirect	Codage binaire
Parent1	ACD|EFB	122|221	000 001 01|01 1
Parent2	AED|FCB	143|321	000 011 10|10 1
Enfant1	ACD|F**C**B	122|321=ACD|FEB	000 001 01|01 1= ACD|FEB
Enfant2	AED|**E**FB	143|221=AED|CFB	000 011 10|01 1= AED|CFB

TAB. 1 – Exemple de passage du codage de permutation direct à un codage indirect binaire

Enfin la 4ème colonne du tableau présente un codage indirect binaire équivalent. Dans celui-ci, le codage du dernier symbole (toujours égal à 1) disparaît. Les symboles restants correspondent aux groupes de bits (séparés ici artificiellement par des espaces pour faciliter la compréhension, mais codés en une seule chaine de bits pour la résolution). Ainsi, le premier symbole du codage indirect doit être codé sur trois bits car il peut prendre au maximum *n-1*= 5 valeurs (1, 2, 3, 4, ou 5). Mais toutes les valeurs générées à partir de ces trois bits ne correspondent pas à des valeurs acceptables. Dans cet exemple, et par convention, les 5 valeurs possibles sont en fait codées entre 0 et 4. Ceci implique de neutraliser les valeurs « inutilisables ». Différentes solutions sont envisageables et doivent être comparées (utilisation d'un masque, d'un opérateur modulo, …). Malgré cette légère difficulté, ce codage présente néanmoins plusieurs avantages :

- Comme le codage indirect, il permet de garantir l'absence de répétition des symboles, tout en conservant des mécanismes très simples pour les opérateurs (croisements k-points) ;
- Comme dans le codage sans séparateur de Prins, l'absence de séparateurs entre les bits permet un parcours de l'espace de recherche différent et qui limite la génération de solutions infaisables en reportant le découpage en tournées sur des procédures telles que *split*.
- Sa forme binaire permet de bénéficier de tous les avantages de ce type de codage, notamment mis en lumière par Goldberg [1] et par Whitley [4].

3 Conclusion et perspectives

Cette présentation propose une comparaison de performances des différents codages décrits, basée sur des benchmarks de la littérature. Il s'agit en particulier d'évaluer le rapport du coût de codage/décodage en binaire par rapport aux gains, et d'estimer l'intérêt de ces modélisations en fonction des tailles d'alphabets considérées. Pour prendre en compte les différentes contraintes des problèmes, le codage, quel qu'il soit, sera couplé à une procédure de type « *split* ». L'une des perspectives envisagée est de comparer aussi cette approche d'algorithmique génétique classique à une approche par programmation génétique, peut-être plus proche des problèmes de tournées (dans la mesure où la résolution de ceux-ci consiste à ordonner des actions/instructions).

Références

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Professional, 1989.

[2] M. Haj Rachid, W. Ramdane-Cherif, P. Chatonnay, and C. Bloch. A study of performance on crossover and mutation operators for vehicle routing problem. *ILS'10, 3rd int. Conf. on Information Systems, Logistics and Supply Chain*, Casablanca, Morocco, April 2010.

[3] C. Prins, A simple and effective evolutionary algorithm for the VRP, *Comp. Oper. Res.* 31(12): p. 1985-2002, 2004.

[4] D. Whitley.