Optimisation multiobjective pour le calibrage d'un simulateur à événements discrets : modélisation du trafic routier

Valentin Vendi, Aléxis Pestelle, Sara Tari, Eric Ramat

Laboratoire LISIC, Université du Littoral Côte d'Opale, France
{valentin.vendi, alexis.pestelle, sara.tari, eric.ramat}@univ-littoral.fr

Mots-clés : *optimisation de simulateurs, optimisation multiobjective, mobilité*

1 Introduction

Simuler des systèmes complexes permet de répondre à de nombreux enjeux, dont certains en lien avec la mobilité tel que l'évaluation de scénarios et de nouveaux aménagements. Cependant, la calibration des simulateurs est souvent l'une des étapes coûteuses pour obtenir un modèle réaliste. Nos travaux s'inscrivent dans le cadre du développement d'un simulateur à événements discrets pour le trafic routier en collaboration avec la société Neovya, experte en analyse et simulation du trafic. L'intérêt de développer notre simulateur à événements discrets est de pouvoir répondre aux attentes de couplage multi-échelles et de performances, qui ne sont pas totalement couvertes par le simulateur mésoscopique actuel écrit en Python. Dans cet article, nous nous focalisons uniquement sur la calibration automatique du simulateur qui peut être assimilée à un problème d'optimisation. Le simulateur devant être régulièrement, selon des scénarios d'aménagement, calibré manuellement, automatiser cette tâche permettrait d'effectuer cette opération en quelques heures au lieu des deux semaines nécessaires actuellement tout en laissant un degré de liberté dans le choix de la solution finale. Nos travaux nous ont mené à coupler notre simulateur avec des méthodes d'optimisation multi-objectifs pour répondre à ce besoin.

2 Automatisation de la calibration du modèle

2.1 Formalisation du problème d'optimisation

À l'issue d'une simulation, le simulateur retourne deux jeux de données X et Y avec :
— $x_t^c \in X$: le nombre de véhicules passant sur une boucle de comptage c sur une période de temps t
— $y_t^p \in Y$: le temps de parcours moyen sur un parcours p sur une période de temps t

Le réseau utilisé pour les simulations est un cas d'étude fourni par Neovya, pour lequel un calibrage manuel des paramètres du modèle a été effectué. Nous utilisons la version non calibrée avec des paramètres par défaut. Les tronçons sont divisés en deux catégories : les voies d'autoroute et les voies d'insertion/de sortie, qui possèdent chacunes 3 paramètres que nous cherchons à optimiser. Une solution est considérée comme voisine dès qu'au moins une valeur d'un des 6 paramètres est modifiée à hauteur de $\pm 10\%$ (bornes données en Tableau 1).

	$m_{capacity}$	m_{wave}	$m_{density}$	$ml_{capacity}$	ml_{wave}	$ml_{density}$
Bornes	[1200, 2700]	[10, 35]	[25, 175]	[1200, 2700]	[10, 35]	[25, 175]
Unité	veh/h/lane	km/h	veh/km/lane	veh/h/lane	km/h	veh/km/lane

TAB. 1 – Ensemble de définition des variables à optimiser.

Nous obtenons alors nos deux fonctions objectifs à minimiser (S_{MSE} et T_{MSE}) en faisant la somme des erreurs moyennes quadratiques (MSE) entre les jeux de données X et Y obtenus avec notre simulateur et les jeux de données X_N et Y_N fournis par Neovya.

$$S_{MSE} = \sum_{c=c_0}^{c_{max}} \sum_{t=t_0}^{t_{max}} MSE(x_t^c \in X,\ x_t^c \in X_N), \quad T_{MSE} = \sum_{c=c_0}^{c_{max}} \sum_{t=t_0}^{t_{max}} MSE(y_t^c \in Y,\ y_t^c \in Y_N)$$

2.2 Algorithmes multiobjectifs utilisés

Durant nos travaux, nous avons comparé les performances de deux algorithmes multi-objectifs sur notre problème : MOEA/D et *Hypervolume-Based MultiObjective Local Search* (HBMOLS). Plus précisément, nous utilisons la version de MOEA/D steady-state variant proposée dans [2]. Pour les paramètres de l'algorithme, nous utilisons 10 directions réparties uniformément, une distance de voisinage des sous-problèmes égale à 2, et un critère d'arrêt fixé à 100 itérations. HBMOLS [1] est une recherche locale multiobjective basée sur l'hypervolume comme indicateur de qualité pour les solutions. Dans nos expérimentations, la taille de la population est fixée à 10 individus. Sur notre scénario, environ 100 à 200 pas de recherche locale sont nécessaires pour atteindre le critère d'arrêt naturel.

3 Résultats préliminaires

Nous avons comparé les deux algorithmes pour le même nombre d'exécutions. L'ensemble des solutions obtenues ont été réunies, pour ne garder ensuite que le front de Pareto à des fins de comparaison (Figure 1). L'hypervolume de chaque population finale a également été calculé avec un point de référence situé en [3 000 000, 300 000] (Tableau 2). Selon ces résultats, HBMOLS serait plus efficace pour atteindre de bonnes solutions, cependant le temps d'exécution d'HBMOLS est assez variable et un seul paramétrage a été utilisé pour chaque algorithme.

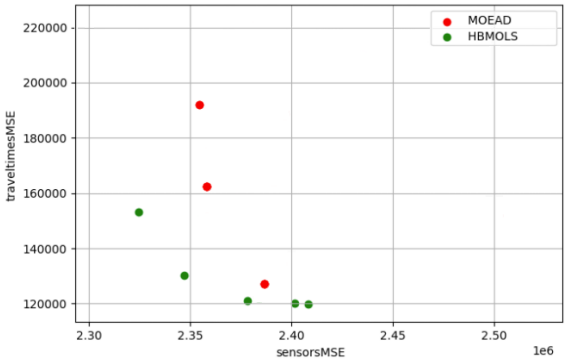

FIG. 1 – Front de Pareto par type d'algorithme.

	Hypervolume	Temps d'exécution [1]
MOEA/D	$9.8 * 10^{10}$	42min ~ 43min
HBMOLS	$1.1 * 10^{11}$	46min ~ 71min

TAB. 2 – Hypervolume moyen et temps d'exécution par algorithme.

4 Conclusions et perspectives

Pour la suite, nous prévoyons de tester d'autres représentations des solutions (par exemple, en individualisant les tronçons). Cela permettra d'opérer sur d'autres paramètres du réseau. Nous envisageons également d'étudier d'autres relations de voisinage et paramètrages des algorithmes d'optimisation. Davantage de résultats seront présentés lors de la conférence.

Références

[1] Matthieu Basseur, Rong-Qiang Zeng, and Jin-Kao Hao. Hypervolume-based multi-objective local search. *Neural Computing and Applications*, 21(8) :1917–1929, 2012.

[2] Qingfu Zhang and Hui Li. MOEA/D : A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6) :712–731, 2007.

1. Sur AMD Ryzen 5 4600H, 6 cores, 12 threads, 3 GHz (turbo 3.9 GHz)