Pratiques de modeleurs, un retour d'expérience avec OPL, les modeleurs sont ils dépassés par Julia JuMP et Python-MIP ?

Nicolas Dupin[1]

Laboratoire Interdisciplinaire des Sciences du Numérique, CNRS, Université Paris-Saclay, France
nicolas.dupin@lisn.upsaclay.fr

Mots-clés : *Optimisation Linéaire, Modeleurs, Applications industrielles, Enseignement*

Cet exposé fait suite à l'appel de partage de bonnes pratiques : "ceux qui ont souffert (en thèse) en utilisant des outils/langages mal adaptés pour la RO sont invités à présenter leur expériences et surtout leurs solutions". Pour ma part, j'ai été très satisfait d'utiliser OPL, le modeleur de Cplex, sur des problèmes d'optimisation industriels. Je partage ici des pratiques d'implémentation avec OPL qui m'ont été profitables. Cet exposé n'est pas un discours commercial pour OPL et Cplex, et n'engage que moi par rapport à mon utilisation de l'outil face aux problématiques qui étaient les miennes. J'ai découvert OPL assez tôt dans mon parcours, et utilisé pour les résultats numériques des publications [1, 2, 3, 4, 5, 6]. Des cadres d'utilisations adaptées pour OPL seront énoncés, mais aussi les limites de l'utilisation d'OPL dans ces différents contextes. Des alternatives avec d'autres modeleurs ou bibliothèques de modélisation de PLNE telles que Julia JuMP et Python-MIP apparus depuis seront également mentionnées.

Une version longue de ce papier est disponible au DOI 10.13140/RG.2.2.27455.43682, à l'url https://www.researchgate.net/publication/357630277_Pratiques_de_modeleurs_un_retour_d%27experience_avec_OPL_les_modeleurs_sont_ils_depasses_par_Julia_JuMP_et_Python-MIP

Les aspects discutés de l'intérêt d'un modeleur tel qu'OPL, en comparaison avec Julia JuMP et Python-MIP notamment portent sur les aspects suivant :

- Comparaisons et analyses de formulations PLNE [1, 6] ;
- Matheuristiques itérant des calculs de PLNE [3, 5, 6] ;
- Implémentation de méthodes de décomposition [2, 3, 4] ;
- Utilisation pour l'enseignement.

Les conclusions de ces réflexions et de ces retours d'expérience sont les suivantes :

On peut revenir sur certaines idées reçues sur les modeleurs et sur OPL en particulier. On entend parfois dire que ces outils sont exclusivement à réserver pour l'enseignement. Il est indéniable que les modeleurs sont très adaptés au cadre de l'enseignement comme cela a été évoqué précédemment. OPL a comme tout autre modeleur un potentiel industriel, j'ai déjà vu des praticiens utiliser OPL pour résoudre des PLNE d'application opérationnelle, en utilisant les fonctionnalités décrites de la section 1. Les types d'utilisation présentées en section 2 (et également en section 3 dans une moindre mesure), ont également un potentiel fort de prototypage : un modeleur comme OPL permet d'analyser avec un faible temps de développement des variantes de modélisations et de matheuristiques, pour fournir des résultats sur le bon design de l'approche à implémenter finement et soigneusement, en évitant une telle implémentation pour des opérateurs ou variantes non sélectionnés dans l'approche finale. C'est dans cet esprit que ma thèse a été réalisée et rédigée, ce sont des pratiques que j'encourage pour de l'application industrielle.

Un autre questionnement pertinent et récurrent est d'établir si les modeleurs ont encore un intérêt en 2020 avec notamment l'essor de Python, qui répond à un besoin similaire en terme de concision et de facilité d'utilisation. Il est vrai que de nombreuses utilisations peuvent être réalisées indifféremment avec OPL, Julia JuMP ou Python-MIP. Cet exposé a présenté des avantages et inconvénients assez différents, et le choix du modeleur ou de la bibliothèque de modélisation PL/PLNE peut se faire en fonction du besoin spécifique de l'étude. Utiliser Julia JuMP permet d'utiliser des bibliothèques développées spécifiquement en Julia par des équipes de recherche de pointe (optimisation continue, black-box, optimisation robuste, génération de colonne automatique ...). Julia me semble être un excellent choix pour des travaux de recherche en RO. Pour de l'application industrielle, avoir un code sous Python peut présenter de nombreux avantages, notamment si d'autres bibliothèques de Python sont utilisées, ou de par la popularité du langage, induisant une bonne capacité de maintenabilité également. Python-MIP pour de la PL/PLNE ou Pyomo si on souhaite également utiliser des solveurs non linéaires peuvent être préconisés dans un tel cadre. Lorsque des callbacks d'heuristiques primales ou de génération de coupes sont à l'étude, ou lorsqu'un solveur libre doit être appelé, Python-MIP ou Julia JuMP ont sans aucune contestation l'avantage sur OPL. A l'heure actuelle, OPL me semble intéressant spécifiquement pour sa décomposition de Benders automatisée, mais également par sa facilité d'utilisation depuis un code C++ pour utiliser de manière concise une résolution PL/PLNE, et avoir un codage de bas niveau, une parallélisation efficace pour des phases de recherche locales standards ou des résolutions de sous problèmes par programmation dynamique.

Références

[1] N. Dupin. Tighter MIP formulations for the discretised unit commitment problem with min-stop ramping constraints. *EURO Journal on Computational Optimization*, 5(1) :149–176, 2017.

[2] N. Dupin. Column generation for the discrete UC problem with min-stop ramping constraints. *IFAC-PapersOnLine*, 52(13) :529–534, 2019.

[3] N. Dupin, R. Parize, and E. Talbi. Matheuristics and column generation for a basic technician routing problem. *Algorithms*, 14(11) :313, 2021.

[4] N. Dupin and E. Talbi. Machine learning-guided dual heuristics and new lower bounds for the refueling and maintenance planning problem of nuclear power plants. *Algorithms*, 13(8) :185, 2020.

[5] N. Dupin and E. Talbi. Parallel matheuristics for the discrete unit commitment problem with min-stop ramping constraints. *International Transactions in Operational Research*, 27(1) :219–244, 2020.

[6] N. Dupin and E. Talbi. Matheuristics to optimize refueling and maintenance planning of nuclear power plants. *Journal of Heuristics*, 27 :63–105, 2021.