Sujet de TER 2018-2019

Xavier Gandibleux (Université de Nantes)
Anthony Przybylski (Université de Nantes)

Octobre 2018

Julia, JuMP et programmation linéaire multi-objectif

1 Contexte

Le travail de TER proposé s'inscrit dans le contexte de l'étude, la conception et la réalisation de "vOptSolver" [3, 4], un environnement gratuit et open-source (`https://github.com/vOptSolver`) permettant la modélisation, la résolution et l'analyse de MOMILP, problèmes d'optimisation linéaires en variables mixtes comportant 2 ou 3 objectifs. "vOptSolver" est construit autour de deux packages et se présente comme un backbone. Il dispose d'un langage de modélisation permettant d'exprimer un problème donné, et un ensemble de solveurs spécifiques et génériques permettant de résoudre des problèmes allant jusqu'à relever de la classe des MOMILP.

"vOptGeneric" est un des deux packages du projet "vOptSolver"; il est codé en Julia [1] (actuellement compliant avec la version 0.6.4 du langage) et a été intégré à la collection de packages du langage en 2017. Il a pour objectif de calculer X_E, un ensemble complet de solutions efficaces, pour un problème linéaire multi-objectif sans structure a priori, pouvant aller jusqu'à la classe des MOMILP. Le langage de modélisation est dérivé de JuMP [5] étendu à l'optimisation multi-objectif, et comporte un ensemble d'algorithmes de résolution génériques faisant appel si besoin à un solveur MIP open-source ou commercial.

A ce jour, un solveur de type simplexe multi-objectif open-source couplé à un langage de modélisation algébrique n'est pas disponible dans "vOptSolver" et à notre connaissance, il n'existe pas dans la discipline de réalisation informatique répondant à cette ambition. Cette absence est d'autant plus sensible que l'on constate un nombre sans cesse croissant de situations qui demandent de prendre en compte simultanément plusieurs objectifs de différentes natures (économiques, qualité de service, environnementaux, etc.) souvent non commensurables, au cours d'un processus d'optimisation. Le travail d'étude et de recherche vise donc à combler ce manquement sur base de vOptSolver, Julia 1.0 et JuMP 0.19.

Le candidat aura des connaissances solides en optimisation (modélisation, programmation linéaire en variables continues, algorithmes, calcul scientifique, moteur de résolution LP) et une aisance avérée pour la programmation d'algorithmes en langage Julia dans un environnement linux/ubuntu. Par contre, il n'est pas nécessaire d'avoir des connaissances en optimisation multiobjectif à cette étape, les étudiants retenus seront formés aux éléments de bases qui seront nécessaires.

2 Méthode du simplexe paramétrique en forme révisée et avec des variables bornées

Dans ce lot de travaux, nous nous intéressons à la classe des Programmes Linéaires comportant uniquement deux fonctions objectifs. Le cas bi-objectif permet l'utilisation d'hypothèses spécifiques menant à des méthodes de résolution généralement beaucoup plus simples et efficaces que dans le cas multi-objectif (i.e. trois objectifs et plus). La Programmation Linéaire ne fait pas exception, la méthode du simplexe paramétrique (très bien décrite dans le chapitre 6 de [2]) ne s'adresse qu'au cas bi-objectif. Son principe est simple et consiste en une énumération des solutions de base efficaces du problème. Il est établi que ces

solutions de base efficaces sont adjacentes. On part donc de l'une d'entre elles pour énumérer toutes les autres en faisant les bons choix de variables entrante et sortante. De plus, ces solutions de base sont simplement énumérées par valeur croissante de la première fonction objectif, et décroissante pour la seconde. Il n'y a jamais de retour en arrière, ni de bifurcation dans l'algorithme.

Le but de ce lot de travaux est de réaliser une implémentation efficace de cette méthode. Cela signifie que l'on va s'écarter de la version pédgogique (i.e. consistant à calculer entièrement le tableau simplexe à chaque itération), pour passer à la forme révisée, et que nous ne considérerons pas les bornes sur les variables comme des contraintes. Ce lot de travaux fait donc directement suite aux distanciels 1 et 2 de l'UE "Optimisation discrète et combinatoire". Nous ne pourrons que très partiellement nous appuyer sur une implémentation existante de l'algorithme du simplexe (mono-objectif). En effet, nous considérons une fonction objectif de plus que ce que gère un solveur, et nous devons donc maintenir les coûts réduits pour une seconde fonction objectif. De plus, l'interface des solveurs permet éventuellement de définir des règles de choix de variables entrante et sortante, menant à une solution de base optimale, mais pas de choisir manuellement des variables entrante et sortante pour stocker des bases intermédiaires. Il sera donc nécessaire de maintenir manuellement la décomposition LU de la matrice de base, pour ensuite reconstruire les données utiles du tableau simplexe. Heureusement, certains traitements algébriques déjà implémentés dans Julia pourront être très utiles.

3 Méthode du simplexe multi-critère

Dans ce lot de travaux, nous nous intéressons à la classe des Programmes Linéaires comportant trois fonctions objectifs et plus. Les hypothèses spécifiques du cas bi-objectif ne sont donc plus valables. La méthode du simplexe paramétrique n'est donc plus applicable. Pour résoudre des Programmes Linéaires avec trois fonctions objectifs et plus, la méthode du simplexe multi-critère (décrite dans le chapitre 7 de [2]) peut être utilisée. La propriété indiquant que les bases efficaces sont adjacentes reste vraie. On peut donc toujours partir de l'une d'entres elles pour énumérer toutes les autres en faisant les bons choix de variable entrante et sortante. Cependant, l'énumération n'est plus simplement ordonnée comme dans le cas bi-objectif. En partant d'une base initiale efficace, plusieurs choix de variable entrante sont possibles pour obtenir une autre base efficace. Par conséquent, des retours en arrière seront nécessaires dans l'algorithme. De plus, il est également possible de cycler parmi les solutions de base efficaces (indépendamment de la dégénérescence du problème), et un mécanisme évitant ce cyclage sera nécessaire.

L'algorithme tel que décrit dans le chapitre 7 de [2] demandera des efforts pour mener à une implémentation, même en se contentant d'une version "tableau" de l'algorithme. Le but de ce lot de travaux est donc juste de réaliser une implémentation "simple". Pour les mêmes raisons que pour la méthode du simplexe paramétrique, l'utilisation que l'on pourra faire d'un solveur mono-objectif restera limitée.

4 Extensions possibles

Suivant les avancements réalisés dans les deux lots de travaux et le temps disponible, des extensions seront possibles. En particulier, la décomposition LU de la matrice de base ainsi que sa mise à jour utilisée dans le cas bi-objectif, pourront être intégrés dans le cas multi-objectif. Ensuite, les deux lots de travaux s'adressent à la résolution de problèmes initialement complètement posés. On pourra se poser la question de la réoptimisation de ce type de problème, par exemple en cas d'ajout d'une contrainte. Comme la théorie de la dualité ne se généralise pas facilement au cas multi-objectif, il s'agit d'une question encore largement ouverte.

5 Résumé

Tâches :

- étude des documents remis et des resources indiquées
- conception, implémentation et validation d'algorithme d'optimisation
- intégration du développement logiciel dans la plateforme existante
- documentation des productions sur la plateforme existante

- rédaction d'un rapport scientifique détaillé sous latex
- présentation des travaux à l'occasion d'une soutenance publique

Mots-clef : optimisation linéaire, optimisation continue, optimisation multiobjectif, langage de modélisation, solveur, Julia, JuMP.

Prérequis : optimisation discrète et combinatoire — langage de programmation de haut niveau.

Nombre d'étudiants maximum pour ce sujet : 4

Lieu des travaux : FST/LS2N/Bâtiment 11

Outils utilisés : langage de programmation Julia, git sur GitHub, latex, markdown

References

[1] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A Modeling Language for Mathematical Optimization. *SIAM Review*, 59(2):295–320, 2017.

[2] Matthias Ehrgott. *Multicriteria Optimization*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[3] Xavier Gandibleux, Gauthier Soleilhac, Anthony Przybylski, Flavien Lucas, Stefan Ruzika, and Pascal Halffmann. vOptSolver, a "get and run" solver of multiobjective linear optimization problems built on Julia and JuMP, 2017. MCDM2017: 24th International Conference on Multiple Criteria Decision Making. July 10-14, 2017. Ottawa (Canada).

[4] Xavier Gandibleux, Gauthier Soleilhac, Anthony Przybylski, and Stefan Ruzika. vOptSolver: an open source software environment for multiobjective mathematical optimization, 2017. IFORS2017: 21st Conference of the International Federation of Operational Research Societies. July 17-21, 2017. Quebec City (Canada).

[5] Miles Lubin and Iain Dunning. Computing in operations research using Julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.