Metrics : un unique outil pour l'analyse d'expérimentations

Thibault Falque, Romain Wallon, Hugues Wattez

Metrics : un unique outil pour l'analyse d'expérimentations

Thibault Falque[1,2], Romain Wallon[2], Hugues Wattez[2]

[1] Exakis Nelite, France
[2] CRIL, Université d'Artois & CNRS, 62300 Lens, France
{falque,wallon,wattez}@cril.univ-artois.fr

Mots-clés : *analyse statistique, expérimentation, reproductibilité.*

Dans le cadre du développement de solveurs, tels que ceux de programmation linéaire en nombres entiers ou de programmation par contraintes, il est souvent nécessaire de réaliser des expérimentations pour s'assurer que ces solveurs fonctionnent comme prévu. En particulier, il est important de vérifier que les ressources utilisées par ces programmes restent raisonnables. Dans ce but, différentes solutions logicielles telles que *runsolver* [5] ont été proposées, à la fois pour mesurer et limiter l'utilisation des ressources temporelles et spatiales par le programme en cours d'évaluation. Cependant, respecter les limites fixées est en général insuffisant pour évaluer le comportement du programme. Il est souvent nécessaire de collecter des statistiques supplémentaires, qui peuvent être fournies par le programme lui-même (par exemple, via ses *logs*) ou par l'environnement d'exécution (par exemple, *runsolver*). Les données collectées doivent ensuite être agrégées pour évaluer la qualité des résultats du programme au travers d'une analyse statistique.

Dans ce domaine, de nombreux outils mathématiques peuvent être utilisés, et faire un choix parmi ceux-ci peut introduire des biais dans les résultats ou leur analyse. Il est donc important d'appliquer les principes de *science ouverte* et de *reproductibilité* pour permettre de rejouer l'analyse des résultats. Ainsi, ces principes ont fait l'objet d'une recommandation de l'OCDE [4], et des chercheurs ont déjà introduit diverses approches favorisant la reproductibilité dans le contexte d'expérimentations logicielles [2, 3]. En particulier, il est recommandé de rendre disponible le code source des logiciels étudiés (ou, *a minima*, leurs exécutables sous forme binaire), et de diffuser les données utilisées pour les évaluer (par exemple, dans des forges logicielles). Il est également important de rendre l'analyse des résultats reproductibles, en utilisant des outils tels que le *RMarkdown* ou les *notebooks Jupyter*.

Dans le cas particulier des solveurs, il y a souvent peu de différences sur la manière d'exécuter ces derniers : ils proposent en effet des interfaces en ligne de commande qui doivent respecter les règles imposées par l'environnement dans lequel ils sont exécutés (par exemple, durant les compétitions). De plus, la plupart des données collectées lors de l'exécution des solveurs restent le plus souvent les mêmes (par exemple, le temps d'exécution, l'utilisation de la mémoire, etc.). Dans ce contexte, la création d'un outil permettant d'exécuter le programme, de collecter les données qu'il produit et de les analyser présenterait plusieurs avantages : tester de nouvelles fonctionnalités serait plus simple, à la fois en termes d'exécution et d'analyse, et la reproductibilité des résultats serait systématiquement assurée.

Partant de ces observations, nous présentons ici *Metrics* (*mETRICS* signifie *rEproducible sofTware peRformance analysIs in perfeCt Simplicity*), une bibliothèque Python sous licence LGPL (`https://github.com/crillab/metrics`) visant à unifier la manière d'expérimenter les solveurs et à soulager les souffrances liées à l'analyse des résultats obtenus [1]. L'ambition de *Metrics* est de fournir une chaîne complète d'outils de l'exécution du solveur à l'analyse de ses performances. Actuellement, cette bibliothèque possède deux composants principaux : *Scalpel* (*sCAlPEL* signifie *extraCting dAta of exPeriments from softwarE Logs*) et *Wallet* (*wALLET* signifie *Automated tooL for expLoiting Experimental resulTs*). Dans un premier temps, *Scalpel* est conçu pour simplifier la récupération des données produites par les expérimentations, en

permettant à la fois la collecte de statistiques liées à la consommation des ressources par le solveur (souvent fournies par l'environnement d'exécution), mais également celle d'informations propres à l'exécution de l'algorithme implanté par le solveur (nombre d'affectations ou d'itérations, solution(s) obtenue(s), etc.). Dans ce but, *Scalpel* est capable de lire une grande variété de formats, incluant les formats CSV, XML, JSON ou encore la sortie brute produite par le solveur, qui peut être décrite par l'utilisateur dans un fichier de configuration. Cette approche fait de *Scalpel* un outil à la fois flexible et simple à configurer. Dans un second temps, les données expérimentales obtenues par *Scalpel* peuvent être exploitées dans *Wallet*. Ce module propose une interface simple d'utilisation pour tracer des diagrammes souvent utilisés pour l'analyse de solveurs (tels que les *scatter plots* et les *cumulative distribution functions*) et pour calculer diverses statistiques relatives à leur exécution (en particulier, leurs scores en utilisant différentes mesures classiques). La conception de *Wallet* facilite l'intégration de l'analyse dans des *notebooks Jupyter* qui peuvent être facilement partagés en ligne (par exemple, *GitHub* ou *GitLab* sont capables d'afficher de tels fichiers), favorisant également la reproductibilité de l'analyse. En effet, la distribution des données expérimentales et de ces *notebooks* permet à tout un chacun de pouvoir rejouer l'analyse des résultats (et donc, de s'assurer de son intégrité), mais également de la compléter suivant ce qu'il estime nécessaire pour vérifier les conclusions de cette même analyse.

Dans cette présentation, nous illustrerons les différentes capacités de *Metrics* dans le cadre de l'étude expérimentale de plusieurs solveurs de programmation linéaire en nombres entiers. Nous montrerons notamment comment extraire les données pertinentes issues de l'exécution de ces solveurs, pour ensuite tracer différentes figures permettant d'exploiter les résultats obtenus. Plus précisément, nous montrerons comment de telles figures peuvent faciliter la comparaison des solveurs considérés, en permettant une interprétation visuelle de leurs performances.

Références

[1] Thibault Falque, Romain Wallon, and Hugues Wattez. Metrics : Mission Expérimentations. In *Actes des 16es Journées Francophones de Programmation par Contraintes (JFPC'21)*, 2021.

[2] Juliana Freire, Norbert Fuhr, and Andreas Rauber. Reproducibility of Data-Oriented Experiments in e-Science (Dagstuhl Seminar 16041). *Dagstuhl Reports*, 6(1) :108–159, 2016.

[3] Yang-Min Kim, Jean-Baptiste Poline, and Guillaume Dumas. Experimenting with reproducibility : a case study of robustness in bioinformatics. *GigaScience*, 7(7) :giy077, July 2018.

[4] Dirk Pilat and Yukiko Fukasaku. OECD Principles and Guidelines for Access to Research Data from Public Funding. *Data Science Journal*, 6 :4–11, 06 2007.

[5] Olivier Roussel. Controlling a Solver Execution : the runsolver Tool. *Journal on Satisfiability, Boolean Modeling and Computation*, 7 :139–144, 2011.