Analyse expérimentale de la complexité temporelle des algorithmes

David Colliard, Hadrien Cambazard, Nicolas Catusse

Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, F-38000 Grenoble, France
{prenom.nom}@grenoble-inp.fr, david.colliard@grenoble-em.com

Mots-clés : *Complexité temporelle, évaluation automatique, formation, régression linéaire.*

Caseine est une plateforme pédagogique qui permet d'évaluer automatiquement les codes informatiques écrits par les étudiants afin qu'ils puissent travailler de manière autonome. Actuellement, les étudiants ont principalement un retour sur la justesse de leur réponse par l'utilisation de tests écrits par l'enseignant, et nous cherchons à étendre cette évaluation automatique à des critères de qualité. Pour cela, nous présentons une méthode expérimentale pour estimer la complexité temporelle théorique d'un algorithme proposé par un étudiant.

1 Contexte

La plateforme pédagogique caseine.org permet entre autres la soumission de code informatique par les étudiants et le passage de test automatique défini par l'enseignant. Le code de l'étudiant est transmis du serveur web à un serveur d'exécution. Ce serveur exécute l'ensemble des tests définis par l'enseignant sur ce code. Il transmet ensuite le résultat au serveur web, et l'étudiant peut prendre connaissance du résultat de cette évaluation.

L'analyse de la complexité algorithmique est utilisée pour évaluer la quantité de ressources (temps ou espace) nécessaire à l'exécution d'un algorithme en fonction de la taille de l'entrée. Ici, nous nous intéressons plus particulièrement à la complexité temporelle moyenne. Les étudiants soumettent leur programme en ligne et nous souhaitons leur donner une indication sur la complexité de leurs algorithmes. De son côté, l'enseignant indique une complexité attendue. Si elle est différente de celle déterminée pour le programme de l'étudiant, un message invite l'étudiant à contacter l'enseignant pour vérifier la complexité de son approche.

2 Méthodologie

Pour analyser la complexité temporelle de l'algorithme soumis par l'étudiant, nous mesurons le temps d'exécution en pratique de son programme. Le programme est exécuté plusieurs fois, pour différentes tailles d'entrées. Soit $Y = (y_1, ...y_m)$ les différentes mesures de temps pour les différentes tailles d'entrée. Nous calculons ensuite les régressions linéaires avec différentes classes de complexité prédéfinies :

- $H_\theta^1 = \theta_0 + \theta_1 log(n)$
- $H_\theta^2 = \theta_0 + \theta_1 n$
- $H_\theta^3 = \theta_0 + \theta_1 n log(n)$
- $H_\theta^4 = \theta_0 + \theta_1 n^2$
- $H_\theta^5 = \theta_0 + \theta_1 n^3$

Pour évaluer la qualité d'une régression linéaire, nous utilisons le coefficient de détermination $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$ avec $\hat{y}_i$ les valeurs prédictives et $\bar{y} = \frac{\sum_{i=1}^{m} y_i}{m}$ la moyenne des observations. Les valeurs des R^2 obtenues pour les différentes régressions sont comparées pour déterminer la régression la plus appropriée. On considère que la complexité est identifiée quand l'une des

régressions possède un R^2 proche de 1 et que l'écart au deuxième meilleur R^2 est suffisamment important. Tant qu'aucune régression ne satisfait à ces critères, de nouvelles mesures de temps pour des tailles plus importantes sont effectuées et les régressions sont mises à jour.

3 Résultats expérimentaux

Le choix du critère d'arrêt de la procédure de test est critique pour la qualité du résultat. Augmenter la taille des données d'entrée permet souvent d'améliorer la véracité de la réponse, mais demande un temps de calcul plus important et de l'espace mémoire supplémentaire. Dans un contexte d'interaction avec l'utilisateur via un site internet, on peut difficilement se permettre un temps de réponse supérieur à quelques secondes. En pratique, nous obtenons des bonnes qualités de résultat pour discriminer des complexités $O(n)$, $O(n^2)$ et $O(n^3)$. En revanche, il est difficile de différencier des complexités $O(n \log n)$, $O(n)$, $O(\log n)$ et $O(1)$ car cela nécessite une taille de données exponentielle en entrée. Afin de diminuer le temps d'exécution du calcul, nous mettons en oeuvre des calculs en parallèle (multithreading). La robustesse est primordiale, un contrôle fin du temps d'exécution laissé aux mesures est implémenté pour éviter une charge serveur trop importante. Les premiers résultats expérimentaux obtenus avec cet outil sont présentés dans le tableau 3 sur plusieurs milliers de tests.

Complexité de l'algorithme soumis	Taux de succès	Temps	R^2 moyen
$O(n)$	98,2	1482 ms	0,98
$O(n^2)$	91,975	1649 ms	0,99
$O(n^3)$	98,5	609 ms	0,99

FIG. 1 – Résultats expérimentaux