Évaluer des performances: un cours rapide et évident mais que personne ne respecte!

Optimisation 2

Dimitri Watel

dimitri.watel.free.fr

Tests unitaires / d'intégration

Définition

Un test unitaire vérifie qu'un programme gère des cas particuliers. Cela concerne généralement une petite partie d'un gros programme.

Définition

Un test d'intégration vérifie des paires d'entrée/sortie d'un programme. Cela concerne généralement un gros programme dans son ensemble.

Plus type

- Tests partiels : on génère un sous-ensemble des instances vérifiant ce cas
- Tests complets : on génère toutes les instances vérifiant ce cas
- Tests *au fur et à mesure* : si on détecte un bug, on rajoute un test unitaire.

Evaluation

Définition

Une évaluation permet de déterminer les performances (les forces et les faiblesses) d'un algorithme.

Un test unitaire n'est pas une évaluation ! ! !

Évaluer les performances d'un algorithme

Il faut se poser les bonnes questions

- quelles sont les performances théoriques de mon algorithme ?
- sur quelles catégories d'instances vais-je tester mes algorithmes ?
- sur combien d'instances vais-je tester mes algorithmes ?
- quelles sont les mesures à faire ?
- sur quelle machine les tests ont-il été faits ? Seront-il reproductibles facilement ?

Performances théoriques

Mesure des performances théoriques

Généralement, on regarde

- la complexité en temps
- la complexité en espace
- le nombre de variables ou de contraintes pour un programme mathématique
- le ou les pires cas
- le ou les cas où l'algorithme est optimal
- le ou les cas où l'algorithme est "presque" optimal
- la complexité algorithmique, paramétrée ou l'approximabilité du problème
- ...

Catégories d'instances

Un algorithme ne fonctionne pas de la même manière sur toutes les instances. L'évaluation peut permettre de déterminer pour quelles instances il est efficace et pour lesquelles il vaut mieux se tourner vers un autre algorithme.

Exemples

Par exemple

- un graphe avec peu d'arêtes
- un graphe avec peu de cycles
- un graphe complet ou quasi-complet
- un graphe parfait
- ...

Combien d'instances

Etant donnée l'infinité des instances, des tests sur des instances fixées ne sont pas une preuve exacte de l'utilité de votre algorithme. Il s'agit au mieux d'une preuve statistique.

- pensez à la représentativité statistique de vos instances : combien de paramètres devez vous faire varier ? les tailles croissent-elles linéairement, logarithmiquement, exponentiellement ? combien d'instances d'une même taille ?
- pensez à faire plusieurs fois les tests pour des algorithmes probabilistes
- pensez aussi au temps de calcul de votre machine, avez-vous le temps de tout faire avant la date limite ?

Mesures

Mesurer les résultats

Mesures habituelles :

- Temps de calcul
- Valeur de la solution
- Comparaison à l'optimal si celui-ci est connu (si heuristique)
- Combien de fois l'optimal est atteint ? (si heuristique)
- Jusqu'à quelle taille l'algorithme donne une réponse en temps raisonnable ?
- Pour un programme mathématique : combien d'inégalités valides ont été ajoutées, quelle amélioration en temps ? sur la borne de la relaxation continue ?
- Pour un branch and bound et un programme mathématique : combien de branchements ont été effectués, à quelle vitesse la borne se rapproche-t-elle de l'optimal ?
- ...

Présenter les résultats

Pensez à comment regrouper les instances pour faire des moyennes, des écart-types, des quartiles, ...

Faites des courbes, des tableaux synthétiques, des histogrammes, des boîtes à moustaches.

Reproductibilité

Donnez toutes les informations possibles de votre machine (entre autres le processeur, RAM, ...). Le code source existe-t-il ? est-il disponible ? Les instances testées sont-elles disponibles ?

Si aléatoire, quelles graines ont été utilisées ? Le résultat est-il bien meilleur si la graine n'est pas fixée ?