Avant-propos

L'essentiel

– On peut classer les problèmes d'un jeu d'essai selon plusieurs mesures de difficultés cohérentes entre elles, c'est-à-dire donnant approximativement la même relation d'ordre.

– Ces mesures sont valides pour les optimiseurs qui supposent, explicitement ou implicitement que « plus près c'est mieux », en probabilité.

– Certaines peuvent être estimées par une méthode de Monte-Carlo, lorsque que l'on ne connait pas la structure des paysages des problèmes du jeu d'essai (bassins d'attraction, plateaux, minima locaux). Les calculs peuvent cependant être très longs.

– L'une des mesures présentées nécessite de se donner un seuil d'erreur acceptable, mais son codage ne prend que quelques lignes. L'autre, un peu plus complexe car à base de triplets de positions, ne nécessite pas de définir *a priori* un seuil d'erreur, mais est moins fiable en présence de plateaux.

– Un autre indicateur de difficulté, dit δ_0, est proposé, fondé sur la structure du paysage du problème. Il est cohérent avec les précédents et rapide à calculer, mais nécessite de connaitre les bassins d'attraction et les plateaux du paysage du problème.

– Cette information étant rarement donnée et difficile à trouver pour les jeux d'essai classiques, le programme LandGener est spécifiquement conçu pour créer des paysages de structure connue, sur lesquels δ_0 est facilement calculable.

– Une typologie de tous les paysages possibles est établie, essentiellement en termes de modalité et de ratio de plateaux, avec l'estimation de l'importance relative de chaque classe.

– Quelques jeux d'essais classiques sont analysés en fonction de cette typologie. On montre qu'ils sont biaisés en faveur de certains types de problèmes, par exemple unimodaux ou sans plateaux. L'application d'une mesure de difficulté permet cependant de classer leurs problèmes, du plus facile au plus sélectif.

– Des définitions rigoureuses des notions d'exploitation et d'exploration sont données, les rendant mesurables. Les principaux types d'évolution de leur rapport (profils de balance) sont étudiés.

Introduction

Les difficultés augmentent à mesure qu'on approche du but.

Johann Wolfgang von Goethe

Il y a bien moins de difficultés à résoudre un problème qu'à le poser.

Joseph de Maistre

De la fiabilité des jeux d'essai

Tous les ans, pour ne pas dire tous les mois, est proposé un nouvel algorithme d'optimisation, dont l'inventeur proclame la supériorité sur les précédents. Ce genre d'affirmation est fondé sur des résultats de tests obtenus grâce à un jeu d'essai, c'est-à-dire une sélection de problèmes auxquels l'algorithme est appliqué.

Dans (Clerc 2015) j'ai montré qu'il était facile de choisir un jeu d'essai et la manière dont les résultats sont traités pour apparemment « justifier » cette supériorité.

Considérons par exemple le jeu d'essai du tableau I.1, qui a été effectivement utilisé dans un article[1]. Sur un tel ensemble de problèmes il suffit de définir un algorithme dont la signature (voir section A.3) est biaisée en faveur du centre de l'espace de recherche pour obtenir de bons résultats. À l'extrême, si

1. Je ne donne pas la référence, car il ne semble pas utile d'augmenter le nombre de citations de cet article. Notons d'ailleurs que d'autres très semblables ont été également publiés.

l'on ne veut voir l'algorithme que comme une boîte noire dans laquelle on introduit le jeu d'essai et qui propose une solution, il existe une boîte « magique » qui trouve la solution parfaite pour chacune des fonctions en un temps quasi nul[2].

Nom	Formule
Sphere	$\sum_{d=1}^{D} x_d^2$
Rosenbrock	$\sum_{d=1}^{D-1} \left(100 \left(x_d^2 - x_{d+1}\right)^2 + \left(1 - x_d\right)^2\right)$
Ellipsoid	$\sum_{d=1}^{D} \left(\lvert x_d \rvert + 0,5\right)^2$
Rastrigin	$\sum_{d=1}^{D} \left(x_d^2 - 10\cos\left(2\pi x_d\right) + 10\right)$
Ackley	$-20e^{-0,2\sqrt{\frac{1}{n}\sum_{d=1}^{D} x_d^2}} - e^{\frac{1}{n}\sum_{d=1}^{D}\cos(2\pi x_d)} + 20 + e$

Tableau I.1. *Un jeu d'essai biaisé dans un article publié (en anglais). Les noms des fonctions ont été conservés*

Naturellement, si un utilisateur, séduit par les conclusions optimistes de la présentation de l'algorithme, tente de l'appliquer à ses propres problèmes, il obtiendra presque sûrement de très mauvais résultats.

Ce manque de fiabilité est dû au fait que le jeu d'essai n'est pas représentatif des problèmes que l'utilisateur sera amené à résoudre.

Par ailleurs, en pratique, il est intéressant qu'un jeu d'essai contienne des problèmes de différents niveaux de difficulté. En effet, cela permet de mieux hiérarchiser les algorithmes testés sur ce jeu d'essai, sachant qu'un utilisateur ne préférera pas forcément l'algorithme capable de résoudre les problèmes les plus difficiles si les siens ne le sont pas autant et que, de plus, cet algorithme est très coûteux en ressources informatiques. Une définition précise du terme « difficulté » est dès lors nécessaire. Ceci est loin d'être trivial, car, comme le montre le petit exemple ci-dessus, la difficulté d'une optimisation dépend de l'optimiseur utilisé.

Ainsi, après avoir lu cet ouvrage, vous devriez être capable, en principe :

– d'analyser la structure d'un jeu d'essai et d'en déduire quels types d'optimiseurs il favorise ;

2. L'algorithme contenu dans la boîte est simplement : évaluer le point $(0\,;\,0\,;\,\ldots\,;0\,)$; évaluer le point $(1\,;\,1\,;\,\ldots\,;\,1)$; proposer comme solution le meilleur des deux.

– de construire votre propre jeu d'essai, avec des caractéristiques données (classe d'optimiseurs concernée, niveaux de difficulté) ;

– de porter un jugement critique bien fondé sur les optimiseurs présentés dans la littérature.

Guide de lecture

Cet ouvrage, court mais dense (méfiez-vous !), est en fait un recueil d'études liées à la question « comment comparer de façon fiable des optimiseurs ? ». Du coup, les chapitres peuvent être lus presque indépendamment les uns des autres. Il est cependant conseillé de commencer par le chapitre des définitions, certaines n'étant pas tout à fait classiques.

En fonction de votre connaissance du sujet et de vos centres d'intérêt, vous pouvez donc, par exemple, consulter directement le chapitre 6, même si, pour sa toute dernière section, il est nécessaire d'avoir vu auparavant une mesure de difficulté présentée dans le chapitre 2.

De même, si vous êtes surtout un praticien curieux de voir comment créer des paysages « sur mesure », le chapitre 4 sur le programme LandGener est fait pour vous. À l'inverse, le chapitre sur la typologie des paysages, rempli de formules mathématiques, risque de vous sembler bien rébarbatif et vous pourrez vous contenter d'en parcourir les conclusions, voire de l'ignorer complètement.

Et, bien sûr, les différentes sections de l'annexe sont encore plus indépendantes et non essentielles, même si elles peuvent apporter des informations utiles, comme des exemples de fonctions trompeuses et de petits codes sources. Au sujet de ceux-ci, d'ailleurs, il faut noter que les principaux ne sont pas donnés ici mais peuvent être téléchargés (http://www.iste.co.uk/codes_optimiseurs_iteratifs.zip).

Plus complètement, voici quelques remarques sur chacun des chapitres (hors annexe) :

– chapitre 1 : « Quelques définitions ». Les plus importantes sont celles de bassin et de plateau et de structure, mais si vous envisagez d'éviter le chapitre 3, vous pouvez, dans un premier temps, également sauter ce chapitre et vous contenter de notions intuitives ;

– chapitre 2 : « Difficulté de la difficulté ». Les mesures de difficulté qui seront ensuite utilisées un peu partout sont définies ici. Un chapitre à lire presque nécessairement, donc ;

– chapitre 3 : « Typologie des paysages ». À sauter si vous êtes allergique aux calculs combinatoires. Néanmoins, jeter un coup d'œil sur les conclusions peut être utile ;

– chapitre 4 : « LandGener ». C'est essentiellement la présentation de principes permettant la création de paysages « sur mesure » – et, donc, de jeux d'essai – avec quelques exemples ;

– chapitre 5 : « Jeux d'essai ». L'analyse critique de deux jeux d'essais classiques. Elle est faite en fonction de la typologie vue au chapitre 3, mais il n'est pas vraiment nécessaire d'avoir lu ce dernier... si vous faites confiance à ses conclusions ;

– chapitre 6 : « Difficulté *versus* dimension ». Une petite étude autour de l'idée, pas toujours juste, que la difficulté augmente avec la dimension (le nombre de variables) du problème ;

– chapitre 7 : « Exploitation et exploration *versus* difficulté ». En fait, ce chapitre et le suivant forment un tout. Les notions d'exploitation et d'exploration, souvent juste qualitatives, sont formellement définies et mesurables. Les évolutions possibles de leur rapport (profils de balance) sont étudiées ;

– chapitre 8 : « L'algorithme Explo2 ». On montre, juste à titre d'exemple, qu'il est possible d'écrire un petit optimiseur utilisant explicitement un profil de balance prédéfini, et qu'il est relativement efficace même si gourmand en temps de calcul. C'est sans doute le chapitre le plus susceptible de développements fructueux ;

– chapitre 9 : « Balance et difficulté perçue ». Un peu hétérogène, mais les deux études expérimentales qu'il contient sont trop courtes pour justifier des chapitres spécifiques. L'une concerne l'influence du profil de balance sur les performances et l'autre permet d'insister sur le fait qu'une mesure de difficulté ne donne qu'une relation d'ordre sur un ensemble de problèmes, sans préjuger des écarts quantitatifs entre les difficultés réellement perçues par un optimiseur donné.