IN104 - Projet informatique
Introduction aux algorithmes génétiques

Vladimir Paun
d'après un sujet de Bruno Monsuez
U2IS - ENSTA ParisTech

Sujet détaillé

15 mars 2016

Présentation du projet

Le cours offre une opportunité d'approfondir un langage de programmation à travers la réalisation d'un projet informatique. A travers ce projet vous serez amenés à effectuer une analyse des besoins ainsi que des moyens (algorithmiques, mathématiques) dont vous disposez suivi par l'identification des solutions et démarches à effectuer pour répondre à un client fictif.
A la fin de ce projet il vous serra demandé de fournir un livrable composé de

1. une implémentation de l'algorithme - respectant les *spécifications du client* (sujet détaillé) ;
2. les commentaires détaillés du code ;
3. le documentation du produit - *rapport du projet* (commentaires du code, un rapport avec les justifications des choix algorithmiques et des structures des données, etc.).

Une soutenance du projet est également prévu qui consiste dans une présentation orale suivi d'une démonstration de l'implementation.

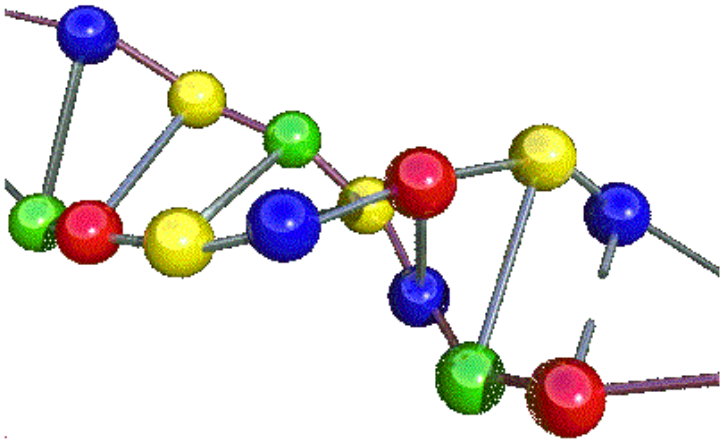
Table des matières

1 Introduction

L'optimisation est l'une des branches les plus importantes des mathématiques appliquées modernes, et fait l'objet de nombreuses recherches, à la fois pratiques et théoriques. Si on met de côté les problèmes d'optimisation discrète ou multicritère, alors la théorie de l'optimisation peut être séparée en deux grandes branches :

1. l'optimisation locale ;
2. l'optimisation globale.

Si on peut considérer que la première est presque "entièrement connue", la seconde est encore partiellement méconnue. La tâche principale de l'optimisation globale est la recherche de la solution qui minimisera un critère de coût donné, appelée "optimum global". L'optimisation globale vise donc à rechercher non seulement un minimum local, mais surtout le plus petit de ces minima locaux.



Il existe deux grandes approches à l'optimisation globale. L'une est dite déterministe : les algorithmes de recherche utilisent toujours le même cheminement pour arriver à la solution, et on peut donc "déterminer" à l'avance les étapes de la recherche. L'autre est aléatoire : pour des conditions initiales données, l'algorithme ne suivra pas le même cheminement pour aller vers

la solution trouvée, et peut même proposer différentes solutions. C'est vers cette seconde branche, la recherche globale aléatoire, que vont s'orienter nos travaux, et plus particulièrement vers un type bien précis d'algorithme de recherche aléatoire, les algorithmes génétiques.
La sélection naturelle est un concept ayant prouvé son efficacité dans de nombreux domaines et bien au delà de la théorie de l'évolution en biologie. Les concepts de l'évolution et de l'hérédité peuvent être utilisés dans le cadre de nombreux problèmes d'optimisation et ont permis l'émergence d'une approche connaissant de plus en plus de succès dans le domaine de l'optimisation "à multicritère", les algorithmes génétiques. Dans ce projet, nous nous proposons d'implanter une version générique des algorithmes génétiques et d'en dériver deux version permettant de résoudre des problèmes classiques d'optimisation.

1.1 Présentation du problème d'optimisation

Un problème typique d'optimisation s'exprime ainsi :
Maximise

$$f(x_1, x_2, \ldots, x_k)$$

En respectant l'ensemble des contraintes suivantes :

$$\begin{aligned}
&h_1(x_1, x_2, \ldots, x_k) \leq 0 \\
&h_2(x_1, x_2, \ldots, x_k) \leq 0 \\
&\vdots \\
&h_m(x_1, x_2, \ldots, x_k) \leq 0 \\
&h_{m+1}(x_1, x_2, \ldots, x_k) = 0 \\
&\vdots \\
&h_{m+n}(x_1, x_2, \ldots, x_k) = 0
\end{aligned}$$

L'expression $f(x_1, x_2, \ldots, x_k)$ est appelée la fonction objectif. Cette fonction peut avoir différentes formes, il peut aussi bien s'agir d'une fonction calculant un taux de rendement interne ou alors le nombre de connexions

entres les différents composants d'un circuit. L'ensemble des contraintes peut inclure des inégalités aussi bien que des égalités. De plus, les variables de ces égalités ou inégalités peuvent être des variables discrètes ou continues.
Pour résoudre ce type de problèmes, il est possible d'utiliser différentes techniques pouvant être classé grosso modo en trois catégories :

- **Les techniques de programmation linéaire ou non-linéaire** Ces algorithmes sont basés sur des propriétés mathématiques des fonctions optimisées. Si ces algorithmes sont relativement efficaces, leur domaine d'utilisation est assez étroit.
- **Les heuristiques de recherche** Ces algorithmes sont désormais facilement exploitables étant donné l'explosion des performances des ordinateurs et de la mémoire disponible. Ces approches sont très efficaces pour explorer les optima locaux de la fonction objectif et peuvent être facilement applicable à des grandes classes de problèmes.
- **Les algorithmes génétiques** Ces algorithmes reprennent deux concepts de la théorie de l'évolution : la définition d'un processus de sélection ainsi qu'un recours à des pseudo-mutations aléatoires permettant d'obtenir un nouvel ensemble de solutions à partir de solutions précédemment envisagés. Par rapport aux heuristiques de recherche, cette approche permet d'une part des recherches sur des grands domaines de définition, incluant de facto l'étude de plusieurs optima locaux, d'autre part cette approche ne nécessite que des adaptations mineures à chacune des classes de problèmes.

2 Eléments théoriques

2.1 Fonctionnement d'un algorithme génétique

Avant de pouvoir utiliser un algorithme générique, il est nécessaire de pouvoir définir pour le problème donnée :

- Une méthode permettant de représenter la solution sous une forme manipulable par la machine, la plupart du temps, il s'agit d'un vecteur

de valeurs booléennes.

— Une fonction permettant de calculer la qualité d'une solution.

Initialisation Un ensemble de plusieurs solutions est engendré de manière aléatoire. Cet ensemble constitue ce qui est appelé la population initiale. La taille de la population initiale dépend de la nature du problème et surtout de l'existence de nombreux optima locaux. Typiquement, les populations peuvent contenir plusieurs milliers de solution possible. Habituellement, la population initiale est engendrée de manière aléatoire afin de couvrir complètement le domaine des solutions possibles.

Sélection A une époque donnée, un échantillon de la population existante est sélectionné afin d'engendre la nouvelle génération. La sélection s'effectue selon un processus de sélection des solutions ayant la meilleure adéquation. Plusieurs fonction de sélection sont possibles, la plus simple consisté à sélectionner les solutions ayant la meilleur adéquation, d'autres fonctions stochastiques de sélections sont moins sélectives et ont pour but de préserver des solutions ayant une "médiocre adéquation" afin d'éviter une convergence vers un optimum local.

Reproduction L'étape suivante consiste à construire un nouvel ensemble de solutions à partie de l'ensemble des solutions venant d'être sélectionné. Des nouvelles solutions peuvent être engendrées à partir de solutions courantes par deux techniques, soit par le croisement d'une paire de solutions, soit par la mutation de deux solutions.

Croisement Une paire de solutions parmi les solutions précédemment est sélectionnée. Chacune des solutions est en fait un vecteur de bits $({}^1b_1, \ldots, {}^1b_n)$ et $({}^2b_1, \ldots, {}^2b_n)$. Nous choisissons au hasard un point de coupure k tel que $k = 1$ et $k < n$. Un premier descendant des solutions $({}^1b_1, \ldots, {}^1b_n)$ et $({}^2b_1, \ldots, {}^2b_n)$ est construit en concaténant aux k premiers bits de la première solutions les $k-1$ derniers bits de la seconde solution. De même un deuxième descendant en concaténant aux k premiers bits de la première solutions les $k-1$ derniers bits de la seconde solution. Au final la paire de solutions s'écrit comme suit :

$({}^1b_1, \ldots, {}^1b_k, {}^2b_{k+1}, \ldots {}^2b_n)$ et $({}^2b_1, \ldots, {}^2b_k, {}^1b_{k+1}, \ldots {}^1b_n)$

Mutation La mutation standard dans le cadre d'un algorithme génétique consiste en la probabilité qu'un bit appartenant à une séquence de bits représentant une solution possible puisse changer d'état, c'est-à-dire passer de zéro à un ou passer de un à zéro. La méthode standard consiste à générer une variable aléatoire pour chacun des bits dans une séquence. Cette variable aléatoire va permettre de déterminer si le bit doit être modifier ou non.

Terminaison Le processus générationnel continue tant qu'une condition de terminaison n'est pas atteinte. Les conditions de terminaison sont plus diverses et parmi les conditions de terminaisons les plus courantes nous pouvons citera :

- Une solution vérifie un critère d'adéquation
- Le nombre maximal d'itérations est atteint
- Le temps d'exécution maximal est atteint
- La qualité des meilleures solutions a atteint un plateau et les itérations successives ne semblent plus produire de meilleurs résultats.

Bien entendu, ces conditions de terminaisons peuvent être combinées entre elles.

2.2 Description de l'algorithme en pseudo-code

Algorithm 1 Algorithmes Génétiques

Sélectionne la population initiale

repeat

Sélectionne une partie de la population en fonction de leur adéquation.

Sélectionne les paires des meilleures solutions pour construire de nouvelles solutions.

Produit une nouvelle génération en combinant les paires des meilleures solutions et en introduisant des mutations.

until la condition de terminaison est vérifiée

3 Sujet du projet - application des algorithmes génétiques à un problème concret

3.1 Un petit problème NP-complet

Un problème intéressant en théorie de la complexité mais aussi en cryptographie est le suivant : considérons un ensemble d'entiers relatifs, est-ce que la somme des éléments d'un sous-ensemble de cet ensemble a pour valeur 0. Par exemple, pour l'ensemble $E = \{7, 3, 2, -5, 8\}$, la réponse au problème est "oui" puisqu'il existe la somme des éléments du sous-ensemble $s = \{3, 2, -5\}$ est égale à 0. De plus, il est souhaitable de trouver le ou les sous-ensembles ayant le plus grand nombre d'éléments. Les algorithmes génétiques permettent de trouver de manière relativement simple les solutions à ce type de problèmes.

3.2 Objet du projet

Le projet a pour but d'implanter un algorithme génétique et de l'utiliser pour déterminer les plus grands sous-ensemble d'un ensemble d'entiers relatifs tel que la somme des éléments de ce sous-ensemble est 0.

4 Déroulement et évaluation

4.1 Jalons

Le projet de déroulera sur 5 séances de TP. La séance du 29 mars sera l'occasion de valider des objectifs intermédiaires - présenter l'implémentation d'une structure de données (avec les justifications des choix et la documentation) qui sera capable de répondre à toutes les demandes du projet.

4.2 Critères d'évaluation finale

Les fournitures attendues (livrabe) consistent dans :

1. une implémentation de l'algorithme sous la forme d'une binaire - respectant les *spécifications du client* (sujet détaillé) accompagné des détails concernant le lancement du programme ainsi que des méthodes spécifiques de compilation (par exemple si des librairies externes sont utilisées) ;
2. les commentaires détaillés et pertinents du code ;
3. le documentation du produit - *rapport du projet* (commentaires du code, justifications des choix algorithmiques, des structures des données, répartitions des taches, analyse des résultats, evaluaion des performances de la solution, etc.).

L'efficacité de l'algorithme et l'empreinte mémoire du programme sont des critères d'évaluation importants tout comme la complexité des méthodes mises en oeuvre. Concernant la présentation, un certain recul par rapport au problème traité et aux méthodes mise en marche pour le résoudre est attendu.

References

1. Banzhaf, Wolfgang ; Nordin, Peter ; Keller, Robert ; Francone, Frank (1998). Genetic Programming – An Introduction. San Francisco, CA : Morgan Kaufmann. ISBN 978-1558605107.
2. Holland, John (1992). Adaptation in Natural and Artificial Systems. Cambridge, MA : MIT Press. ISBN 978-0262581110.
3. Hingston, Philip ; Barone, Luigi ; Michalewicz, Zbigniew (2008). Design by Evolution : Advances in Evolutionary Design. Springer. ISBN 978-3540741091.