Projet de M2 d'Optimisation Stochastique

Mise au point d'un système complexe pour l'optimisation continue parallèle

1 Cadre du projet

l'informatique est étroitement dépendante des ordinateurs sur lesquels les algorithmes tournent.

Jusqu'à maintenant, l'algorithmique qui s'est développée était essentiellement séquentielle car l'architecture des ordinateurs était basée sur celle proposée par von Neumann juste après la deuxième guerre mondiale, c'est-à-dire un processeur, connectée par un bus à une mémoire globale contenant les instructions et les données.

Depuis quelques années, les ordinateurs sont en pleine mutation, et l'architecture qui semble se stabiliser est, pour les ordinateurs de bureau, un CPU multi-coeurs indépendants comportant un gros cache mémoire connecté à une mémoire globale, mais aussi une carte graphique avec un processeur massivement parallèle composé de multi-processeurs SIMD et une mémoire propre avec peu de mémoire cache, mais beaucoup de registres mémoire, pour l'implémentation d'un parallélisem spatio-temporel.

D'autre part, les super-ordinateurs sont maintenant composés de clusters de telles machines (1400 PC avec 3 cartes graphiques pour TSUBAME 2.0).

L'informatique doit donc évoluer, ce qui passe par repenser une nouvelle algorithmique adaptée à ces nouvelles architectures et parmi les paradigmes disponibles, les systèmes complexes semblent particulièrement bien adaptés à ces architectures.

En effet, on peut définir les systèmes complexes comme des entités indépendantes en interaction locale, structurées sur plusieurs niveaux émergents d'organisation, où le tout ne peut se comprendre sans les parties et les parties sans le tout.

Les nouveaux ordinateurs ont l'architecture qu'il faut pour implémenter de tels systèmes : chaque entité peut tourner de manière indépendante sur

aussi car sur une carte GPU, les coeurs (groupés en multi-processeurs) peuvent parfaitement communiquer entre eux à un bas niveau, et ensuite, au fur et à mesure que l'on change de niveau, on peut ré-implémenter la même chose entre les multi-processeurs, puis entre les cartes GPU, entre les processeurs CPU, et à un niveau supérieur, entre les différents ordinateurs.

C'est dans ce cadre que ce projet de M2 se situe.

L'optimisation trouve de nombreuses applications aussi bien dans l'industrie que dans la recherche, et permet d'accéder à l'intelligence artificielle, ainsi que le montre les résultats obtenus par la Programmation Génétique. Il est donc primordial de développer de nouveaux algorithmes d'optimisation capables d'exploiter ces nouvelles machines, et un moyen d'y parvenir consiste à concevoir un système complexe capable de faire de l'optimisation, c'est-à-dire, pour les problèmes continus, trouver le minimum d'une fonction multi-modale possiblement irrégulière.

2 Présentation du projet

L'objectif est de prendre une version simple d'un algorithme simple (*Differential Evolution*) et de lui trouver une implémentation parallèle inspirée des systèmes complexes (multi-échelles, dynamique, ago-antagoniste, massivement parallèle, et tout et tout).

L'idée de base derrière DE est de trouver, dans un espace de recherche potentiellement irrégulier, une vallée menant à un optimum local à une échelle supérieure aux irrégularités.

Pour ceci un algorithme DE est doté d'une population d'individus que l'on fera évoluer de la façon suivante : en partant d'une population initialisée de manière uniforme sur l'espace de recherche, on prend 2 individus parmi les meilleurs (par exemple sélectionnés avec un tournoi n-aire) pour former un vecteur, allant du moins bon des 2 vers le meilleur.

S'il y a une vallée présente sous les individus, les meilleurs individus l'échantillonneront et le nuage d'individus prendra sa forme. De plus, en sélectionnant deux individus parmi les meilleurs, il y a plus de chances que le segment entre ces individus aille dans la direction de la vallée avec la différence de valeur entre les deux individus en indiquant le sens de ce qui devient donc un vecteur. La norme de ce vecteur indiquera la distance entre les individus.

Dans un premier temps, on implémentera un algorithme générationnel, c'est à dire que si un îlot comporte p individus parents, alors on créera e enfants avant de passer à la génération suivante. Le passage de $p + e$ à la nouvelle génération p s'effectuera avec une sélection de type tournoi n-aire (7 est une bonne valeur).

3 Idée à implémenter

Pour qu'un système complexe fonctionne bien, il est bien qu'il comporte un système ago-antagoniste. Là, on se propose d'implémenter un certain nombre d'îlots à population dynamique devant se reconfigurer automatiquement d'après l'espace de recherche sous-jacent.

4 Nombre d'îlots et échange entre îlots

On commence par définir un nombre d'îlots qui sera au moins égal au nombre de cœurs du processeur faisant tourner l'algorithme (histoire de charger la machine parallèle au mieux).

Ensuite, pour créer une dynamique multi-échelles, les îlots s'échangeront périodiquement des informations, pour se reconfigurer en tenant compte les uns des autres.

Localement, nous dirons qu'à chaque "génération", chaque îlot calcule et mémorise:

- Pour chaque individu, le vecteur qui a permis de le créer.
- Pour chaque îlot : un certain nombre d'informations comme le centre de gravité de la population, la somme des vecteurs de la population, etc. suivant les informations nécessaires dans l'algorithme.
- La direction, le sens et la norme d'un vecteur moyen pour la population et ...
- en gros, tout ce dont il peut y avoir besoin :-)

Pour que le système reste dynamique pour un bon compromis exploration/exploitation, je propose la stratégie suivante : à chaque génération, un îlot envoie à un autre (pris au hasard) ses statistiques, pour information. À chaque génération, un îlot pourra donc recevoir les statistiques d'un autre. Dans ce cas, celles-ci sont étudiées et comparées avec les statistiques locales pour modifier le paramétrage de l'îlot local. On ne fait rien lors de l'*envoi* de statistiques, mais uniquement à la *réception* de statistiques, et là, la dynamique suivante est proposée :

1. Si l'on reçoit des statistiques d'un îlot plus petit que soi, alors, cela signifie que nous sommes un îlot qui est plus en phase d'exploration que l'autre îlot plus petit, qui est donc plus en phase d'exploitation. Nous aurons donc besoin de plus de puissance de calcul et l'autre de moins de puissance de calcul.

   En conséquence, on augmentera la taille de la population de l'îlot de a individus.

   L'augmentation peut se faire soit en ajoutant des individus initalisés aléatoirement (pour promouvoir l'exploration), ou alors en créant plus d'individus dans la boucle de création des enfants.

2. Si l'on reçoit des statistiques d'un îlot plus gros que soi, alors, c'est le moment d'être altruiste : je suis plus en phase d'exploitation que l'autre îlot, donc, je vais diminuer ma taille de population de d individus, libérant ainsi de la puissance de calcul pour les plus gros îlots.

3. Ces deux stratégies aboutiront à la disparition des petits îlots et au contraire, au grossisement des plus gros. Ce sera la phase agoniste. Pour la contrebalancer, je propose donc une phase antagoniste qui consiste à scinder en deux les îlots au dessus d'une certaine taille, qui peut être $2p$. Découper en deux des îlots de taille $2p$ reformera deux îlots de taille p, comme au lancement de l'algorithme.

Il reste à déterminer une métrique pour déterminer qu'un îlot est plus petit ou plus gros que soi. Vous pouvez proposer celle que vous voulez, tant qu'elle n'est pas trop coûteuse (car il peut potentiellement y avoir des milliers d'individus, si on utilise une carte massivement parallèle). Celle que je propose est la somme de la norme des vecteurs ayant abouti à la génération courante. En effet, si les vecteurs sont grands, alors, cela signifie que l'îlot occupe une plus grande surface que si les vecteurs sont petits.

Cette métrique n'est pas coûteuse à implémenter en temps de calcul : il suffit de faire la somme des vecteurs pour chacun des individus. Sa complexité est donc en $\mathrm{O}(n)$.

Ensuite, pour diviser un îlot en deux, on pourrait découper la population en deux aléatoirement, mais peut-être qu'une classification de style k-mean des individus sur leur coordonnées (avec k=2) serait plus efficace (attention au temps de calcul).

Un autre moyen consisterait à trouver le vecteur moyen de la population pour voir dans quel direction et quel sens se dirige l'îlot. Ensuite, tous les individus issus d'un vecteur allant dans un sens opposé peuvent être des bons candidats pour être regroupés dans un îlot...

5 Benchmarking

Le *benchmarking* est un très beau sujet, central en recherche (mais aussi dans le monde de l'industrie), car il concerne la validation d'un travail par expérimentation et campagne de mesures.

Vous pouvez prendre les fonctions parmi le jeu de BBOB2013 et parmi celles-ci, des fonctions multimodales (c'est pour ça qu'on fait plusieurs îlots : pour que chacun des îlots aille explorer un autre optimum local. `http://coco.gforge.inria.fr/doku.php?id=bbob-2013`

À noter qu'en relançant exactement le même algorithme plusieurs fois, il est normal de trouver des résultats différents... et c'est l'une des choses désagréables lorsqu'on utilise un algorithme stochastique (= à composante aléatoire).

Pour tous les benchmarking de ces algorithmes, on doit donc effectuer une *moyenne* sur un nombre de *runs* significatifs. Généralement, on fait une moyenne sur 30 runs, mais pour vous économiser du temps, je propose que pour ce projet, vous n'effectuiez que des moyennes sur 10 runs.

Attention : les moyennes, c'est bien, mais il faut aussi évaluer la reproductibilité des expériences. Pour ceci, il faut toujours associer à la moyenne la variance et l'écart-type sur l'ensemble des runs et possiblement aussi la valeur minimale et la valeur maximale obtenue, pour se faire une idée raisonnable de la qualité d'un algorithme.

Pour ce projet, on ne s'en tiendra qu'aux moyennes pour les courbes, mais sachez que pour un article, par exemple, il faudra au moins l'écart-type.

6 Travail à rendre

Vous rédigerez un rapport de 5 à 10 pages (idéalement sous LaTeX) décrivant vos choix, vos essais et bien sûr les courbes obtenues.

Je suis joignable par mail pour toute question sur ce sujet.

Pierre Collet