TER : COLORATION DE GRAPHES

Notes sur les logiciels et API utilisés

Tutrice :
Nadia BRAUNER

Etudiant :
Michaël GABAY

LABORATOIRE G-SCOP

Année 2010

Table des matières

1 Intoduction

Ce document est un bref résumé des mes impressions sur les logiciels et API utilisés au cours de mon TER.

2 Produits IBM - ILOG

ILOG (qui a été racheté par IBM) propose des solveurs très puissants pour la programmation par contrainte et la programmation linéaire en nombres entiers.
En tant qu'académique vous pouvez accéder gratuitement à tous leurs produits. La page web du programme d'IBM est https://www.ibm.com/developerworks/university/ et les logiciels sont accessibles à partir de https://www.ibm.com/developerworks/university/software/get_software.html.

Tous les logiciels d'IBM sont fournis avec des interfaces permettant de les utiliser directement dans des programmes en C++ ou en Java. Ils sont également disponibles et optimisés pour de très nombreuses architectures de machines.

2.1 CPLEX

CPLEX est un solveur permettant de résoudre des programmes linéaires en nombres entiers. Il est très puissants et relativement rapide à prendre en main dans son fonctionnement de base. Toutefois, il dispose de (très) nombreuses options.
Je pense que sa puissance réside également dans ses options mais je n'ai utilisé pour le TER que le "minimum vital" de ce point de vue. Une lecture approfondie du manuel pourra se révéler très interessante pour comprendre celles-ci ainsi que le fonctionnement interne du solveur.

Il est assez difficile d'estimer le temps nécessaire à la résolution d'un problème par CPLEX ; en particulier car le solveur peut passer une très grande partie du temps de calcul à prouver une solution trouvée très rapidement. CPLEX se comporte également parfois de manière obscure (une contrainte spécifié avant ou après une autre peut radicalement changer le temps de calcul).

J'ai utilisé les versions 10 et 12 du logiciel et j'ai remarqué des différences notables :
- J'ai constaté une nette amélioration générale des performances dans CPLEX 12.
- CPLEX 12 semble prouver beaucoup plus vite les solutions.

– La gestion des contraintes a également, à mon avis, radicalement changé entre ces deux versions : j'ai constaté ce changement en résolvant plusieurs problèmes avec deux modèles différents - l'un correspondant au deuxième dans lequel on a dupliqué une contrainte pour la spécifier explicitement. Lors de la résolution par CPLEX 10, les deux modèles étaient équivalents (même temps de calcul, même nombre de noeuds explorés, comptés, restants...) alors que dans la version 12 on observe une différence considérable entre les deux modèles.

CPLEX est également difficile a utiliser sans un langage de modélisation. À ce titre, OPL est un candidat parfait. Je reviendrai dessus un peu plus loin.

CPLEX est, à mon avis, un excellent solveur, toutefois je pense qu'il est nécessaire d'étudier ses options si l'on veut véritablement exploiter tout son potentiel.
Il peut être également intéressant de tester les performances de CPLEX contre CP ; il est possible que certains PLNE soient résolus beaucoup plus rapidement si on utilise la programmation par contraintes.

2.2 CP

CP est le solveur d'IBM - ILOG pour la programmation par contrainte. J'ai utilisé la version 2 et ai tout simplement été bluffé par ses performances (il y a un facteur > 10 par rapport à choco sur les cas testés et les temps de calcul ne sont même pas comparables avec CPLEX).

Comme CPLEX il est à utiliser avec un langage de modélisation, comme OPL. En revanche, je n'ai pas regardé ses options.

Je serais très intéressé de connaître les résultats de tests des performances approfondis de CP contre CPLEX. Les exemples que j'ai traité donnant très largement l'avantage à CP...

2.3 OPL

OPL est le langage de modélisation d'IBM - ILOG. Il est très intuitif est très simple à prendre en main. Il est fournit avec une IDE pour sa version windows (je ne l'ai pas testé).
L'exécutable d'OPL permet de résoudre directement les problèmes sans avoir à les exporter préalablement pour les soumettre au solveur ensuite (chose qu'il permet également).

D'après moi, OPL est tout simplement le langage "naturel" pour résoudre des problèmes avec CP ou CPLEX.

3 Logiciels libres

J'ai également utilisé plusieurs logiciels libres au cours de mon TER. Ils sont tous très intéressants dans leurs domaines.

3.1 Boost

Boost [1] est une librairie très riche dont j'ai utilisé la partie graphe.
C'est une librairie C++ implémentée entièrement à base de templates ce qui lui octroie une grande flexibilité. Toutefois cette flexibilité est coûteuse car la compilation des template est très longue ce qui ralentit le processus de développement. Pour palier à celà, Boost dispose de bindings Python permettant de s'affranchir de l'étape de compilation des programmes en programmant en Python plutôt qu'en C++.
Boost différencie concepts (graph_traits) et implémentation pour un graphe. Cette différence est assez subtile et rend Boost extrèmement flexible mais très difficile à manipuler. Pour une utilisation intensive, la lecture d'un livre de référence s'impose et la documentation ne suffira probablement pas.

Boost dispose de très nombreux algorithmes classiques (exploration de graphes, plus courts chemins, composantes connexes, flot max, ...). Toutefois il ne propose pas d'algorithmes pour les problèmes $NP-durs$, bien que quelques heuristiques sont données à titre d'exemples dans la documentation.

Boost est très puissant et complet. Cette librairie est optimisée et destinée a être utilisé dans des programmes en C++, ce qui permet d'obtenir du code très performant. Toutefois, son utilisation n'est pas toujours naturelle même pour des programmeurs expérimentés comme en témoignent les forums.

3.2 Choco

Choco [2] est un projet français de solveur pour la programmation par contraintes. C'est une librarie Java très simple à manipuler.

La force de Choco réside à la fois dans sa simplicité (très simple à utiliser, dans un langage de programmation facilement accessible) et ses performances qui sont à mon goût très satisfaisantes même si elles sont bien inférieures à celles de CP qui doit, vraisemblablement, disposer de toutes les optimisations réalisées dans Choco puisque ce dernier est distribué sous licence BSD, une des moins restrictives du monde logiciel libre.

Choco dispose également d'options avancées comme la possibilité de maximiser/minimiser une fonction objective ou de chercher une solution ou toutes les solutions à un problème.

Toutefois, ce qui fait la force de Choco fait aussi sa faiblesse : le choix de Java comme langage de programmation ne permet pas d'espérer pouvoir atteindre des performances maximales.

Malgré ce dernier défaut, Choco est un projet extrèmement intéressant et n'ayant pas à pâlir de ses performances. Des tests de performances se révèlerait intéressants pour lever le voile sur la différence

[1] http://www.boost.org/
[2] http://www.emn.fr/z-info/choco-solver/index.html

avec son concurrent `CP`. Même si l'issue ne fait pas de doûte, l'estimation du facteur peut se révéler des plus instructives.

3.3 Cliquer

`Cliquer`[3] est un projet de Patric R. J. Östergård's[4]. Il propose un algorithme exact pour trouver une clique max dans un graphe quelconque (fichier fournit au format `Dimacs`) ainsi qu'une API permettant d'utiliser ses programmes dans des programmes en `C/C++`. Le programme principal est très simple à utiliser dans la ligne de commande.

L'efficacité de son programme est très impressionnante (de l'ordre du dixième de secondes pour trouver une clique max dans un graphe à plusieurs centaines de sommets et arêtes) mais ceci reste à confirmer par d'autres tests puisqu'il semblerait que dans les fichiers que j'ai testé les sommets de la clique max sont en tête.

4 Conclusion

Tous ces programmes sont intéressants dans leurs domaines et présentent des points forts comme des points faibles. Je vous ai fait part de mes impressions tout en essayant d'être le plus objectif possible mais je n'ai eu qu'un aperçu de la plupart des ces programmes et tout ce qui concerne les performances reste à confirmer par de véritables benchmarks.

J'espère que ces quelques lignes vous auront été utiles :)

[3] http://users.tkk.fi/pat/cliquer.html
[4] http://users.tkk.fi/~pat/