EasyChair Preprint

№ 2529

On the importance of programming practices to simplify the implementation of OR algorithms

Daniel Porumbel

February 2, 2020

Sur la place en RO des techniques pour rendre la programmation (avec `cplex` ou `gurobi`) plus facile

Daniel Porumbel[1]

CEDRIC-CNAM, 292 rue Saint Martin, Paris France

daniel.porumbel@cnam.fr

1 Aspects techniques et code source

Je présente une réalisation logicielle d'environ 1000 lignes de code C++, conçue pour écrire plus facilement des algorithmes de plans coupants. Au début, j'ai pensé que je n'allais jamais présenter ce code dans une communication officielle. Mais un jour j'ai compris que de nombreux travaux peuvent bloquer plus facilement sur des questions d'implémentation que sur d'autres problématiques qui bénéficient de plus d'attention (comme la modélisation).

Il s'agit juste de deux fichiers `CuttingPlanesEngine.cpp` et `CuttingPlanesEngine.h` qui permettent de faire fonctionner le code à droite. La première moitie de ce code minimise :

$\texttt{obj}^\top \mathbf{x} : \texttt{c}^\top \mathbf{x} \geq 4.2, x_i \in [0, 100] \, \forall i \in [1..n]$

L'appel `runCutPlanes(...)` exécute un algorithme de plans coupants qui appelle de manière itérative la fonction `sepFunc` pour trouver une nouvelle coupe. Cette fonction reçoit un paramètre n qui indique le nombre de variables et un paramètre `double* x` qui représente la solution courante. La fonction doit remplir un tableau `double* a` et une variable d pour indiquer qu'il faut ajouter la coupe $\mathbf{a}^\top x \geq d$.
Grâce à la technique « cheshire cat », il n'est pas nécessaire d'inclure de fichier `ilocplex.h`, ce qui accélère la compilation. Cela m'a permis d'utiliser le même programme à droite avec `gurobi`. Un 1er test avec un problème de séparation très simple a donné 19.2s avec `gurobi8.11` et 23.7s avec `cplex12.6` pour résoudre $\min \sum x_i : 2(x_i + x_j + x_k) \geq 5 \forall i, j, k \in [1..n]$, $n = 3500$.

```
int n=5;                    //nb de variables
double obj[]={3, 3, 3, 6, 8};
double c[]   ={1, 1, 1, 1, 1};

//sepFunc: a function/oracle to solve the
//separation sub-problem for a given x
CuttingPlanesEngine cutPlanes(n,sepFunc);
cutPlanes.setVarBounds(0,100);
cutPlanes.setObjCoefsMinimize(obj);

cutPlanes.modelAddCut(c,4.2);
cutPlanes.solve();
//cutPlanes.turnAllVarsInteger();
cout<<"Obj val before cutting planes:"
    <<cutPlanes.getObjVal()<<endl;

int itersUsed;
double time;
cutPlanes.runCutPlanes(itersUsed,time);

double finalObj = cutPlanes.getObjVal();
cout<<"Final obj="<<finalObj<<" reached
    after "<<itersUsed<<" iterations.\n";

double*x = new double[n];
cutPlanes.getPrimals(x);
for(int i=0;i<n;i++)
        cout<<"x["<<i<<"]="<<x[i]<<"; ";
```

Le `Makefile` cherche tout seul le chemin d'installation `cplex`. Comme les deux fichiers `CuttingPlanesEngine.h/.cpp` ont 1000 lignes, ils offrent plusieurs fonctionnalités que je ne peux pas décrire ici. Le code complet est disponible à `cedric.cnam.fr/~porumbed/cp.zip`

2 Des aspects techniques vers des réflexions philosophiques

D'abord, on peut se poser la question : pour quoi se permettre de sortir du cadre strictement technique pour s'intéresser à des aspects philosophiques sans impact sur le concret ? Le réponse est que l'impact réel puisse être plus important qu'on ne le pense. Un problème réel qui trouve

la racine dans « la philosophie régnante de l'époque » a été bien décrit par un philosophe du 19ème siècle [1], bien que certaines choses se sont améliorées depuis son époque :

> *"... les fruits prennent le goût du sol sur lequel ils ont mûri. Ce qui est prôné hautement, publiquement, en tout lieu, est lu, et constitue en conséquence la nourriture intellectuelle de la génération qui se forme ; cette nourriture a l'influence la plus décidée sur la substance de cette dernière, et ensuite sur ses productions. Par suite, la philosophie régnante d'une époque détermine son esprit. Donc, si la philosophie du non-sens absolu domine, si des absurdités sans fondement et exposées en un langage d'aliénés passent pour de grandes idées, cet ensemencement produit une belle génération sans esprit, sans amour de la vérité, sans sincérité, sans goût, sans élan pour rien de noble, pour rien qui s'élève au-dessus des intérêts matériels, dont font partie aussi les intérêts politiques."*

L'importance de la « philosophie régnante d'aujourd'hui » en informatique peut être analysée sous le prisme du texte ci-dessus. Revenons à la question de départ : quelle place pour des contributions d'implémentation dans la RO ? Mon intuition me dit qu'on nous fait souvent comprendre qu'il ne faut pas trop s'attarder sur ces aspects, comme s'ils n'étaient pas vraiment dignes de notre attention. Une telle conception de vie peut générer une fracture dans le monde scientifique, c.a.d, un clivage entre :

— des sommités académiques qui n'ont jamais vu un algorithme en exécution et qui se bornent à essayer de tout comprendre en gros. Ces personnes – surtout les très très âgés – peuvent manquer de contact avec le pays réel, un peu comme un très haut fonctionnaire.

— des programmeurs qui exécutent des algorithmes tout les jours mais qui n'ont pas toujours – surtout les très jeunes – les outils conceptuels et les bonnes grilles d'analyse pour déconstruire certains pièges du système, ce qui les empêche d'avoir une vision très large et exacte du domaine.

J'ai déjà abordé des questions similaires à l'édition précédente de la Roadef dans un article « Sur un détail d'implémentation : quelques idées pour accélérer l'exécution et la programmation C++ pour la RO » [2]. Cet article comporte que deux pages (la limite Roadef), mais je garde toujours en tête l'idée de le compléter pour faire une argumentation d'une douzaine de pages (qui sera déposé à `http://cedric.cnam.fr/~porumbed/soft/`). Je ne crois pas qu'il y aura beaucoup de gens (des hautes sphères) à me lire car l'histoire commencera comme ça :

> *"La vitesse d'exécution de tout algorithme pratique dépend d'un facteur constant de complexité δ étroitement lié à la qualité de l'implémentation. Je présente des communications à la Roadef depuis dix ans et je viens seulement de réaliser que tous mes exposés se terminent invariablement avec une phrase comme « Et pour finir, je vais vous épargner les détails d'implémentation parce que je préfère me focaliser sur l'essentiel et sur les concepts de haut niveau ». Malgré ses mérites, cette phrase est trop simpliste. Elle est de nature à éloigner le lecteur d'attention moyenne de certaines vérités.*
>
> *Comme une porte dérobée dans un logiciel, une conséquence inavouée est la suivante : le cout en temps de calcul est très peu impacté par (le facteur constant de complexité associé à) la qualité de l'implémentation et on peut l'ignorer. Ce facteur constant δ est aussi invisible lorsqu'on calcule des complexités théoriques comme $O(n^2)$, $O(n^2 logn)$, etc. Mais imaginer un vendeur qui dit que le coût à payer serait de 1000€ multiplié par un facteur constant $\delta = 2$ ou $\delta = 6$ sans aucune importance ! Cela est peut-être vrai pour certaines personnes, mais je suis heureux que le destin de m'a épargné le vide d'une existence dans un luxe si démesuré."*

Références

[1] Arthur Schopenhauer. La philosophie universitaire. (version numérisée par G. Heff, p. 61). `www.schopenhauer.fr/oeuvres/philosophie-universitaire.html`

[2] Daniel Porumbel. Sur un « détail» d'implémentation : quelques idées pour accélérer l'exécution et la programmation C++ pour la RO. *Le 20ème congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision, Le Havre, 2019.* `cedric.cnam.fr/~porumbed/soft/` (une version de 12 pages y sera un jour déposée).