Résolution optimale du problème d'allouer un projet par étudiant

Loïs Vanhée

15 mars 2018

Résumé

1 Problème

Instance : une liste de couples (allocation, rang), où allocation = E $\times$ P, un couple étudiant/projet.

Problème : Choisir un ensemble d'allocations $S = 2^A$ qui minimise $(1 \rightarrow S_1, 2 \rightarrow S_2, \dots)$, selon l'ordre lexicographique inverse, où $x \rightarrow S_x = |\{a \in A | rang(a) = x\}|$, le nombre de rangs x dans S. Informellement, on cherche à minimiser le nombre de plus insatisfaits au rang k, puis étant donné ce nombre, on cherche à minimiser le nombre de plus insatisfaits au rang $k-1$, etc.

2 Formalisation

Pour résoudre ce problème, je m'en suis remis aux pouvoirs mystiques de CPLEX, qui m'avait beaucoup impressionné lors de mes essais avec le DMCP. Voici la formalisation sous forme d'un programme linéaire.

Variables : $a_{e,p} \in A_R$, l'ensemble allocations étudiants/projets avec un rang inférieur à R. Ce sont variables binaires en $\{0, 1\}$, le problème est NP-complet.

Objectif : minimiser $\sum_{e,p} |E|^{rank(a_{e,p})} \times a_{e,p}$. Le coefficient multiplicateur $|E|^{rank}(a_{e,p})$ fait qu'incrémenter un étudiant du rang n au rang $n+1$ est toujours pire que de faire passer tous les étudiants du rang 1 au rang n. La limite est celle donnée par la taille maximum des double (i.e. 2^{1024}). Ainsi, avec 64 étudiants, on ne peut considérer un R plus grand que 32, ce qui n'est pas bien grave parce que pour les 64 étudiants, la pire insatisfaction est de 4. Au besoin, le package contient un algo qui calcule ce rang maximum (paradoxalement, cet algo est beaucoup plus lent, 10s, au lieu des 0.5s requises pour calculer l'allocation optimale, sur l'exemple donné par Greg).

Contraintes : exactement un projet par étudiant :

$$\sum_{p \in P} a_{e,p} = 1$$

et au plus un étudiant par projet :

$$\sum_{e \in E} a_{e,p} \leq 1$$

3 Résultats empiriques

Réponse sous une seconde pour une soixantaine d'étudiants, pire rang 4.