l'algorithme du simplexe

Algorithme efficace pour résoudre des problèmes de programmation linéaire

(ou de trouver qu'il n'y a pas de solution)

Beaucoup utilisé et réussit sur les problèmes "réels"

Théoriquement trop lent pour des problèmes très bizarres

L'idée de base (géométrique)

- Choisir une solution réalisable de base (sommet du polytope)

- Si la fonction objective a une valeur supérieure à un sommet voisin, aller à ce sommet

- Répéter (tant que possible...)

Pourquoi ça marche?

Chaque arête parcourue augmente la fonction objective: donc pas de boucles

Le nombre de sommets est fini: donc terminaison garantie

Pas de maximaux locaux qui ne sont pas de maximaux globaux: donc l'algorithme ne peut se terminer qu'à une solution optimale (s'il y en a)

Comment est-ce que ça s'arrête?

Aucun sommet voisin avec une valeur supérieure de z

On a trouvé la valeur optimale (si elle existe)

C'est quoi, un sommet voisin?

- Le sommet actuel est déterminé par n équations
- Une arête du polytope qui passe par ce sommet est déterminée par $n-1$ parmi elles
- Un voisin est donnée par les $n-1$ et une nouvelle équation
- Un ensemble de $n-1$ des équations actuelles et une nouvelle peut échouer (aucune solution ou plusieurs solutions ou pas réalisable ou valeur pas améliorée)
- Sur une arête il n'y a que deux solutions de base réalisables

Un très petit exemple de parcours itératif calculé de façon naïve

Deux variables x_1 et x_2
Deux contraintes explicites: $3x_1 + 2x_2 \leq 30$
$-2x_1 + x_2 \leq 8$
les 2 contraintes implicites: $x_1,\ x_2 \geq 0$
$z = -3x_1 + 2x_2$ à maximiser.

La solution de base $x_1 = x_2 = 0$ est réalisable avec $z = 0$. Les solutions voisines sont obtenues en supprimant une des équations $x_1 = 0$ ou $x_2 = 0$ et en ajoutant une des équations $3x_1 + 2x_2 = 30$ ou $-2x_1 + x_2 = 8$

Donc quatre possibilités:

- $x_1 = 0,\ 3x_1 + 2x_2 = 30$: $x_2 = 15$ n'est pas réalisable.

- $x_1 = 0,\ -2x_1 + x_2 = 8$: $x_2 = 8$ réalisable avec $z = 16$.

- $x_2 = 0,\ 3x_1 + 2x = 30$: $x_1 = 10$ réalisable avec $z = -30$.

- $x_2 = 0,\ -2x_1 + x_2 = 8$: $x_1 = -4$ n'est pas réalisable.

On continue de la solution:
$x_1 = 0, \ -2x_1 + x_2 \ = \ 8, \quad z = 16$.

Supprimer une des équations:
$x_1 = 0, \ -2x_1 + x_2 \ = \ 8$
Ajouter une équation choisie parmi
$x_2 = 0, \ \ 3x_1 + 2x \ = \ 30$

En fait on ne trouve qu'une nouvelle solution:
$3x_1 + 2x_2 \ = \ 30, \ -2x_1 + x_2 \ = \ 8$:
$x_1 = 2, \ x_2 = 12$ réalisable avec $z = 18$, donc une amélioration.

Et ainsi de suite (mais pour ce petit exemple il n'y a plus de possibilités d'amélioration).

Impressionnant?

Non.

On a fait 3 itérations et pour chaque itération on calcule 4 voisins; donc 12 sommets calculés sur les 6 qui existent!

Mais pour un exemple de taille raisonnable c'est mieux:
10 variables et 10 contraintes explicites: 184756 sommets
Si l'algorithme prend 40 itérations (un peu pessimiste), on n'en calcule que $40 \times 10 \times 10 = 4000$.
(Et on va voir que le vrai algorithme fait encore moins de calcul)

Plus la taille du problème est élevée, plus cette méthode gagne par rapport à la méthode très naïve de calculer toutes les solutions de base.

Le “vrai” algorithme du simplexe de Dantzig

Calcul beaucoup plus vite d'un bon voisin

On considère d'abord le cas où $(0, 0, ..., 0)$ est une solution de base réalisable et l'algorithme part de là:

On va supprimer une seule des équations $x_j = 0$; donc cette variable x_j va devenir positive et les autres resteront nulles. Donc, il faut choisir une variable à coefficient positif en z.

En choisir une (peut-être mais pas forcément celle au coefficient le plus élevé?)

Chaque contrainte avec un coefficient $a_{i,j} > 0$ donne une borne supérieure sur la nouvelle valeur de x_i;

Choisir la plus petite de ces bornes; c'est la contrainte qu'il faut rajouter.

Manipulation algébrique remet le programme dans une forme où la nouvelle solution de base est $(0, 0, ..., 0)$ pour un choix différent de variables, ce qui permet de continuer de la même façon.

Deux Problèmes avec l'algorithme du simplexe: (1) Amélioration nulle

Comment? Si le nombre d'équations des contraintes vérifiées à une solution est supérieure à n.
Une itération jette une contrainte et en ajoute une autre; ça peut donner la même solution.
Et l'algorithme pourrait boucler (même point; ensembles différents d'équations).

Pourquoi ne pas jeter les contraintes inutiles?
Trop difficile de les trouver!

Un programme robuste doit être capable de traiter ce cas.

Et une règle simple suffit: à chaque choix (colonne ou ligne du pivot), choisir la première variable possible pour sortir ou entrer.

Trouver une solution de base pour commencer

- Modifier le problème de sorte que le nouveau problème *auxiliaire* aie une solution de base triviale et une solution optimale du nouveau problème soit forcément une solution réalisable (de base) de l'ancien;

- Résoudre ce nouveau problème par l'algorithme du simplexe!

- Enfin commencer a résoudre le vrai problème

- Pour construire le nouveau problème:
  - Ajouter une nouvelle variable, disons y (avec $y \geq 0$)
  - Remplacer toute contrainte $lin \leq const$ par $lin - y \leq const$ (et transformer dans la forme canonique)
  - Il existe une solution réalisable de base avec $n + 1$ équations vérifiées
  - Minimiser y (maximiser $-y$); si le résultat est nul, on a trouvé une solution réalisable de base de l'ancien problème; sinon il n'y en a pas.