Encore une méthode

- Construire un flot partiel où chaque sommet sauf la source transmet au maximum ce qu'il reçoit

- si possible d étendre ce flot, faites-le

- sinon faire reculer ce qui est arrivé à un sommet et ne peut pas être transmis

- basé sur l'idée de hauteur : le flot est toujours du haut vers le bas; pour modifier le flot, il faut incrémenter la hauteur
  (au minimum des hauteurs des voisins utiles $+1$)

- au début hauteur nulle sauf pour la source (hauteur = nombre de sommets); les hauteurs de la source et du puits ne changent pas.

- complexité $O(n^3)$ (si la recherche du prochain sommet à traiter est bien gérée).

Pourquoi cette méthode marche-t-elle?

- un petit exemple

- elle doit trouver un flot (les surplus sont forcément rejetés vers la source; en fait sans dépasser hauteur de $2\times$ nombre de sommets)

- ce flot est-il maximum?

- quand le flot commence à reculer vers la source (première fois une hauteur dépasse le nombre de sommets), s'il y avait un chemin faisable pour passer un surplus vers le puits, ce chemin aurait une différence de hauteur entre deux sommets consécutifs de au moins 2, ce qui est impossible.

Rapport aux méthodes de parcours des solutions de base

L'algorithme de F et F, est-il une méthode de parcours des solutions de base comme le simplexe?

...?

Non. Il peut trouver un flot maximum qui n'est pas une solution de base du PL.

(Considérer un réseau ne possédant que deux chemins source $-$ X de capacités 1 chacun et deux chemins X $-$ puits de capacité 2.)

L'algorithme de Dinic ou Edmonds-Karp
Un algorithme qui trouve le flot maximum en temps polynomial

A l'itération d, il trouve le flot maximum parmi ceux qui n'utilisent que les chemins de longueur $\leq d$

Pour l'itération d, il construit un nouveau graphe qui est une copie partielle de celui du réseau mais qui ne contient que les arcs susceptibles d'être inclus dans une chaîne augmentante de longueur d

Il utilise ce graphe pour chercher une chaîne augmentante de longueur d (et termine cette itération s'il n'y en a pas)

Après chaque augmentation, il découpe du graphe tout arc dont le flux vient d'être réduit à nul ou augmenté à sa capacité
(et aussi tout autre arc qui n'est plus utile pour construire des chaînes augmentantes).

Equivaut, *grosso modo* à Ford et Fulkerson, mais avec recherche en largeur pour une chaîne augmentante au lieu de recherche en profondeur.

Complexité $O(m^2n)$ (ou $O(mn^2)$ avec des idées de Dinic) pour un graphe de n sommets et m arêtes.

Flot maximum avec coût minimum
L'algorithme de Busacker et Gowen

Itérativement améliorer un flot (initialisé à 0)

A chaque itération construire un graphe R qui dépend du flot actuel f et le graphe du réseau G: si G a un arc (x, y) avec coût c,

- si (x, y) n'est pas saturé, R a un arc (x, y) valué à c
- si (x, y) a un flux non nul, R a un arc (y, x) valué à $-c$

On cherche un chemin de la source au puits avec une valeur minimum
s'il existe, on augmente le flot selon la chaîne augmentante correspondant
sinon l'algorithme termine.

Cette recherche est effectuée par un algorithme bien connu de recherche du plus court chemin entre deux sommets (seule complication ici, arcs de "distance" négative mais c'est pas grave)

La raison pour laquelle cet algorithme réussit est plus compliquée, en particulier, si on commencait d'un flot aléatoire, il ne serait pas garanti qu'on arrive á un flot optimal;
mais, en commençant au flot nul, oui.

Equivaut à Ford-Fulkerson avec choix de la chaîne augmentante de coût minimum.

La Méthode de Balas pour les PLNE en 0-1

- Considérer les cas où z (à minimiser) n'a pas de coefficients négatifs et les variables sont triées en ordre croissant de ces coefficients (pas de restriction; tout programme peut être exprimé de cette façon)

- Ici séparation consiste à fixer une variable soit à 0 soit à 1

- Ce qui permet à simplifier énormément les sous-problèmes

- Si un problème n'a pas de b_j négatif, on a déjà une solution réalisable (en fixant

toutes les variables restant à 0) et c'est la solution optimale de ce (sous-)problème

- Parcourir l'arbre en profondeur

- Plusieurs règles d'élagage permettent de savoir qu'un problème donné est infaisable ou ne peut donner une solution meilleure d'une solution déjà trouvée.

Un exemple

Minimiser $z = 3x_1 + 5x_2 + 6x_3 + 9x_4 + 10x_5 + 10x_6$ avec les contraintes

$$-2x_1 + 6x_2 - 3x_3 + 4x_4 + x_5 - 2x_6 \geq 2$$
$$-5x_1 - 3x_2 + x_3 + 3x_4 - 2x_5 + x_6 \geq -2$$
$$5x_1 - x_2 + 4x_3 - 2x_4 + 2x_5 - x_6 \geq 3$$