M2ISN, RO: Les graphes

Bernhard Beckermann
Labo Painlevé (ANO-EDP), Univ. Lille
bbecker@math.univ-lille1.fr
http://math.univ-lille1.fr/∼bbecker

Villeneuve d'Ascq, 8 novembre 2024

Récapitulatif définition graphes

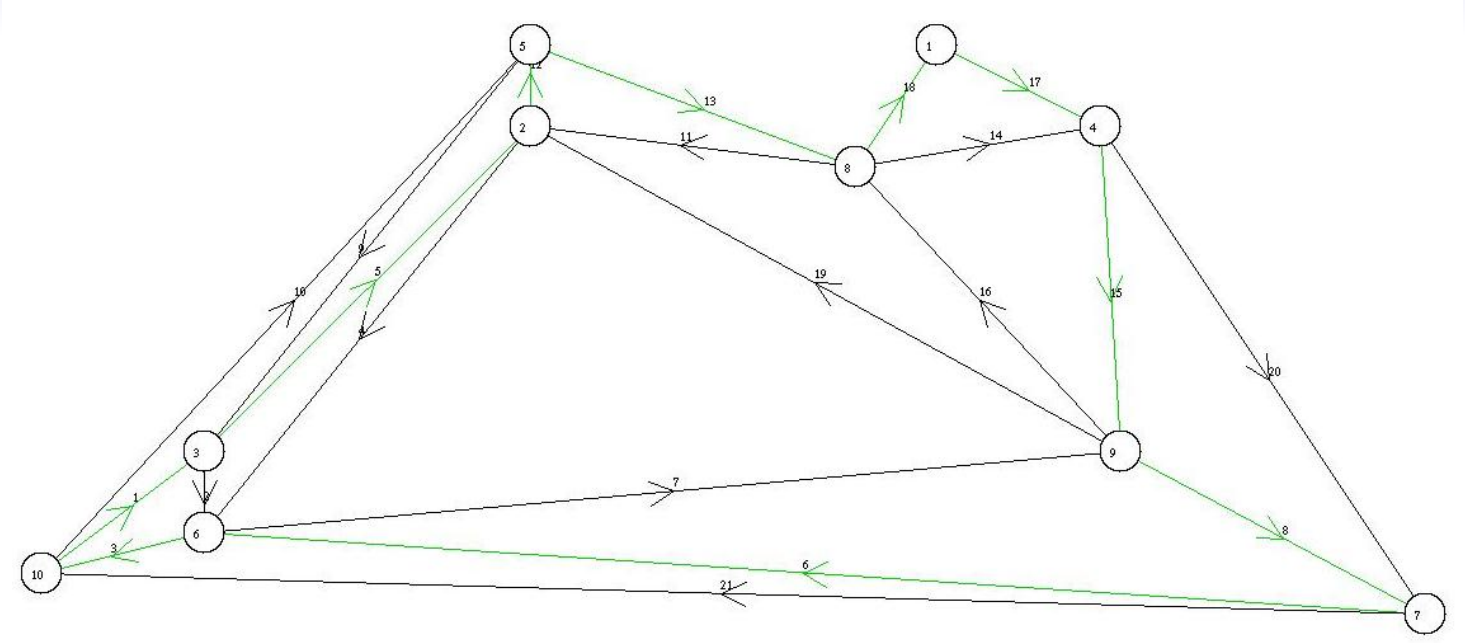
Un graphe est un couple $G = (S, A)$, $A \subset S \times S$, avec $m = |S|$ nb sommets, $n = |A|$ nb d'arêtes

Graphes non orientés: exemple simple exemple transpole	**Graphes orientés:** exemple simple, exemple transpole
arête $\{a, b\}$, extrémités a, b $\{a, b\} = \{b, a\}$ $a \neq b$ (?) voisins = sommets adjacents, $\Gamma(a)$ degré de a: nb voisins	arête orientée [arc] (a, b) origine [source] a, extrémité [but] b $a = b$ boucle successeurs $\Gamma^+(a) = \{b : (a, b) \in A\}$ prédécesseurs $\Gamma^-(b) = \{a : (a, b) \in A\}$
chaîne = suite de sommets adjacents longueur = nb d'arêtes chaîne fermée = cycle	chaîne orientée [chemin] = suite de sommets (entre crochets) ou d'arcs $\gamma, = [1, 4, 9, 7, 6, 10, 3, 2, 5, 8, 1]$ $= (17, 15, 8, 6, 3, 1, 5, 12, 13, 18)$ chemin fermé = circuit

Vocabulaire

Sous-graphe engendré par $S' \subset S$: $G' = (S', A')$,
$A' = \{(a, b) \in A : a, b \in S'\}$.
Sous-graphe partiel : $G' = (S', A')$, $S' \subset S$,
$A' \subset \{(a, b) \in A : a, b \in S'\}$.
chaîne/cycle/chemin/circuit simple : passant au plus une fois par un arc/arête
chaîne/cycle/chemin/circuit élémentaire : passant au plus une fois par un sommet (sauf si fermé)
chaîne/cycle eulerien : empruntant une et une seule fois chaque arête (graphe non orienté)
chaîne/cycle/chemin/circuit hamiltonien : passant une et une seule fois par chaque sommet (sauf si fermé)

graphe complet : comporte pour deux sommets distincts $a, b \in S$ soit l'arc $[a, b]$ soit l'arc $[b, a]$.
graphe connexe : partition en composantes connexes
$S = S_1 \cup S_2 \cup ... \cup S_\ell$ avec $a, b \in S_j$ si eu seulement s'il existe une chaîne avec extrêmités a et b (ou $a = b$) $\longrightarrow$ relation d'equivalence.
Un graphe est dit *connexe* si $\ell = 1$ (une seule composante connexe).
graphe fortement connexe : partition en composantes strictement connexes $S = S_1 \cup S_2 \cup ... \cup S_\ell$ avec $a, b \in S_j$ si eu seulement s'il existe une chaîne orientée avec source a et but b plus une avec source b et but a (ou $a = b$) $\longrightarrow$ relation d'equivalence. Un graphe est dit *strictement connexe* si $\ell = 1$.
distance entre deux sommets : longueur (en nombre d'arêtes) de la plus courte chaîne entre ces deux sommets. *diamètre* : la plus grande de ces distances.
Un *graphe étiqueté/valué/pondéré* est un graphe orienté $G = (S, A, d)$ ou à chaque arc j on associe une étiquette d_j (un nombre réel).
Le *poids/longueur d'un chemin/d'une chaîne* est la somme des poids des arêtes qui la composent (peut être généralisé).



Voici un exemple d'un graphe à 10 sommets (énumérés par $1, ..., 10$) et 21 arcs (énumérés par $1, ..., 21$). En vert on voit un circuit (partant du sommet 1, disons)
$\gamma = [1, 4, 9, 7, 6, 10, 3, 2, 5, 8, 1]$ (par sommets),
$\gamma = (17, 15, 8, 6, 3, 1, 5, 12, 13, 18)$ (par arcs). Ceci est un *circuit hamiltonien* (on rencontre une et une seule fois chaque sommet).

SCILAB et METANET

Pour visualiser les graphes et faire l'algorithmique sur les graphes, SCILAB comporte dans la version *3.0–4.1?* (sous windows et Linux, de préférence sous l'ancien mode graphique) le paquetage *METANET*:

- type de variable *graph-list* regroupant l'ensemble des données d'un graphe, entre outre:
  - source/but des arcs, noms pour sommets/arcs (chaînes de caractères)
  - 1 valuation pour sommets et 7 valuations pour arcs (coût, longueur, capacité,...)
  - coordonnées, diamètre, couleur sommets, forme, couleur arcs, etc
- édition (*edit_graph*) et affichage (*show_graph, plot_graph*) des graphes
- des *divers algorithmes* pour l'accessibilité, cheminement, flot

Il y a une aide sur ligne importante avec des exemples, taper *help graph-list*, *help autre-commande*

graph-list

```
// creating a directed graph with 13 nodes and 14
arcs.
ta=[1 1 2 7 8 9 10 10 10 10 11 12 13 13];
he=[2 10 7 8 9 7 7 11 13 13 12 13 9 10];
g=make_graph('essai',1,13,ta,he);
g('node_x')=[120 98 87 188 439 698 226 127 342 467
711 779 477];
g('node_y')=[ 21 184 308 426 435 428 129 360 435 55
109 320 321];
show_graph(g);
```

Ici le graphe est stocké dans la variable *g*, on accède aux différents champs de ce tableau en tapant *g('toto')* ou toto est un élément de la liste suivante: *name, directed, node_number, tail, head, node_name, node_type, node_x, node_y, node_color, node_diam, node_border, node_font_size, node_demand, edge_name, edge_color, edge_width, edge_hi_width, edge_font_size, edge_length, edge_cost, edge_min_cap, edge_max_cap, edge_q_weight, edge_q_orig, edge_weight, default_node_diam, default_node_border, default_edge_width, default_edge_hi_width, default_font_size, node_label, edge_label*

Quelques commandes/algorithmes METANET

Voici quelques commandes utiles avec nom parlant, pour la syntaxe voir la page aide correspondante:
add_edge - add_node - check_graph - delete_arcs - delete_nodes - edit_graph - edit_graph_menus - graph-list - graph_2_mat - graph_union - load_graph - make_graph - mat_2_graph - plot_graph - save_graph - show_arcs - show_graph - show_nodes - subgraph

Liste incomplète d'algorithmes:

accessibilité, plus court chemin : circuit - connex - find_path - graph_power - hamilton - is_connex - strong_con_nodes - strong_connex - trans_closure

flot : max_cap_path - max_flow - min_lcost_flow1 - min_lcost_flow2 - min_qcost_flow

divers : knapsack - qassign - salesman

Où est-ce qu'on rencontre des graphes? (1)

On rencontre les *graphes* (pondérés ou non, orientés ou non) dans des *contextes variés*:

Descriptif d'une séquence d'événements : Ici les sommets représentent les différents états d'un système, et les arcs représentent des possibilités de passage entre deux états (evt. avec un coût). On se pose la question comment passer d'un état initial à un état final.
Exemple: le batelier, le loup, la chèvre et le chou.

Réseau de transport (voiture, train, bus,...): Ici les arcs sont des connections (à sens unique si orienté) entre deux villes = sommets. A partir des connections on construit des parcours=chaînes/chemins. Poids naturel: distance (vol d'oiseau), péage, restriction pour les poids lourds, restriction de capacité pour le trafic.

Où est-ce qu'on rencontre des graphes? (2)

Réseau de conduite (eau, électricité,...): Ici on dispose de tuyaux entre stations de pompage. Poids naturel: capacité maximum, capacité minimum, coûts unitaires d'utilisation, pertes par tuyau,...

Autres: Réseaux de télécommunications, circuits imprimés (graphes planaires), organisation d'une file d'attente, problèmes d'ordonnancement, stockage des matrices creux (comportant peu d'éléments $\neq 0$) etc etc

Quelques problèmes classiques sur les graphes

Accessibilité : On se pose la question si on peut construire à partir d'un sommet donné des chemins allant vers un autre sommet ou vers tout autre sommet.

Tarjan, algorithmes de marquage, complexité $\mathcal{O}(n)$*. Warshal, complexité* $\mathcal{O}(m^3)$*.*

Problèmes de plus court chemin : Le poids d'un chemin est la somme des poids des arcs composant ce chemin. Parmi les multiples chemins répondant au problème d'accessibilité, trouver le plus court (c.-à-d., celui de poids minimum). Exemple longueur, nombre d'arcs.

> Ce problème est bien posé ssi il n'y a pas de circuits de poids < 0, et dans ce cas les chemins les plus intéressants sont élémentaires (on rencontre au plus une fois chaque sommet).

L'ensemble des plus courts chemins/chaînes allant d'un sommet à tout autre sommet forme une arborescence.
Bellman $\mathcal{O}(n)$*. Dijkstra* $\mathcal{O}(m^2)$*,* $\mathcal{O}(n\log(n))$*, Roy-Warshal* $\mathcal{O}(m^3)$*.*
Problème du voyageur de commerce: recherche d'un plus court circuit hamiltonien. *Complexité non polynômiale*.

Problèmes de coloriage : comment colorier une carte (les sommets d'un graphe planaire) sachant que deux pays adjacents devraient avoir une couleur différente?

Problèmes de flot : Ici on dispose d'un réseau de conduite avec une loi de conservation sur chaque sommet (sauf la source et le puits): la quantité entrante coïncide avec la quantité sortante.

- Flot maximum: transporter un maximum dépuis la source vers le puits: *Ford-Fulkersson, fini si données entières* Exemple : Dans le réseau de métro de Paris, envoyer un maximum de personnes depuis la Place d'Italie vers la Gare du Nord sachant que chaque rame peut transporter au maximum 100 personnes.
- Flot compatible: trouver un flot respectant les contraintes max/min de capacité: *Hoffman*. Exemple : ajuster le flot dans un graphe à 10 sommets.
- Flot de coût minimum: chaque unité transportée dans un tuyeau engendre un coût unitaire, trouver la façon la plus économique : *SIMPLEX (en termes des graphes)*. Exemple : Dans le réseau de métro de Paris, envoyer 120 personnes depuis la Place d'Italie vers la Gare du Nord dans un temps minimum.
- Multiflots: plusieurs demandes (conflictuelles) de transport

Arbres de poids minimum : Etant donné un graphe $G = (S, U)$ non orienté, chercher à extraire un sous-graphe $G' = (S, U')$ de poids minimum qui soit connexe et sans cycles (=arbre).

Exemple : On souhaite construire un réseau de canaux permettant de relier 10 villes entre elles. Dans le graphe non orienté on indique les tracés potentiels des canaux. Sachant que le coût de réalisation d'un canal entre deux villes est proportionnel à leur distance (vol d'oiseau), quel ensemble de canaux faut-il réaliser?
Solution par l'algorithme de Prim (ressemble à Dijkstra).

Paquetage ”graphs”

Le paquetage ”graphs” a été developpé par moi, il peut être téléchargé et propose une interface conviviale pour manipuler les graphes.

En cliquant sur le shortcut `scilab` dans le répertoire `graphs`, on lance Scilab. On se trouve dans le menu principal du paquetage graphes. En total on dispose de 3 fenêtres: du menue nommé ”graphes”, du fenêtre comportant le dessin du graphe en cours, et de la fenêtre principale de Scilab, pratiquement pas utilisé.

Voici quelques explications concernant le menue.

Menue principal: Manipulation du graphe

- `Charger graphe` permet de choisir un graphe dans une collection existante de graphes.
- `Editer graphe` permet d'éditer le graphe courant et de le sauvegarder. On peut donner la valeur 0 (ou une valeur aléatoire) à une variable associée a chaque sommet (couleur/demande) ou arc (couleur:longueur/coût/capacité min/capacité max). Notons en particulier la possibilité d'éditer chaque sommet/arc d'une manière graphique en le sélectionnant avec le bouton gauche de la souris (on sort de ce mode en cliquant sur le bouton à droite).
- `Paramètres affichage` permet de changer les paramètres d'affichage pour les sommets et/ou arcs.

Menue principal: Algorithmes Scilab

`Algorithmes Scilab` comporte certains procédures déjà incluses dans scilab.

- `Circuit/rang` permet de détecter des circuits (voir le graphe `mesh10.graph`) ou, dans le cas contraire, déterminer le rang de tous les sommets (voir le graphe `maison.graph`) (rang = longueur du plus long chemin en nombre d'arcs à partir du sommet no 1).
- on peut détecter des composantes (strictement) connexes, des circuits hamiltoniens (dans le cas écheant comme dans le graphe `mesh10.graph`)
- on peut colorier des successeurs etc d'un sommet donné.
- finalement on propose des "boîtes noires" pour des questions de cheminement/flot et le voyageur de commerce (très gourmand en espace mémoire).

Menue principal: Mes Algorithmes

`Mes Algorithmes` comporte un certain nombre d'algorithmes en mode "pédagogique" qui on été ajoutés à SCILAB.

- `accessibilité chemin par marquage`: variante de Tarjan (FIFO, FILO);
- Recherche du plus court chemin: `Dijkstra`, `Bellman`, `Roy-Warshal`;
- Problèmes de flot : Ford-Fulkerson et Hoffman
- Parcours euleriens : (éventuellement d'abord compléter le graphe;
- `montrer lignes d'un réseau metro/bus`pour expliquer des graphes comme `paris.graph` (metro de Paris orienté) `paris2.graph` (metro de Paris non orienté), `transpole.graph` (metro/bus de la metropôle lilloise orienté), `transpole2.graph` (metro/bus de la metropôle lilloise non orienté).

Algorithme de marquage (Tarjan)

Tâche: dans un graphe orienté $G = (S, A)$, déterminer l'ensemble de sommets $x \in S$ pour lesquels on peut construire un chemin avec source $s \in S$ et but x (en bref: x est accessible depuis s).

Invariant: On marque le sommet x

- au crayon si on a su construire un chemin avec source s et but x.
- à l'encre si de plus on a déjà examiné tous les successeurs du sommet x.

Initialisation: Le sommet s est marqué au crayon.

Itération: Tant qu'il existe encore un sommet x marqué au crayon faire
marquer ce sommet à l'encre et
tous ses successeurs non marqués au crayon

Résultat: Tout sommet marqué à l'encre est accessible, tout autre sommet n'est pas accessible.

Problèmes sur ordinateur: il faudra gérer une file d'attente des sommets marquées au crayon (retirer au début, ajouter à la fin (= FIFO) ou au début (= FILO)). Aussi, il faudra savoir faire appel aux successeurs.

Preuve de l'algorithme

Il y a trois choses à démontrer

- à la fin il n'y a plus de sommets marqués au crayon: sinon, on aurait continué l'algorithme
- un sommet x marqué à l'encre est accessible depuis s: preuve par récurrence sur l'ordre de marquage à l'encre. Le sommet $x = s$ est clairement accessible. Tout sommet $x \neq s$ marqué à l'encre à été marqué d'abord au crayon, parce qu'il y avait un sommet y prédécesseur de x qui était marqué à l'encre. Par hypothèse de récurrence, y est accessible depuis s, et donc aussi x.
- à la fin, un sommet x non marqué n'est pas accessible depuis s: par absurde, supposons qu'il existe un chemin $\gamma = [y_1, y_2, ..., y_k]$ avec source $s = y_1$ et but $x = y_k$. Le sommet s étant marqué, il existe un indice $j \in \{1, 2, ..., k-1\}$ de sorte que y_j est marqué, mais pas y_{j+1}. D'après la première partie on sait que y_j est marqué à l'encre, et y_{j+1} est successeur de y_j d'après la définition d'un chemin. Par conséquent, on aurait oublié de marquer y_{j+1} au crayon au moment où on a marqué y_j à l'encre, ce qui contredit les règles de l'algorithme.

Variantes de Tarjan

- *Construire explicitement les chemins* par une étiquette père : père(y) $= x$ si le sommet x a permis de marquer au crayon le sommet y.
- *Trouver l'ensemble des sommets x de sorte que le sommet s est accessible depuis x* : **Modification:** colorier au crayon les prédécesseurs au lieu des successeurs.
- *Trouver la composante connexe à laquelle appartient s* (aussi pour graphes non orientés): construire des chaînes au lieu des chemins. **Modification:** colorier au crayon les voisins.

Recherche des circuits par marquage/étiquetage

Lemme 1 : S'il existe un sous-ensemble $S' \subset S$ non vide de sorte que tout sommet dans S' admet un prédécesseur dans S' alors le sous-graphe engendré par S' contient un circuit (=cycle orienté).

Preuve constructive: on donne un algorithme de construction d'un circuit.

affecter à tout $x \in S'$ l'étiquette père$(x) = \emptyset$ et choisir un $y \in S'$

tant que père$(y) = \emptyset$ faire

choisir comme nouveau x le sommet y

choisir comme nouveau $y \in S'$ un prédécesseur de x

affecter l'étiquette père$(x) = y$

Preuve de l'algorithme

- L'algorithme s'arrête car chaque fois on examine un nouveau sommet y et l'ensemble des sommets est fini.
- à la fin on a construit un circuit car on dispose d'une suite de sommets $x_0, x_1, ..., x_k$ avec père$(x_{j-1}) = x_j$, c.-à-d., x_j est prédécesseur de x_{j-1}, et père$(x_k) = x_0$. Donc $[x_k, x_{k-1}, ..., x_0, x_k]$ est un circuit.

Reformulation du Lemme 1: Soit $G = (S, A)$ un graphe, et soit $S' \neq \emptyset$ l'ensemble des sommets non marqués à l'encre dans un instant de l'algorithme de marquage. Alors au moins une des propriétés suivantes est vraie:

- *il existe un sommet non marqué à l'encre avec tous ses prédécesseurs déjà marqués à l'encre*.
- On peut construire dans le sous-graphe engendré par S' un circuit par l'algorithme de la preuve. Exemple.

L'ordre de marquage à l'encre

> $(*)$ *On marque seulement le sommet x à partir du moment où on a marqué tous ses prédécesseurs.*

peut être réalisé en affectant à chaque sommet x une étiquette compteur(x) comptant le nombre de prédécesseurs pas encore marqués. L'ensemble de candidats pour marquage est donc l'ensemble de sommets portant l'étiquette 0.

---

affecter à tout $x \in S$ l'étiquette compteur$(x) = |\Gamma^-(x)|$

tant qu'il existe x non marqué avec compteur$(x) = 0$ faire

  marquer x à l'encre

  pour tout $y \in \Gamma^+(x)$: diminuer l'étiquette compteur(y) par 1

Remarque : L'ordre $(*)$ de marquage joue un rôle dans des divers applications, en particulier dans le problème d'ordonnancement: dans quel ordre faut-il exécuter des différentes tâches si on veut construire une maison?
Montrer graphe. Faire marquage..

Les plus courts/longs chemins

Position du problème: On se donne un graphe orienté $G = (S, A, d)$ à poids, c'est-à-dire, chaque arc $a \in A$ admet un poids $d(a)$ (réel pas forcement positif). Sachant que le poids d'un chemin est la somme des poids des arcs qui composent ce chemin, on se pose le problème de déterminer pour tout $x \in S$ la quantité $\pi(x)$ étant le poids d'un chemin à poids minimum avec source s et but x pour un sommet fixé s. On cherche parfois également à construire un tel chemin.

Convention: $\pi(x) = +\infty$ si $x \in S$ n'est pas accessible depuis s. Rappelons qu'un algorithme devrait également détecter s'il existe un circuit de poids < 0 (dit *circuit absorbant*): dans ce cas, le problème ci-dessus est mal posé pour certains sommets car il vaut mieux d'emprunter le circuit un nombre illimité de fois.

Principes d'optimalité

(si le graphe ne comporte pas de circuits absorbants)

1. Si $[x_0, x_1, x_2, ..., x_\ell]$ est un chemin de poids minimum entre x_0 et x_k alors tout sous-chemin $[x_i, x_{i+1}, ..., x_j]$ pour $0 \leqslant i < j \leqslant \ell$ est un chemin de poids minimum entre x_i et x_j.
   Preuve simple: on ne peut pas avoir un chemin plus court entre x_i et x_j car sinon on obtient aussi un chemin plus court entre x_0 et x_ℓ.
2. $\pi(s) = 0$: il vaut mieux de ne pas se déplacer.
3. $y \neq s : \pi(y) = \min\{\pi(x) + d((x, y)) : x \in \Gamma^-(y)\}$: tout chemin allant de s à y admet comme avant-dernier sommet un prédécesseur de y.

Algorithme de Bellmann dans un graphe sans circuits

Deux idées:

• importance de l'ordre de calcul: si on connaît $\pi(x)$ pour tout prédécesseur x de y, alors aussi $\pi(y)$

• il existe un marquage dans l'ordre $(*)$: on marque y seulement si on a marqué tous ses prédécesseurs. Ce marquage permet de marquer tout sommet à condition que

- tout sommet est accessible depuis s
- il n'y a pas de circuits.

Trouver une énumération $x_1, x_2, ..., x_m$ de S respectant $(*)$, $s = x_1$

initialiser $\pi(s) = 0$ et pour $x \neq s$: $\pi(x) = +\infty$

pour $j = 1, 2, ..., m$ faire

  pour tout y successeur de x_j faire

    $\pi(y) = \min\{\pi(y), \pi(x_j) + d((x_j, y))$

Une version plus efficace combine l'idée précédente avec l'algorithme de marquage:

Hypothèse : $G = (S, A)$ est un graphe pondéré sans cycles, $s \in S$

Tâche: Pour $x \in S$, calculer le poids $\pi(x)$ d'un chemin à poids minimum avec source s et but x.

Initialisation:

Pour tout $x \in S$ faire

affecter étiquettes père$(x) = \emptyset$, $\pi(x) = +\infty$, compteur$(x) = |\Gamma^-(x)|$

affecter étiquette $\pi(s) = 0$, marquer s au crayon.

Itération:

tant qu'il existe x marqué au crayon

marquer x à l'encre

pour tout y successeur de x faire

si la somme entre $\pi(x)$ et $d((x, y))$ est $< \pi(y)$ alors

poser $\pi(y) = \pi(x) + d((x, y))$, père$(y) = x$

diminuer l'étiquette compteur(y) par 1

si compteur(y) s'annule alors marquer y au crayon

Invariants: $\pi(x)$ est le poids d'un chemin de s à x à poids minimum dans le sous-graphe engendré par l'ensemble des sommets marqués à l'encre plus x.

Résultat: $\pi(x)$ pour tout sommet x accessible depuis s (= à l'encre).
Exemple.

Application aux problèmes d'ordonnancement

Un projet (construction d'une maison) consiste à effectuer plusieurs tâches. Connaissant pour chaque tâche t sa durée d_t et les tâches qui doivent la précéder, déterminer la date la plus tôt $\psi(t)$ pour l'exécution de la tâche t, et la *durée minimum du projet*. Indiquer également les tâches dites *critiques* pour lesquelles un retard impliquera un retard pour l'achèvement du projet.

Nom de la tâche	durée (en semaines)	tâches antérieures
Travaux de maçonnerie	7	aucune *(début)*
Charpente de la toiture	3	maçonnerie
Toiture	1	charpente
Installations sanitaires	8	maçonnerie
Façade	2	toiture, installations
Fenêtres	1	toiture, installations
Aménagement du jardin	1	toiture, installations
Travaux de plafonnage	3	fenêtres
Mise en peinture	2	plafonnage
Emménagement	1	façade, jardin, peinture
début	*0*	*aucune*
fin	*0*	tâches sans successeurs, ici *emménag*

Début du projet: $\psi(\text{début}) = 0$.
Fin du projet: $\psi(\text{fin})$.
Contraintes d'antériorité :
pour toute tâche $t \neq$début nous avons
$\psi(t) = \max\{\psi(s) + d_s : \text{tâche } s \text{ est antérieure à } t\}$.
Considérons le graphe $G = (S, A)$ avec comme sommets les tâches, $(s, t) \in A$ de poids d_s si et seulement la tâche s est antérieure à la tâche t. Alors

pour tout $t \in S$, la quantité $\psi(t)$ coincide avec le poids d'un chemin de poids maximum du sommet début à t.
Les tâches critiques sont ceux composant un chemin de poids maximum entre le sommet début et le sommet fin.

NB: notre graphe ne comporte pas de circuits et tout sommet est accessible depuis le sommet début.

Montrer graphe. Faire calcul.

Exercice 10

M. Urgent veut vendre sa voiture le plus rapidement possible, mais la voiture est vétuste et ne passera pas le contrôle technique avant d'avoir effectué des réparations. Pour effectuer chacune de ces tâches, M. Urgent peut faire appel à des garages *A* ou *B*, sachant qu'il y a 20 minutes de route pour joindre le garage A, 30 minutes de route pour joindre le garage *B* et 40 minutes de route entre les deux garages. De plus, les garages ont besoin des délais différents (voir tableau) pour effectuer une des trois tâches. Finalement, M. Urgent se trouve encore à sa maison, et souhaite y retourner avec les bénéfices de vente.

Durée en minutes	réparations	contrôle technique	vente de la voiture
Garage A	120	100	70
Garage B	140	40	80

Donner une modélisation mathématique comme un problème de recherche d'un chemin de poids minimum dans un graphe pondéré à préciser. À qui faut-il confier au mieux les différentes tâches ?

Solution.

Algorithme de Dijkstra dans un graphe à poids positifs

On se donne un graphe pondéré, à poids positifs. Par conséquent, tout circuit est de poids $\geqslant 0$, et le problème des chemins de poids minimum est bien posé.

L'algorithme de *Dijkstra* est un *algorithme de marquage*. Sa particularité est la sélection d'un sommet dans la file d'attente: on prend *le sommet le plus prometteur*.

Hypothèse : $G = (S, A)$ est un graphe pondéré, de poids $\geqslant 0$, $s \in S$

Tâche: Pour $x \in S$, calculer le poids $\pi(x)$ d'un chemin à poids minimum avec source s et but x.

Initialisation: écrire étiquette $\pi(s) = 0$ au crayon.

Itération:

tant qu'il existe encore une étiquette écrite au crayon

parmi les sommets portant une étiquette écrite au crayon, trouver

le sommet x avec étiquette $\pi(x)$ de valeur minimum

écrire $\pi(x)$ à l'encre

pour tout y successeur de x faire

former Σ, la somme entre $\pi(x)$ et le poids de l'arc (x, y)

si y n'a pas encore une étiquette ou si Σ est plus petit que $\pi(y)$ alors

marquer au crayon $\pi(y) = \Sigma$, père$(y) = x$

Invariants:

- *Toute étiquette $\pi(y)$ représente le poids d'un chemin de s à y.*
- *Une étiquette à l'encre $\pi(x)$ représente le poids d'un chemin de s à x de poids minimum.*

Exemple d'un graphe à 10 sommets, Exemple d'un graphe à 100 sommets.

Preuve de l'algorithme (1)

- Toute étiquette $\pi(y)$ représente le poids d'un chemin de s à y.
  Preuve par récurrence sur l'ordre de marquage: c'est vrai au début pour $y = s$ (chemin trivial de s à s). Si $y \neq s$, toute valeur de $\pi(y)$ est égale à une somme de $\pi(x)$ plus le poids de l'arc (x, y). Par hypothèse de récurrence, $\pi(x)$ représente le poids d'un chemin γ de s à x. En ajoutant l'arc (x, y) à γ, on obtient bien un chemin de s à y ayant le poids $\pi(y)$.
- A la fin de l'algorithme, tout sommet accessible depuis s porte une étiquette à l'encre, et tout autre sommet ne porte pas d'étiquette : il s'agit d'un algorithme de marquage.

Preuve de l'algorithme (2)

- Une étiquette à l'encre $\pi(x)$ représente le poids d'un chemin de s à x de poids minimum.

  Notons par $x_1, ..., x_\ell \in S$ dans l'ordre les sommets marqués à l'encre par l'algorithme de Dijkstra. Nous montrons la propriété par récurrence sur $k = 1, ..., \ell$.

  Au début, s est le seul sommet avec étiquette écrite au crayon, et donc $x_1 = s$ et $\pi(s) = 0$. Comme tout chemin est de poids $\geqslant 0$, la propriété est bien vrai pour $k = 1$.

  Soit $k > 1$, et γ un chemin de poids minimum de s à x_k. Si le poids de γ est $\geqslant \pi(x_k)$, alors d'après la première partie on a égalité, et on a démontré notre propriété. Par absurde, supposons que le poids de γ soit strictement plus petit que $\pi(x_k)$. La source de γ est égale à s, et le chemin γ comporte alors un arc de la forme (x_j, y) avec $1 \leqslant j < k$ et $y \notin \{x_1, ..., x_{k-1}\}$. Comme le poids d'un arc est $\geqslant 0$, le poids de γ est minoré par le poids d'un chemin de poids minimum de s à x_j plus le poids de l'arc (x_j, y). Par hypothèse de récurrence, on peut alors minorer le poids de γ par $\pi(x_j)$ plus le poids de l'arc (x_j, y), ce qui est $\geqslant \pi(y)$ au moment qu'on marque x_k à l'encre. Par conséquent, $\pi(y) < \pi(x_k)$ ce qui contredit les règles de marquage à l'encre.

Remarques sur la complexité des algorithmes

Pour un graphe $G = (S, A)$: $m := |S|$, $n = |A| \leqslant m^2$. Généralement $n \geqslant m$.

Besoins mémoire : $\mathcal{O}(n)$

- $\mathcal{O}(1)$ tableaux de taille n: successeurs, poids,
- $\mathcal{O}(1)$ tableaux de taille m: sommets, marquage, compteur, père, π, pile, ...

Opérations élémentaires (sauf Dijkstra) : $\mathcal{O}(n)$

- $\mathcal{O}(1)$ opérations pour chaque sommet: marquage, gestion pile,
- $\mathcal{O}(1)$ opérations pour chaque arc: test marquage, mise à jour π, pere, compteur, ...

Pour Dijkstra il s'ajoute la tâche d'extraire à au maximum m reprises le plus petit élément d'une pile à au plus m éléments.

- $\mathcal{O}(m^2)$ pour la version élémentaire: déterminer chaque fois le plus petit élément de la pile.
- $\mathcal{O}(n \log(n))$ pour une version plus sophistiquée: garder une pile triée par ordre croissant des éléments. Il y a au plus $\mathcal{O}(n)$ mises à jour des éléments: retirer l'ancien élément et insérer le nouveau par une méthode de dichotomie en complexité $\mathcal{O}(\log(m))$.

La matrice d'un graphe et ses puissances

Définition : la matrice d'adjacence B d'un graphe orienté $G = (S, A)$ est une matrice énumérée en lignes/colonnes par les sommets: $B(j,k) = 1$ si $(j,k) \in A$, et $B(j,k) = 0$ sinon (si G n'est pas orienté alors A symétrique).
Lemme : La position (j,k) de la puissance B^ℓ donne le nombre de chemins distincts de longueur ℓ.
Preuve par récurrence sur ℓ : Des chemins de longueur $\ell = 1$ sont des arcs, voir la définition de B. Pour obtenir les différents chemins de longueur $\ell \geqslant 2$ avec source j et but k, on compte combien de chemins de longueur $\ell - 1$ existent avec source j et but i, $i \in S$, et lesquelles parmi eux peuvent être prolongés par un arc (i,k) pour obtenir un chemin de longueur ℓ de j à k

$$B^\ell(j,k) = \sum_{i \in \Gamma^-(k)} B^{\ell-1}(j,i) = \sum_{i \in S} B^{\ell-1}(j,i) B(i,k).$$

NB1: si l'opération $+$ est remplacée par l'opération $\oplus = \max$, alors $B^\ell(j,k) \in \{0,1\}$, et vaut 1 si et seulement s'il existe un chemin de longueur ℓ avec source j et but k.

NB2 (graphes probabilistes): si $B(j,k) \in [0,1]$ décrit la probabilité que l'espèce j se transforme en espèce k (*matrice de transition* d'une chaîne de Markov), alors $B^\ell(j,k)$ décrit la probabilité que l'espèce j se transforme en espèce k après ℓ mutations.

Cadre abstrait

$(D, \oplus, \otimes)$ une algèbre, $G = (S, A)$ un graphe pondéré sans boucles à poids $d(a) \in D$.
Un chemin $\gamma = (a_1, ..., a_\ell)$ admet un poids $d(\gamma) = d(a_1) \otimes d(a_2) \otimes ... \otimes d(a_\ell)$.
Problème: chercher si possible $\pi(j, k) = d(\gamma_1) \oplus d(\gamma_2) \oplus ...$ avec $\gamma_1, \gamma_2, ...$ décrivant tous les chemins de source j et de but k.

Exemples:

$D = \mathbb{Z}$, $\otimes = \times$, $\oplus = +$, $d(a) \in \{0, 1\}$: compter tous les chemins de j à k

$D = \{0, 1\}$, $\otimes = \times$, $\oplus = \max$, $d(a) \in \{0, 1\}$: tester l'existence d'un chemin de j à k

$D = \mathbb{R} \cup \{-\infty\}$, $\otimes = +$, $\oplus = \min$: recherche d'un chemin de poids minimum

$D = [0, +\infty]$, $\otimes = \min$, $\oplus = \max$: chemin permettant de passer un tonnage maximum

$D = [0, 1]$, $\otimes = \times$, $\oplus = \max$: recherche d'un chemin de fiabilité maximum

Définissons la matrice B par $B(j,k) = d((j,k))$ si $(j,k) \in A$ et $B(j,k) =$ élément neutre par rapport à $\oplus$ sinon.
Les puissances $B^{\otimes 1} := B$, $B^{\otimes k} := B \otimes B^{\otimes(k-1)}$ nous donnent

$$B^{\otimes \ell}(j,k) = d(\gamma_1) \oplus d(\gamma_2) \oplus ...$$

avec $\gamma_1, \gamma_2, ...$ chemins de longueur ℓ de source j et de but k, et $\pi = B^{\otimes 0} \oplus B^{\otimes 1} \oplus B^{\otimes 2} \oplus B^{\otimes 3} \oplus ...$ peut être déterminé par *l'algorithme de Roy/Warshal*

On suppose sans perte de la généralité que $S = \{1, 2, ..., m\}$.

Initialiser pour $j, k \in S$: $\pi^0(j,k) = \begin{cases} d((j,k)) & \text{si } (j,k) \in A, \\ \text{élément neutre de } \otimes & \text{si } j = k, \\ \text{élément neutre de } \oplus & \text{sinon.} \end{cases}$

pour $i = 1, 2, ..., m$ faire

  pour $j = 1, 2, ..., m$ faire

    STOP si $\pi^{i-1}(j,j) \neq$ élément neutre de $\otimes$

    pour $k = 1, 2, ..., m$ faire

$$\pi^i(j,k) = \pi^{i-1}(j,k) \oplus \left[\pi^{i-1}(j,i) \otimes \pi^{i-1}(i,k)\right]$$

Idée de l'algorithme:

pour $\pi^i(j,k)$ on considère seulement les chemins ayant comme sommets internes les sommets $1, 2, ..., i$.
Résultats:

- Si on s'arrête à STOP alors il existe un circuit absorbant $\Longrightarrow$ problème mal posé.
- Sinon, pour tout $j, k \in S$: $\pi(j,k) = \pi^m(j,k)$, atteint pour un chemin élémentaire.

Complexité: $\mathcal{O}(m^3)$ plus cher que Bellmann et Dijkstra, mais plus général et nécessitant aucune hypothèse sur le graphe.
Exemple d'un graphe à 10 sommets, on cherche à savoir pour tout $j, k \in S$ s'il existe un chemin avec source j et but k.

Le Laplacien d'un graphe orienté sans boucles à poids

Etant donné un graphe $G = (S, A)$ orienté avec poids, sa matrice d'adjacence $B = B(G)$ définie par $B_{j,k} = d((j,k)) > 0$ si$(j,k) \in A$ et $= 0$ sinon, sa matrice de degrés extérieurs par $D = D(G) = \text{diag}(Be)$, $e = (1, ..., 1)^T$, le Laplacien est défini par $L = L(G) = D - B$.

On considère aussi parfois le Laplacien normalisé $\widetilde{L} = D^{-1}L = I - \widetilde{B}$, avec $\widetilde{B} = D^{-1}B$ une matrice stochastique (à éléments dans $[0, 1]$, $\widetilde{B}e = e$), revient à normaliser les poids.

Le Laplacien : propriétés

$$L = D - B, \quad D = \mathrm{diag}(Be), \quad \widetilde{L} = D^{-1}L = I - \widetilde{B}.$$

Lem 1: $Le = 0$, plus précisément, $\dim(\text{noyau}(L))$ =nombre de composantes strictement connexes de G=multiplicité de la valeur propre 1 de $\widetilde{B}$.
Idée de preuve: après permutation simultanée de lignes/colonnes, B dévient triangulaire inférieure par blocs, avec chaque bloc diagonal de taille minimum, ses indices correspondent à une composante strictement connexe.

Thm 2 de Perron-Frobenius: Supposons que G admet une seule composante strictement connexe, alors 1 est valeur propre dominante de $\widetilde{B}$ (toute autre valeur propre est de module < 1) de multiplicité 1, avec vecteur propre à composantes > 0.

Exemple: pagerank sous google

Une page web est bien classée si des pages web bien classées comportent un lien vers cette page $\Longrightarrow$ modélisation par graphe avec poids $d((i,j))$=nombre de liens sur la page i vers la page j, avec vecteur de scores (loi stationnaire d'une chaîne de Markov)

$$p = p\widetilde{B}, \quad p \geqslant 0, pe = 1.$$

Comme ce graphe n'est pas fortement connexe, pour assurer l'unicité du score on remplace $\widetilde{B}$ par

$$\alpha\widetilde{B} + (1-\alpha)ey^T, \quad y \geqslant 0, y^T e = 1, \alpha \in (0,1)$$

de sorte que 1 devient valeur propre dominante, avec vecteur de scores=vecteur propre à gauche approchée par exemple par la méthode de la puissance.

Le Laplacien pour un graphe non orienté

Etant donné un graphe non orienté $G = (S, A)$ à poids positif, son Laplacien est donné par $L = L(G^*)$ avec $G^* = (S, A^*)$ et A^* obtenu en remplaçant une arête $\{i, j\}$ par les deux arcs (i, j) et (j, i), pondéré par $d_{i,j} = d_{j,i} > 0$. Il en suit que L est symétrique,

$$L_{i,j} = \begin{cases} -d_{i,j} & \text{si } i \neq j \text{ et il existe une arête } \{i, j\} \\ \sum_{\text{sommet } k \text{ adjacent à } i} d_{i,k} & \text{si } i = j \\ 0 & \text{sinon.} \end{cases}$$

On peut donc écrire $L = D - B$ avec B la matrice d'adjacence sommets/sommets pondérée vue avant (symétrique avec zéro sur diagonale), et $D = \text{diag}(Be)$.

Avec $C = C_S^A$ la matrice d'incidence sommets/arêtes de G comportant en colonne $\{j, k\} \in A$ exactement un 1 et un -1 dans les lignes $j, k \in S$, et sinon que des zéros (on compte chaque arête une fois), on montre la factorisation

$$L = D - B = C \Delta C^T, \quad \Delta = \text{diag}(d_a)_{a \in A}.$$

Application: centralité d'un sommet

Comment trouver des "influencers" dans un réseau social?

Dans le cas $d(a) = 1$ pour tout $a \in A$ on sait que

$$(B^\ell)_{i,j} = \text{nombre de chaînes de } i \text{ à } j \text{ de longueur } \ell,$$

alors pour des facteurs de pondération $f_\ell > 0$, le sommet j est bien connecté si

$$(\sum_\ell f_\ell \, B^\ell)_{j,j} \geqslant 0 \text{ est elevé.}$$

Pour la fonction $f(z) = f_0 + f_1 z + f_2 z^2 + ...$, par exemple $f(z) = \exp(z)$, on obtient alors un score (relatif) de centralité du sommet $j \in S$ par

$$f(B)_{j,j} / \text{trace}(f(B)).$$

Propriétés spectrales du Laplacien

Par construction, $Le = 0$, et L est symétrique semi-définie positive. Pour un graphe non orienté à ℓ composantes connexes, L admet alors des éléments propres (λ_j, v_j) avec des valeurs propres

$$0 = \lambda_1 = ... = \lambda_\ell < \lambda_{\ell+1} \leqslant ... \leqslant \lambda_{|S|} \leqslant 2 \max_j L_{j,j},$$

et une base orthonormée de vecteurs propres $v_1, v_2, ..., v_m$. Comme $v_1, ..., v_\ell$ on peut choisir les fonctions caractéristiques de chaque composante connexe.
$\lambda_{\ell+1}$ est dit "gap spectral", il est elevé si le graphe est assez connecté.
Suivant Kirchhof, un conducteur j avec charge $q_j(t)$ au temps t lié à d'autres conducteurs k avec une résistance $d_{(j,k)}$ évolue dans le temps comme

$$\frac{dq_j}{dt}(t) = -\sum_{(j,k)\in A} d_{j,k}(q_j(t) - q_k(t)) = -(Lq(t))_j$$

avec solution $q(t) = \exp(-Lt)q(0)$, qui pour $t \to +\infty$ donne $q(\infty)$, une charge constante sur chaque composante connexe, l'erreur $q(t) - q(\infty)$ étant dominé par $\|q(0)\| \exp(-\lambda_{\ell+1}t)$. La charge/information sur un graphe est alors facilement diffusée si le gap spectral est élevé. Voir simulation.

Application: spectral embedding

Dans la suite, soit G connexe (i.e., $\ell = 1$ et $v_1 = e/\sqrt{|S|}$).
Tâche: On cherche à associer à chaque sommet $j \in S$ un point $x_j \in \mathbb{R}^k$ (colonne d'indice j d'une matrice $X \in \mathbb{R}^{k \times S}$) de sorte que deux points liés par une arête de grand poids soient proches. Ce nuage de points devrait satisfaire

$$Xe = 0 \quad \text{points de moyenne 0,}$$
$$XX^T = I \quad \text{dispersion normalisée par matrice de covariance.}$$

Notre objectif à minimiser

$$\sum_{(i,j)\in A} d_{(i,j)} \|x_i - x_j\|^2 = \text{trace}(XLX^T) = \text{trace}(LX^TX)$$

avec un projecteur X^TX de rang k. La solution est donnée par $X^T = (v_2, v_3, ..., v_{k+1})$.

...à comparer avec l'erreur d'une ACP sur les axes $v_{k+2}, ..., v_{|S|}$.

Application: partitionner graphe 1

Tâche: Trouver une partition $S_0, ..., S_k$ de S de sorte que les poids des arêtes reliant deux ensembles distincts soient "petit".

Pour $Y \subset S$ considérons son vecteur caractéristique $e_Y \in \{0,1\}^S$ avec $(e_Y)_k = 1$ ssi $k \in Y$.

$e_Y^T e_Y = |Y|$ le cardinal du sous-ensemble

$$cap(Y) := e_Y^T L e_Y = \sum_{\{i,j\} \in A} d_{i,j} \underbrace{((e_Y)_i - (e_Y)_j)^2}_{\in \{0,1\}} = \sum_{\substack{\{i,j\} \in A \\ i \in Y, j \notin Y}} d_{i,j}.$$

Notre objectif est alors de minimiser

$$ratiocap(S_0, ..., S_k) = \frac{1}{2} \sum_{p=0}^{k} \frac{cap(S_p)}{|S_p|} = \frac{1}{2} \sum_{p=0}^{k} \frac{e_{S_p}^T L e_{S_p}}{e_{S_p}^T e_{S_p}}.$$

Application: partitionner graphe 2

Tâche: Trouver une partition $S_0, ..., S_k$ de S de sorte que les poids des arêtes reliant deux ensembles distincts soient "petit".

Notre objectif est de minimiser

$$ratiocap(S_0, ..., S_k) = \frac{1}{2}\sum_{p=0}^{k} \frac{cap(S_p)}{|S_p|} = \frac{1}{2}\sum_{p=0}^{k} \frac{e_{S_p}^T L e_{S_p}}{e_{S_p}^T e_{S_p}}.$$

NB1: Problème NP-dur, peut se modéliser à l'aide de $k+1$ vecteurs bivalentes dont la somme vaut e.
NB2: Comme $E = [\frac{e_{S_0}}{\sqrt{|S_0|}}, ..., \frac{e_{S_k}}{\sqrt{|S_k|}}]$ vérifie $E^T E = I$,

$2\, ratiocap(S_0, ..., S_k) = \text{trace}(LEE^T) \geqslant$ optimum spectral embedding,

on relâche des contraintes d'intégrité...
NB3: Pour $k = 1$, $\exists$ inégalités réciproques dites de Cheeger.

Algo pour partitionner graphe

1. On associe $x_j \in \mathbb{R}^k$ pour tout $j \in S$ par spectral embedding.
2. On cherche à minimiser (par K-means) sur toute partition et tout centre $\mu_p \in \mathbb{R}^k$

$$\sum_{p=0}^{k} \sum_{j \in S_p} \|x_j - \mu_p\|^2.$$

3. Initialiser centres $\mu_0, ..., \mu_p \in \mathbb{R}^k$
4. Partition minimisante pour centres fixes:

$$\forall j \in S : \quad j \in S_p \text{ si } \|x_j - \mu_p\| = \min_q \|x_j - \mu_q\|$$

5. Centres minimisants pour partition fixe:

$$\forall p = 0, ..., k : \quad \mu_p = \frac{1}{|S_p|} \sum_{j \in S_p} x_j$$

6. Retour à 4 jusqu'a partition stable.

Exercice 11

France Telecom doit satisfaire la demande de ses usagers pendant une période $[0, T]$: soit $f(t)$ le nombre d'abonnés demandant leur raccordement sur un central téléphonique à l'instant t (on suppose f strictement croissante). Pour satisfaire cette demande il va falloir investir dans les modules d'extension.
Chaque module a une capacité C et un coût γ. Avoir installé j modules implique qu'on peut satisfaire la demande dans la période $[0, t_j]$, avec $t_j := f^{-1}(jC)$. A ce moment t_j de saturation, on doit ajouter un certain nombre de modules aux j modules existants. Les coûts de l'ajout de p modules à l'instant t_j (rapportés à l'instant 0) sont donnés par la formule

$$\frac{\delta + p\gamma}{(1+\tau)^{t_j}},$$

avec δ un coût fixe de mise en chantier, et τ le taux d'actualisation.
Donner un graphe pondéré de sorte qu'une politique optimale d'investissements revient à déterminer un chemin de poids minimum entre deux sommets à préciser.
Solution. avec $C = 1000$, $T = 100$, $f(x) = x^2$, $\tau = 0.05$, $\delta = 5$, $\gamma = 1$

Comment le modèle et la résolution changent-ils si les q modules d'extension disponibles ont des capacités C_k et coûts γ_k, $k = 1, 2, ..., q$?

Exercice 12

Considérons le problème suivant de politique de remplacement de matériel: Sur une période de T années on doit décider au début de chaque année si on achète une voiture neuve ou si on garde l'ancienne, sachant que

- au début de la période on doit acheter une voiture neuve;
- la voiture au choix a un prix de vente constant de 10000E pendant la période envisagée;
- au moment de l'achat d'une voiture neuve, on peut vendre l'ancienne pour un prix dépendant de son âge : pendant les premières 3 ans, une voiture perd chaque année la moitié de sa valeur, et après 10% par année;
- à la fin de la période on vend sa voiture pour le prix ci-dessus,
- l'entretien annuel d'une voiture coûte 1000E si la voiture a moins que 3 ans, après il faut ajouter 500E par année supplémentaire.

Définir un graphe pondéré de sorte que la stratégie la moins chère correspond à un chemin de poids minimum. Solution $T = 10$. Solution $T = 20$.
Idée: on introduit un sommet pour décrire l'état qu'on dispose au début de l'année t (après décision d'achat) d'une voiture âgée de n années.

Comment la situation change-t-elle si on suppose que l'argent dépensé pendant la période est déjà disponible au début de la période, et qu'on a le choix de dépenser une partie de l'argent au cours d'une année, ou de percevoir des intérêts de 10 % par ans ? Solution $T = 10$. Solution $T = 20$.

Exercice 13

Un jardinier doit planter au cours du mois k, $k = 1, 2, ..., s$, une quantité de d_k arbres. Il peut s'approvisionner au début de chaque mois au prix de p_k Euros par arbre. Les arbres achetés sont tous d'abord plantés temporairement dans un bout de champs derrière sa maison et ensuite retirés en fonction des besoins dans ce mois. Sur ce champs on peut planter au maximum r arbres. Au début de la période, aucun arbre se trouve sur le champs. Sachant que les prix varient fortement selon la saison, le jardinier se demande à quel mois il faut acheter quelle quantité d'arbres pour minimiser les frais d'approvisionnement.

(a) Modéliser les différents stratégies d'approvisionnement à l'aide d'un graphe où la situation qu'on dispose (après achat) de i arbres au début du mois k est représentée par un sommet. Pourquoi ce graphe ne comporte pas de circuits? Donner un poids de ce graphe de sorte que la meilleure stratégie d'approvisionnement correspond au chemin de valeur minimale entre deux sommets à préciser.

(b) Résoudre le problème pour les données numériques $r = 4$, $s = 4$, et

k	1	2	3	4
d_k	3	1	4	2
p_k	1	5	2	7

Solution $r = 4$. Solution $r = 7$.

Degré, parcours euleriens

QUESTION: peut-on tracer l'ensemble des arêtes d'un graphe non orienté sans lever le crayon?
DEF1: Le *degré* d'un sommet s d'un graphe non orienté $G = (S, A)$ est le nombre d'arêtes dont ce sommet est une extrémité, noté par $\deg_G(x)$.
DEF2: Une *chaîne eulerienne* (un *cycle eulerien*) est une chaîne (fermée) d'un graphe non orienté qui emprunt chaque arête une et une seule fois.

THEOREME D'EULER:
Soit $G = (S, A)$ un graphe non orienté sans sommets de degré zéro (sommets isolés).
(1) G contient un cycle eulerien si et seulement si G est connexe et chaque sommet est de degré pair.
(2) G contient une chaîne eulerienne si et seulement si G est connexe et soit chaque sommet est de degré pair soit il existent exactement 2 sommets de degré impair.

ICI BUT: preuve constructive de la partie (1) par construction explicite du cycle.

REMARQUE: un graphe non orienté est sans boucles, sinon problèmes (mettre connexité dans la préambule du théorème).

Implication "$\Longrightarrow$" cas cycle

Soit γ un cycle eulerien dans G.

- *G doit être connexe*:
  tout sommet de G est l'extrémité de au moins une arête. Donc γ doit passer par tous les sommets de G. Donc deux sommets quelconques dans G sont reliés par une sous-chaîne de γ.
- *le degré de chaque sommet doit être pair*:
  Pour un sommet $x \in S$, notons par $k(x)$ le nombre de fois qu'on rencontre le sommet x en parcourant le cycle γ. Chaque fois qu'on arrive à x, on repart. Sachant que les arêtes composant un cycle eulerien sont disjoints, le nombre d'arêtes dans γ (et donc dans G) ayant l'extrémité x est égal à $2k(x)$, un nombre pair.

Implication "$\Longleftarrow$" cas cycle

• Fonction $\gamma' = maxi(s, G')$, BUT: partant de s, construire une chaîne maximale dans $G' = (S', A')$ avec interdiction de répétition d'arêtes.

Poser $y_1 \leftarrow s, \qquad \gamma' \leftarrow \emptyset, \qquad \ell \leftarrow 1$

Tant qu'il existe $y_{\ell+1} \in S'$ avec $a := \{y_\ell, y_{\ell+1}\} \in A' \setminus \gamma'$ faire

$\gamma' \leftarrow (\gamma', a), \qquad \ell \leftarrow \ell + 1$

LEMME 1: Si on applique $\gamma' = maxi(s, G')$ à un graphe où les sommets sont tous de degré pair et $\deg_{G'}(s) > 0$ alors γ' est un *cycle*, et avec $G'' = (S', A' \setminus \gamma')$ nous avons $\deg_{G''}(s) = 0$.

PREUVE DU LEMME 1: D'abord, l'algorithme est fini car il y a un nombre fini d'arêtes. Comme $\deg_{G'}(s) > 0$, la chaîne résultante $\gamma = [y_1, ..., y_\ell] = maxi(s, G')$ vérifie $y_1 = s$ et $\ell > 1$ par construction. Si on arrive au sommet $y_j \neq s$ par une arête a' pas utilisée avant on en peut repartir par une arête a pas utilisée avant car le nombre d'arêtes ayant l'extrémité y_j est pair. Donc $y_\ell \neq s$ implique qu'on aurait continué l'algorithme, une contradiction.

• Algorithme principal de construction d'un cycle eulerien dans $G = (S, A)$ "partant" de $s \in S$:

Construire $\gamma \leftarrow maxi(s, G)$

Construire graphe résiduel $A' \leftarrow A \setminus \gamma, \quad G' \leftarrow (S, A'), \quad k \leftarrow 1$

Tant que $k <$ longueur(γ) faire

$k \leftarrow k + 1$. Soit x_k le $k$ième sommet rencontré dans γ.

Si $\deg_{G'}(x_k) > 0$ alors faire

Construire $\gamma' = [y_1, ..., y_\ell] \leftarrow maxi(x_k, G')$

Ajouter le bout $[y_2, ..., y_\ell]$ derrière x_k dans γ

Mettre à jour graphe résiduel $A' \leftarrow A' \setminus \gamma', \quad G' \leftarrow (S, A')$

Exemple sans corrections, avec 1 correction, avec 2 corrections

NB: l'algorithme peut facilement être mis en œuvre en coloriant au fur et à mesure les arêtes dans $A \setminus A'$. Dans ce cas, $\deg_{G'}(x_k)$ coïncide avec le nombre d'arêtes non coloriées avec extrémité x_k, et la fonction *maxi* ne travaille que avec les arêtes non coloriées.

PREUVE DE L'ALGORITHME: Rappelons d'abord l'invariant

- $G' = (S, A') = (S, A \setminus \gamma)$.
- avant chaque appel de *maxi*, le degré de chaque sommet dans G' est pair (on enlève des cycles).

Notons par

- γ le résultat de l'algorithme (qui par Lemme 1 est un circuit non trivial)
- par $\widetilde{S} \subset S$ l'ensemble des sommets rencontrés par γ.

Par construction de l'algorithme et Lemme 1 nous avons pour tout $x \in \widetilde{S}$ la relation $\deg_{G'}(x) = 0$. Par conséquent, les arêtes dans $A \setminus \gamma$ ont comme extrémités seulement des sommets dans $S \setminus \widetilde{S}$, et les arêtes dans γ ont comme extrémités seulement des sommets dans $\widetilde{S}$.
En conséquence, aucun sommet dans $S \setminus \widetilde{S}$ n'est accessible depuis $s \in \widetilde{S}$, mais G est supposé d'être connexe. Donc $S = \widetilde{S}$, $A \setminus \gamma$ est vide et par conséquent γ est eulerien.

COMPLEXITE DE L'ALGORITHME: $\mathcal{O}(n)$ (toute arête est examinée au plus deux fois).

Cas cycle implique cas chaîne

Notons que, dans le Théorème d'Euler, (1) implique (2):

- *si G n'est pas connexe:* voir avant: une chaîne eulerienne γ passerait par tous les sommets de G.
- *si G est connexe et il y a plus que deux sommets $s_1, s_2 \in S$ de degré impair:* supposons par absurde que γ soit une chaîne eulerienne dans G. D'après (1), γ ne peut pas être un cycle, notons par $s_1, s_2 \in S$ les extrémités (distincts) de γ. Je construis un nouveau graphe $\widetilde{G}$ en ajoutant l'arête $a = \{s_1, s_2\}$ à A. En ajoutant a à γ j'obtiens un cycle eulerien dans $\widetilde{G}$, en contradiction avec (1) car le degré de tout sommet différent de s_1, s_2 admet le même degré dans G que dans $\widetilde{G}$.
- *si G est connexe et tout sommet est de degré pair:* d'après (1), je peux construire un cycle eulerien étant aussi une chaîne eulerienne
- *si G est connexe et il y a exactement deux sommets $s_1, s_2 \in S$ de degré impair:* je construis un nouveau graphe $\widetilde{G}$ en ajoutant l'arête $a = \{s_1, s_2\}$ à A. Ce nouveau graphe satisfait aux hypothèses de partie (1), soit γ un cycle eulerien dans $\widetilde{G}$. En enlévant a de γ j'obtiens la chaîne désirée.

DETAILS ALGORITHMIQUES:
L'algorithme marche aussi si on ne sait pas d'avance si le graphe contient une chaîne eulerienne ou un cycle eulerien (à condition qu'on part du bon sommet). Il suffit de tester si la chaîne maximale construite par *maxi* est fermée.
Exemple: La maison du Saint Nicolas partant du sommet 1 ou du sommet 4.

Extension au cas d'un multi-graphe=graphe non simple
Ici on permet plusieurs arêtes ayant les mêmes extrémités.

Les notions de degré, chaîne, cycle (eulerien), connexité se généralisent sans difficultés. L'algorithme et le théorème d'Euler restent valables.
Exemple des ponts de Königsberg.

Le postier chinois

Application: *Ramassage scolaire*
Un bus doit passer au moins une fois par toutes les rues d'un village (arêtes d'un graphe non orienté) pour chercher des enfants. Il sera généralement obligé d'emprunter certaines rues à plusieurs reprises pour construire un tel parcours. Chercher à trouver ces rues tout en minimisant la longueur du parcours (on supposera que le bus part de l'école).
Application: *Tracer un graphe/maillage EDP*
On cherche à traçer au mieux l'ensemble des arêtes d'un graphe non orienté sur une table traçante. On a deux modes opératoires:
• *plume basse*: on trace une arête (pas à deux reprises), sa forme étant connue à l'ordinateur,
• *plume haute*: on passe d'un sommet à un autre sans tracer pour positionner la plume.
Proposer une meilleure stratégie minimisant les mouvements de la plume.

Généralement, un graphe $G = (S, A)$ ne permet pas la construction d'un cycle eulerien (le cas d'une chaîne eulerienne se traite d'une manière similaire). On dispose d'un ensemble B d'arêtes supplémentaires ayant leur extrémités dans S (par exemple une copie de A), et pour chaque $b \in B$ d'un poids $d(b) \geqslant 0$. Résoudre

$$\min\Big\{ \sum_{b \in B'} d(b) \quad : \quad B' \subset B, \text{ et le graphe } (S, A \cup B') \text{ permet la construction d'un cycle eulerien} \Big\}.$$

On se ramène à un problème de *couplage parfait de poids minimum* en se basant sur les observations suivantes:

LEMME 1: Soit $G = (S, A)$, alors l'ensemble $S^* \subset S$ des sommets de degré impair est de cardinalité paire.
PREUVE: la somme des degrés des sommets dans S donne deux fois le nombre d'arêtes.

LEMME 2: Supposons que $G = (S, A)$, et que l'on peut construire un cycle eulerien dans $G' = (S, A \cup B')$. Alors pour tout sommet $x \in S^*$ on peut construire une chaîne dans $H = (S, B')$ ayant comme autre extrémité aussi un sommet dans S^*.
PREUVE: Nous avons pour $y \in S$

$$\deg_H(y) = \deg_{G'}(y) - \deg_G(y) \quad \text{impair si et seulement si } y \in S^*.$$

Donc la chaîne est construite par $maxi(x, H)$.

LEMME 3: Soit $(S, A \cup B^*)$ une solution optimale du problème du postier, alors les chaînes du Lemme 2 sont chaînes de poids minimum dans (S, B). De plus, différents chaînes n'ont pas d'arêtes en commun.

PREUVE: Sinon, on trouve une meilleure solution du problème du postier chinois.

Le problème de *couplage parfait de poids minimum* peut être énoncé comme suit:
Etant donné un graphe non orienté pondéré $\overline{G} = (\overline{S}, \overline{A})$, on cherche à extraire un sous-ensemble $\overline{B} \subset \overline{A}$ de sorte que

- couplage parfait: pour tout sommet $s \in \overline{S}$ il existe une arête dans $\overline{B}$ ayant l'extrémité s
- la somme des poids des éléments dans $\overline{B}$ est minimum.

Formulation mathématique:

$$\min\left\{\sum_{a\in\overline{A}} p(a)\,x(a) : \begin{array}{l}\text{Pour tout } a \in \overline{A},\ x(a) \in \{0,1\},\ \text{et}\\ \text{pour tout } s \in \overline{S},\ \text{la somme des } x(a)\\ \text{avec } a \text{ ayant l'extrémité } s \text{ vaut 1}\end{array}\right\}.$$

THEOREME: Considérons le graphe complet $(\overline{S}, \overline{A})$ ayant $\overline{S} = S^*$ comme ensemble de sommets, le poids d'une arête $(s,t) \in \overline{A}$ étant égal à la valeur d'une chaîne de poids minimum de s à t dans (S,B). Alors une solution optimale B^* du problème du postier chinois induit une solution optimale de couplage parfait de poids minimum dans $(S^*, \overline{A})$. La réciproque est aussi valable.

PREUVE: Lemme 3 montre qu'une solution optimale du postier nous donne un couplage parfait, du même poids. Montrons réciproquement qu'un couplage parfait $\overline{B}$ de poids minimum donne lieu à un ensemble $B' \subset B$ candidat pour une solution optimale du postier, du même poids. Par définition, l'ensemble des extrêmités des plus courtes chaînes associées aux éléments de $\overline{B}$ coïncide avec S^*. Donc, en notant par B' l'ensemble des arêtes de ces chaînes, le degré de tout sommet dans le nouveau graphe $(S, A \cup B')$ est pair, et le théorème d'Euler nous assure que la construction d'un cycle eulerien est possible.

DETAILS ALGORITHMIQUES:

Le calcul des plus courtes chaînes demande $\mathcal{O}(m^3)$ opérations (par exemple Warshal)

La résolution du problème de couplage parfait de poids minimum demande $\mathcal{O}(mn^2)$ *ou* $\mathcal{O}(m^3)$ opérations suivant l'approche d'Edmonds ('65), décrite dans Gondran-Minoux, pp.293-297, et peaufinée par Gabow et Lawner ('76)

Notre approche branch and bound peut être bien plus coûteuse...

Voir simulation par résolution d'un PLNB par branch and bound:

- Un exemple ou il faut rajouter 2 arêtes
- L'exemple classique d'Euler: les ponts de Königsberg
- Un exemple ou il faut rajouter 5 arêtes
- Et finalement un exemple ou les plus courtes chaînes ne sont pas réduites à une arête.

Le voyageur de commerce

Un circuit/cycle hamiltonien
est un chemin (pour les graphes orientés) fermé ou une chaîne (sans orientations) fermée qui passe une et une seule fois par tous les sommets.
Le *voyageur de commerce* cherche à faire un parcours permettant de passer une et une seule fois par chaque ville tout en minimisant la longueur du trajet $\longrightarrow$ *circuit hamiltonien de poids minimum*.

On ne connaît aucun algorithme de complexité polynômiale $\mathcal{O}(m^p n^q)$ pour trouver un circuit hamiltonian.

• Les algorithmes de recherche (et de résolution VdC) sont soit heuristiques soit passent par une résolution "branch and bound" d'un programme linéaire aux nombres entiers (PLNE). Voir simulations sous MATLAB. SCILAB comporte aussi des commandes *hamilton, salesman*.

• Un théorème de type Euler pour la caractérisation d'un graphe comportant un circuit/cycle hamiltonien n'est pas connu. Certains auteurs ont donné des conditions suffisantes d'existence ou des conditions nécessaires d'existence, par exemple:

Ore (1960): Dans un graphe non orienté G à m sommets, si pour tout couple de sommets non reliés par une arête la somme des degrés est $\geqslant m$ alors G permet la construction d'un cycle hamiltonien.

CERTAINS PROBLEMES QUI SE REDUISENT A LA RECHERCHE D'UN CIRCUIT HAMILONIEN:

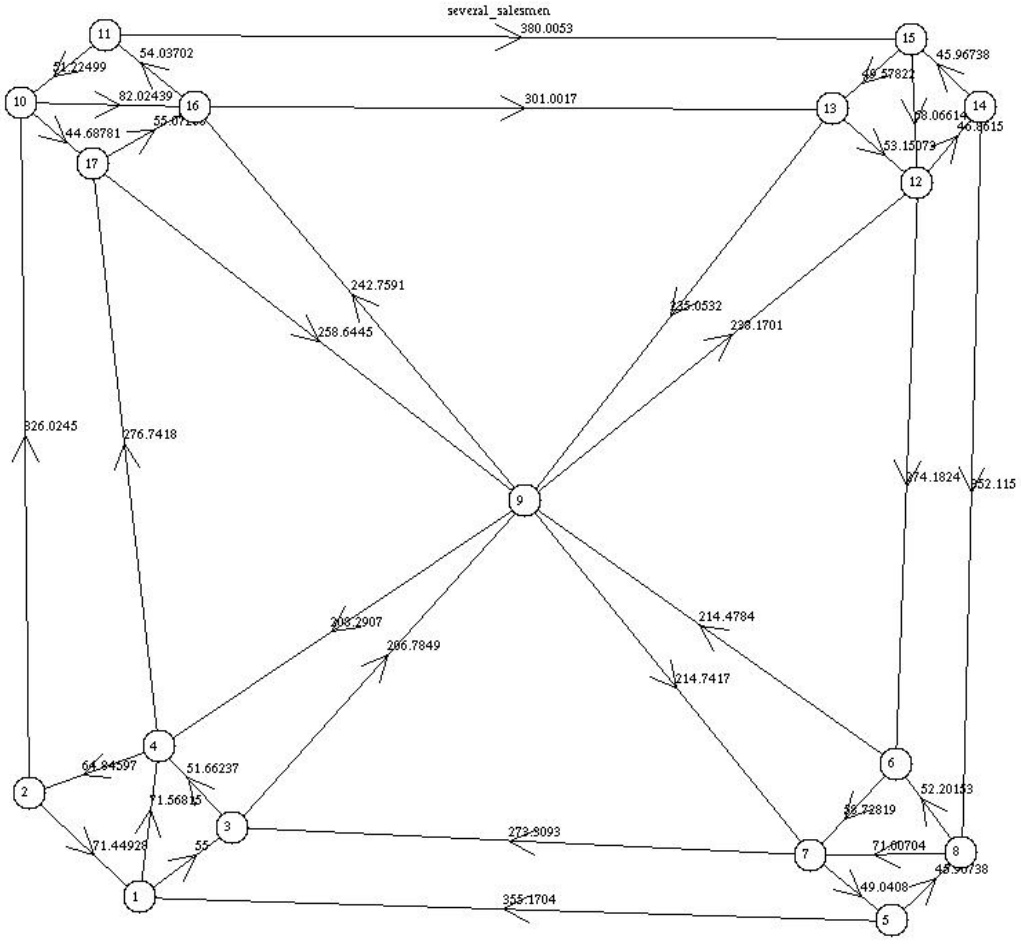
- *recherche d'un chemin hamiltonien:* on ajoute à $G = (S, A)$ un nouveau sommet ω et des arcs (ω, x) et (x, ω) pour tout $x \in S$;
- *recherche d'un cycle hamiltonian:* on peut associer à un graphe non orienté G un graphe orienté G' en faisant correspondre à une arête dans G deux arcs dans G';
- *recherche d'un cycle eulerien:* on peut associer à un graphe non orienté $G = (S, A)$ son "*line graph*" $L(G) = (S', A')$ avec $S' = A$ et $(x, y) \in A'$ si les arêtes x, y dans G ont une extrémité commune;
- *problème des multiples VdC = trouver un ensemble de k circuits ayant un seul sommet ω en commun et rencontrant tous les autres sommets une et une seule fois:*
  remplacer le sommet ω par un graphe complet G_0 à k sommets, chaque arc (x, ω) ou (ω, x) est remplacé par k arcs du même type;
- *recherche d'un circuit (dit préhamiltonien) de poids minimum qui passe au moins une fois par chaque sommet d'un graphe $G = (S, A)$:*
  = VdC pour le graphe complet $G' = (S, A')$, le poids d'un arc (x, y) dans G' étant le poids d'un chemin de poids minimum de x à y dans G.

QUELQUES DETAILS POUR PLUSIEURS VdC

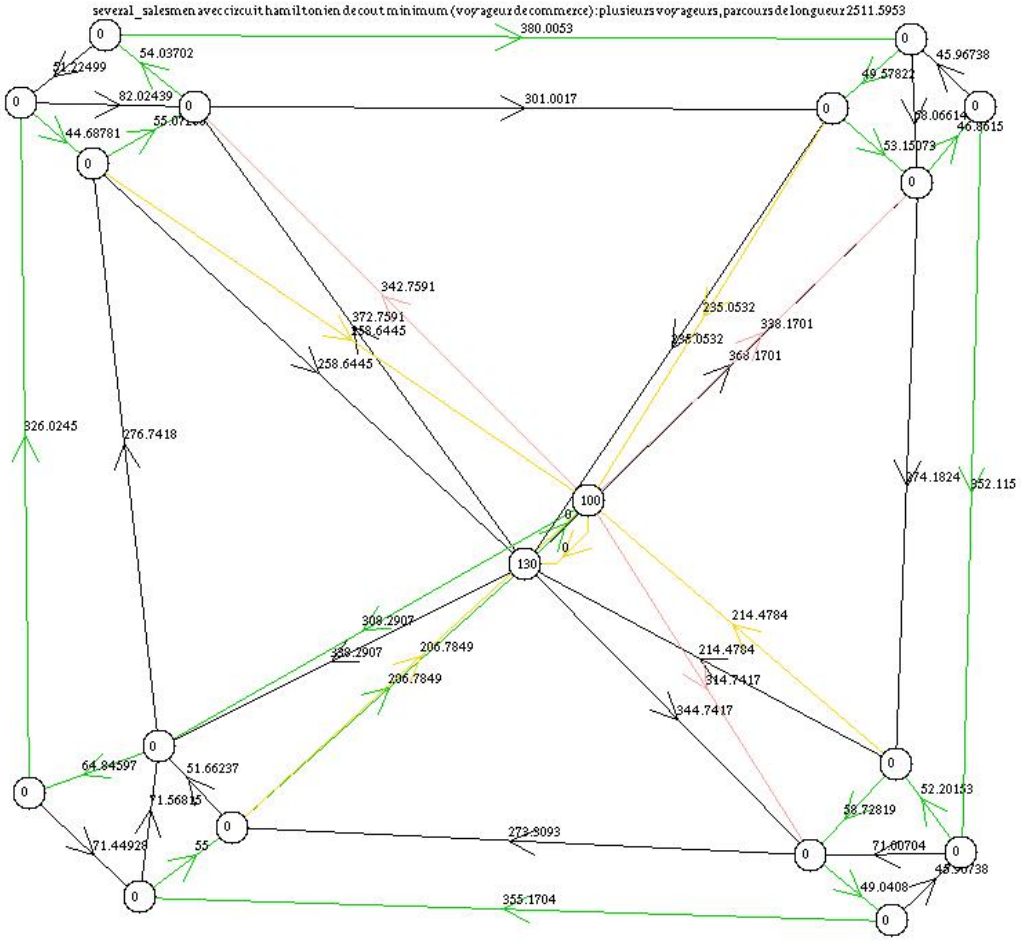
Supposons qu'il y a M voyageurs en un seul sommet qui peuvent chacun faire une tournée sachant que chaque ville est visitée une seule fois par un seul voyageur de commerce, et le coût d'employer le voyageur no k est donné par c_k.
Le sommet de départ y_0 est remplacé par M sommets $y_1, ..., y_M$. On remplace les arcs (pour $j = 1, ..., M$)

$$\begin{aligned} (y_0, x) &\longrightarrow (y_j, x), \quad \text{valeur}(y_j, x) = \text{valeur}(y_0, x) + c_j, \\ (x, y_0) &\longrightarrow (x, y_j), \quad \text{valeur}(x, y_j) = \text{valeur}(x, y_0), \end{aligned}$$

et on ajoute tous les arcs (y_j, y_k) de valeur 0.
Si dans ce nouveau graphe on construit un circuit hamiltonien, alors y_k est utilisé ssi il existe un arc dans ce circuit partant de y_k et allant vers un sommet différent de $y_1, ..., y_M$.
Simulation



Le graphe avant ajout d'arcs supplémentaires.



Le graphe après ajout d'arcs supplémentaires.

Problème de coloration de sommets

TACHE: Colorier les sommets d'un graphe non orienté $G = (S, A)$ de sorte que deux sommets adjacents (reliés par une arête) n'ont pas la même couleur.
DEF: On appelle *nombre chromatique* de G, noté par $\gamma(G)$, le nombre minimum de couleurs nécessaires.

REMARQUE: Cette tâche inclut la tâche de coloriage d'arêtes de sorte que deux arêtes ayant une extrémité en commun n'ont pas la même couleur: colorier les sommets du line graph $L(G) = (A, A')$.
L'*indice chromatique* $q(G)$ est le nombre minimum de couleurs pour la coloration d'arêtes.

APPLICATIONS COLORIAGE SOMMETS:

- coloriage de la carte d'Europe/des régions en France,
- transport de produits chimiques
- problème d'aquariophile,
- organisation d'examen (disponibilité élèves),
- ouverture de magasins, ...

Considérons deux exemples plus en detail.

Organisation d'examens

Un certain nombre d'étudiants doit passer des examens. Pour chaque étudiant $\ell \in \{1, ..., s\}$ on connaît la liste d'examens E_ℓ qu'il doit passer.

Organisation d'examens

Un certain nombre d'étudiants doit passer des examens. Pour chaque étudiant $\ell \in \{1, ..., s\}$ on connaît la liste d'examens E_ℓ qu'il doit passer.

• *Examens écrits=Disponibilité étudiants:* sous l'hypothèse que deux examens sont organisés simultanément (dans la même salle) seulement si aucun étudiant passe les deux épreuves, combien de séances d'examen faut-il organiser au moins?

Organisation d'examens

Un certain nombre d'étudiants doit passer des examens. Pour chaque étudiant $\ell \in \{1, ..., s\}$ on connaît la liste d'examens E_ℓ qu'il doit passer.

• *Examens écrits=Disponibilité étudiants:* sous l'hypothèse que deux examens sont organisés simultanément (dans la même salle) seulement si aucun étudiant passe les deux épreuves, combien de séances d'examen faut-il organiser au moins?

Solution: on colorie les sommets du graphe $G = (S, A)$ avec $S = E_1 \cup E_2 \cup ... \cup E_s$, et $(x, y) \in A$ s'il existe un ℓ avec $x, y \in E_\ell$.

Organisation d'examens

Un certain nombre d'étudiants doit passer des examens. Pour chaque étudiant $\ell \in \{1, ..., s\}$ on connaît la liste d'examens E_ℓ qu'il doit passer.

• *Examens écrits=Disponibilité étudiants:* sous l'hypothèse que deux examens sont organisés simultanément (dans la même salle) seulement si aucun étudiant passe les deux épreuves, combien de séances d'examen faut-il organiser au moins?
Solution: on colorie les sommets du graphe $G = (S, A)$ avec $S = E_1 \cup E_2 \cup ... \cup E_s$, et $(x, y) \in A$ s'il existe un ℓ avec $x, y \in E_\ell$.

• *Examens oraux=Disponibilité étudiants et professeurs:* supposons que chaque étudiant doit être interrogé pendant une heure par le professeur du module suivi. Combien d'heures sont nécessaires au moins?

Organisation d'examens

Un certain nombre d'étudiants doit passer des examens. Pour chaque étudiant $\ell \in \{1, ..., s\}$ on connaît la liste d'examens E_ℓ qu'il doit passer.

• *Examens écrits=Disponibilité étudiants:* sous l'hypothèse que deux examens sont organisés simultanément (dans la même salle) seulement si aucun étudiant passe les deux épreuves, combien de séances d'examen faut-il organiser au moins?
Solution: on colorie les sommets du graphe $G = (S, A)$ avec $S = E_1 \cup E_2 \cup ... \cup E_s$, et $(x, y) \in A$ s'il existe un ℓ avec $x, y \in E_\ell$.

• *Examens oraux=Disponibilité étudiants et professeurs:* supposons que chaque étudiant doit être interrogé pendant une heure par le professeur du module suivi. Combien d'heures sont nécessaires au moins?
Solution: on colorie les arêtes du graphe $G = (S_1 \cup S_2, A)$ avec $S_1 = \{1, 2, ..., s\}$, $S_2 = E_1 \cup E_2 \cup ... \cup E_s$ supposé disjoint de S_1, et $\{x, y\} \in A \subset S_1 \times S_2$ si $y \in E_x$.

Flotte d'avions

Une compagnie aérienne doit satisfaire des demandes de disponibilité d'avions dans des intervalles de temps $I_j = [a_j, b_j]$ données pour $j = 1, ..., m$. On se demande combien d'avions seront nécessaires.

Flotte d'avions

Une compagnie aérienne doit satisfaire des demandes de disponibilité d'avions dans des intervalles de temps $I_j = [a_j, b_j]$ données pour $j = 1, ..., m$. On se demande combien d'avions seront nécessaires.

Solution: On colorie les sommets du *graphe d'intersection* $G = (S, A)$ avec $S = \{1, 2, ..., m\}$, et $\{x, y\} \in A$ si $I_x \cap I_y$ est non vide.

Encadrement du nombre/indice chromatique

Soit $G = (S, A)$ un graphe non orienté (simple), et $d(G)$ le plus grand des degrés des sommets de G.
THM 1: Nous avons $q(G) \in \{d(G), d(G) + 1\}$.

THM 2: Si $d(G) \leqslant m - 2$ alors $\gamma(G) \leqslant q(G)$.

COROLLAIRE: $\gamma(G) \leqslant d(G) + 1$.

NB: Bien entendu, toute coloration concrète donne une autre borne sup pour $\gamma(G)$ ou $q(G)$.

DEF.: On appelle *clique* tout ensemble $C \subset S$ de sorte que le sous-graphe de G engendré par C est complet, c.-à-d., $\{x, y\} \in A$ pour tout $x, y \in C$ disjoints.

THM 2: $\gamma(G) \geqslant \omega(G) :=$ taille maximale d'une clique.
PREUVE: Les sommets d'un graphe complet devraient tous avoir des couleurs différentes.

THM 3: Considérons le graphe complémentaire $\overline{G} := (S, (S \times S) \setminus A)$. Alors $\gamma(G) \geqslant m/\omega(\overline{G})$.

THM 4 (Appel et Haken, '76) : Pour un graphe planaire, $\gamma(G) \leqslant 4$.

Un algorithme heuristique de coloriage

• Trouver un coloriage optimum est un problème difficile... (NP complet)

• L'algorithme suivant donne un coloriage admissible de sommets et donc une borne supérieure pour $\gamma(G)$.

Définir une liste $\Sigma \leftarrow [x_1, ..., x_m]$ des sommets

de sorte que $\deg_G(x_1) \geqslant \deg_G(x_2) \geqslant ... \geqslant \deg_G(x_m)$

Tant que Σ est non vide faire

    Choisir une nouvelle couleur dite d'usage

    Initialiser $adja(x) \leftarrow faux$ pour $x \in \Sigma$ et poser $compteur \leftarrow 1$

    Tant que $compteur \leqslant longueur(\Sigma)$ faire

        Poser $x \leftarrow \Sigma[compteur]$

        Si $adja(x) = faux$ alors faire

            Colorie x avec couleur d'usage et enlever x de Σ

            Pour y voisin de x faire $adja(y) \leftarrow vrai$

        sinon faire $compteur \leftarrow compteur + 1$

COMPLEXITE: $\mathcal{O}(m^2)$.

THM: Le nombre de couleurs utilisées par notre algorithme est $\leqslant d(G)+1$.

PREUVE: Supposons qu'un sommet y n'était pas colorié par une des premières $k := d(G)$ couleurs. Dans ce cas, dans chaque itération $j = 1, ..., k$ il existait un $y_j \in S$ voisin de y qui était colorié à l'étape j. Comme on colorie seulement des sommets pas encore coloriés, les $y_1, ..., y_k$ sont distincts. Par conséquent, au début de l'itération no $j = k+1$, tous les voisins de y sont déjà coloriés, ce qui implique que, au cours de cette itération, la variable $adja(y)$ ne change pas de valeur. Donc y sera colorié à l'itération $k+1 = d(G)+1$.

COROLLAIRE: Soit $\deg_G(x_1) \geqslant \deg_G(x_2) \geqslant ... \geqslant \deg_G(x_m)$ et $k \in \{0, ..., m-1\}$ avec $\deg_G(x_j) \leqslant k$ pour $j = k+1, ..., m$. Alors le nombre de couleurs utilisées par notre algorithme est $\leqslant k+1$.

PREUVE: Voir avant. Par construction, le sommet x_j est colorié au plus tard dans l'itération j car le premier élément de Σ dans la boucle imbriquée est colorié.

Problèmes de flot

Matrice d'incidence "sommets-arcs" pour un graphe $G = (L, J)$ orienté:

$$A = (A_\ell^j)_{\ell \in L}^{j \in J}, \quad \text{avec } A_\ell^j = \begin{cases} 1 & \text{si } \ell = \text{source}(j), \\ -1 & \text{si } \ell = \text{but}(j), \\ 0 & \text{sinon} \end{cases}$$

(pour les graphes non orientés on posera $A_\ell^j = 1$ pour les deux extrémités ℓ de j, mais, à ce moment, la matrice A n'est plus forcément totalement unimodulaire).

Soit $G = (L, J)$ un graphe orienté, avec matrice d'incidence "sommets-arcs" A, et $\underline{x}, \overline{x} \in \mathbb{R}^{|J|}$. On appelle *flot* tout vecteur $\phi \in \mathbb{R}^{|J|}$ vérifiant $A\phi = 0$. Il s'agit d'une lois de conservation:

$$\forall \ell \in L : \quad \underbrace{\sum_{j=(k,\ell) \in J} \phi_j}_{\text{quantité entrante en } \ell} = \underbrace{\sum_{j=(\ell,k) \in J} \phi_j}_{\text{quantité sortante de } \ell} .$$

Le flot est dit *réalisable* (ou compatible) si $\underline{x} \leqslant \phi$ (bornes inf de capacité) et $\phi \leqslant \overline{x}$ (bornes sup de capacité). Pour $s \neq p \in L$, ϕ est dit flot de source s et de puits p si, après l'ajout de l'arc (p, s) dit *de retour* et définition appropriée de $\phi_{(p,s)}$, le ϕ devient un flot.

Problème de flot maximum

En mots, il s'agit d'envoyer un maximum depuis une source s vers un puits p, tout en respectant des contraintes de conservation (flot) et de capacités.
Mathématiquement parlant, on ajoute un arc de retour (p, s) à notre graphe, et on cherche parmi l'ensemble des flots réalisables celui qui maximise $\phi_{(p,s)}$.
On se ramène alors au problème d'optimisation linéaire

$$\min\{f\phi : A\phi = 0, \underline{x} \leqslant \phi \leqslant \overline{x}\}, \quad f^{(p,s)} = -1, \quad \forall j \neq (p,s) : \quad f^j = 0.$$

A cause de la structure particulière de la matrice de coefficients, rappelons que des données entières donnent lieu à une solution optimale qui peut être aussi à composantes entières.
Aussi à on va obtenir un algorithme de résolution plus rapide que SIMPLEX (ne souffrant pas de dégénérescences $\Theta_{\max} = 0$), il s'agit de l'algorithme de Ford-Fulkerson, qui ici est présenté comme cas particulier d'un problème plus général:

on part d'un flot réalisable, on redéfinit les bornes de capacité sur l'arc de retour $\underline{x}_{(p,s)} = \overline{x}_{(p,s)} = +\infty$, et on cherche un flot qui respecte les bornes pour tout arc $j \neq (p, s)$, tout en se rapprochant "au mieux" des exigences de bornes pour l'arc de retour (en maximisant le flot sur l'arc de retour)....

Problème de flot réalisable (aussi dit compatible)

Etant donné un graphe orienté $G = (L, J)$, avec matrice d'incidence "sommets-arcs" A, et $\underline{x}, \overline{x} \in \mathbb{R}^{|J|}$, on cherche un flot réalisable, c'est-à-dire, un vecteur ϕ vérifiant $A\phi = 0, \phi \geqslant \underline{x}, \phi \leqslant \overline{x}$.

On appelle coupe un sous-ensemble $Y \subset L$, de capacité

$$\text{cap}(Y) = \sum_{j=(\ell,k)\in J, \ell\in Y, k\notin Y} \overline{x}_j - \sum_{j=(\ell,k)\in J, \ell\notin Y, k\in Y} \underline{x}_j.$$

La coupe Y est dite séparante pour $s, p \in L$ si $s \in Y$ et $p \notin Y$.

THM DE HOFFMANN : Il existe un flot compatible si et seulement si $\text{cap}(Y) \geqslant 0$ pour toute coupe $Y \subset L$.
PREUVE $\Longrightarrow$: Si ϕ est compatible alors

$$\text{cap}(Y) \geqslant \sum_{j=(\ell,k)\in J, \ell\in Y, k\notin Y} \phi_j - \sum_{j=(\ell,k)\in J, \ell\notin Y, k\in Y} \phi_j = 0.$$

PREUVE $\Longleftarrow$ **:** Etant donné un flot ϕ pas encore compatible, disons, $j = (p, s) \in J$ avec $\phi_j < \underline{x}_j$, nous cherchons à changer le flot ϕ

- en augmentant strictement le flot sur l'arc (p, s) (voir algo de Ford-Fulkerson),
- sans vouloir créer des incompatibilités plus importantes sur d'autres arcs.

Si par contre $j = (s, p) \in J$ avec $\phi_j > \overline{x}_j$, nous souhaitons diminuer strictement le flot sur (s, p) (ce qui revient à la même chose que augmenter sur l'arc opposé (p, s), voir ci-dessus) sans créer des nouvelles incompatibilités.

Nous arrivons à la conclusion en montrant qu'une telle augmentation est toujours possible si $\forall Y \subset L :\ \mathrm{cap}(Y) \geqslant 0$. Plus précisément, nous obtenons un algorithme fini à condition que les données soit entières, car à chaque itération on diminue

$$\sum_{j \in J} \max\{0, \phi_j - \overline{x}_j, \underline{x}_j - \phi_j\}$$

par au moins une unité.

Nous allons supposer pour convenance que si $(\ell, k) \in J$ alors l'arc opposé $(k, \ell) \notin J$ (sinon il faudra travailler avec des multi-graphes, ce qui est possible mais formellement plus compliqué).

Nous cherchons à changer le flot ϕ

- en augmentant strictement le flot sur l'arc (p, s),
- sans vouloir créer des incompatibilités plus importantes sur d'autres arcs.

Ceci se fait par construction d'un graphe d'écart $G(\phi) = (L, J(\phi))$, avec $J(\phi)$ construit comme suit: on parcours l'ensemble des $j = (\ell, k) \in J$

- si $\phi_j < \overline{x}_j$ on pose $i = (\ell, k)$, marge$(i) = \overline{x}_j - \phi_j > 0$, $\gamma_i = 1$, et $G(\phi) \leftarrow G(\phi) \cup \{i\}$.
- si $\phi_j > \underline{x}_j$ on pose $i = (k, \ell)$, marge$(i) = \phi_j - \underline{x}_j > 0$, $\gamma_i = -1$, et $G(\phi) \leftarrow G(\phi) \cup \{i\}$.

Ensuite on cherche à construire un chemin γ dans $G(\phi)$ de s à p (on prend le plus court chemin en nombre d'arcs donnant une meilleure complexité), ce qui devient un cycle dans $G = (L, J)$ en ajoutant l'arc de retour (p, s).
Si un tel chemin n'existe pas, en notant Y l'ensemble des sommets accessibles depuis s dans $G(\phi)$, on trouve que

$$\text{cap}(Y) = \sum_{j=(\ell,k)\in J, \ell\in Y, k\notin Y} (\overline{x}_j - \phi_j) + \sum_{j=(\ell,k)\in J, \ell\notin Y, k\in Y} (\phi_j - \underline{x}_j) < 0.$$

Sinon, on met à jour le flot: avec m le plus petit des marges des arcs composant notre chemin (remplacé par $\underline{x}_{(p,s)} - \phi_{(p,s)} > 0$ ou $\phi_{(s,p)} - \overline{x}_{(s,p)} > 0$ si cette quantité est plus petite)

$$\phi_{(\ell,k)} \leftarrow \phi_{(\ell,k)} \begin{cases} +m & \text{si } (\ell,k) \text{ fait partie du chemin ou si } (\ell,k) = (p,s), \\ -m & \text{si } (k,\ell) \text{ fait partie du chemin ou si } (\ell,k) = (s,p), \\ +0 & \text{sinon.} \end{cases}$$

Exemple :

Problème de flot de valeur minimum

Ici on doit résoudre le problème d'optimisation linéaire

$$\min\{f\phi : A\phi = 0, \underline{x} \leqslant \phi \leqslant \overline{x}\},$$

avec A matrice d'incidence "sommets-arcs" d'un graphe orienté $G = (L, J)$ connexe, mais cette fois f n'a pas de structure particulière. On doit alors faire recours à l'algorithme SIMPLEX, mais il existe une version simplifiée qui tient compte de la provenance de A.

NB1: il faudra supprimer une contrainte d'égalité pour obtenir une matrice de rang plein. Donc des bases $I \subset J$ sont de cardinal $m - 1 = |L| - 1$.

NB2: par unimodularité, toute solution de $A^I y = A^j$ est à composantes $\in \{0, \pm 1\}$. Ceci correspond (après changement d'orientation) à un flot de source(j) vers but(j), autrement dit, (L, I) est un sous-graphe connexe à $(m - 1)$ arcs (ce qui est un nombre minimum). Un tel sous-graphe est nommé **arbre**(=graphe connexe et sans cycles).

NB3: trouver λ avec $\lambda A^I = f^I$ revient à
$\forall j = (\ell, k) \in I : d(I)^j = 0 = f^j - \lambda^\ell + \lambda^k$, la formule de la valeur d'une chaîne. Il suffit alors de chercher dans l'arbre (L, I) l'ensemble des chaînes (et leurs longueurs $-\lambda^k$) depuis le sommet 1 ($-\lambda^1 = 0$) vers l'ensemble des sommets $k \in L$.

NB4: pour $j = (\ell, k) \notin I$ nous avons $d(I)^j = f^j - \lambda^\ell + \lambda^k$, facile à évaluer.

NB5: pour $s = (\ell, k) \notin I$, la solution $y = T(I)^s$ de $A^I y = A^s$ peut se calculer en cherchant dans (L, I) une chaîne de ℓ à k. On obtient alors r, $\Theta_{\max}$, le changement de base et la mise à jour du point de base.

Approche pour le VdC : relaxation lagrangienne

Problème primal (P) : $\min\{f(x) : x \in S, g(x) \leqslant 0\}$ avec S fini, $g : \mathbb{R}^n \mapsto \mathbb{R}^m$.
Lagrangien : $L(\lambda, x) = f(x) + \lambda g(x)$ pour un vecteur ligne λ de taille m.
Hypothèse : on a scindé les contraintes de sorte que pour le problème

$$w(\lambda) = \inf\{L(\lambda, x) : x \in S\} \in \mathbb{R} \cup \{-\infty\} \quad \text{pour tout } \lambda \geqslant 0$$

il soit "facile" d'obtenir une solution optimale $x(\lambda) \in S$ (si $w(\lambda) > -\infty$).
Problème dual (D) : $\max\{w(\lambda) : \lambda \geqslant 0\}$.
On a le même **encadrement** que dans la dualité en optimisation linéaire : pour tout x réalisable pour (P) pour tout λ réalisable pour (D)

$$w(\lambda) \leqslant L(\lambda, x) \leqslant f(x).$$

Cet encadrement permettant de définir un critère d'arrêt est à la base de toute méthode moderne de type primal-dual: on cherche simultanément des approximations d'une solution optimale de (P) et de (D). Un tel encadrement devient de plus en plus fin à condition que les deux valeurs optimales coïncident, ce qui malheureusement n'est pas toujours le cas, même si $x(\lambda)$ est réalisable pour (P)

gap de dualité: $\min\{f(x) : x \in S, g(x) \leqslant 0\} - \max\{w(\lambda) : \lambda \geqslant 0\} \geqslant 0$ (> 0 possible).

Exemple 1 (problème primal linéaire) : $\min\{fx : b - Bx \leqslant 0\}$, $S = \mathbb{R}^n$.
$L(\lambda, x) = (f - \pi B)x + \lambda b$.

$$w(\lambda) = -\infty \text{ si } f - \pi B \neq 0, \quad w(\lambda) = \lambda b \text{ si } f - \pi B = 0.$$

Problème dual : $\max\{\lambda b : f - \lambda B = 0, \lambda \geqslant 0\}$, même valeur optimale.

Exemple 2 (problème primal quadratique) :
$\min\{x_1^2 + x_2^2 : 2x_1 + x_2 + 4 \leqslant 0\}$, $S = \mathbb{R}^n$. $L(\lambda, x) = x_1^2 + x_2^2 + \lambda(2x_1 + x_2 + 4)$.
$x(\lambda) = (-\lambda, -\lambda/2)^t$ annulant $\nabla_x L(\lambda, x) = (2x_1 + 2\lambda, 2x_2 + \lambda)$,

$$w(\lambda) = -\frac{5}{4}\lambda^2 + 4\lambda = -\frac{5}{4}(\lambda - \frac{8}{5})^2 + \frac{16}{5}$$

Problème dual : $\max\{w(\lambda) : \lambda \geqslant 0\} = \frac{16}{5}$, même valeur optimale
$w(\frac{8}{5}) = f(x(\frac{8}{5}))$.

Exemple 3 (affectation généralisée) :

$$\min\{\sum_{i,j} c_{i,j}x_{i,j} : x \in S, \forall i, \sum_{j} a_{i,j}x_{i,j} \leqslant b_i\}, \qquad S = \{x \in \{0,1\}^{2\times 2} : \forall j, \sum_{i} x_{i,j} = 1\},$$

Pour trouver $x(\lambda)$ on doit résoudre un problème à objectif séparable

$$\begin{aligned} w(\lambda) &= \min\{\sum_{i}\Big(\Big(\sum_{j} c_{i,j}x_{i,j}\Big) + \lambda^i\Big(-b_i + \sum_{j} a_{i,j}x_{i,j}\Big)\Big) : x = (x_{i,j}) \in S\} \\ &= \sum_{i}\Big(-b_i\lambda_i\Big) + \sum_{j} \min\Big\{\sum_{i}(c_{i,j} + \lambda^i a_{i,j})x_{i,j} : x_{i,j} \in \{0,1\}, \quad \sum_{i} x_{i,j} = 1\Big\}, \end{aligned}$$

avec solution "évidente" $\forall j : x(\lambda)_{i_j,j} = 1$ avec $i_j = \arg\min_i(c_{i,j} + \lambda^i a_{i,j})$.

Points-cols et gap de dualité

$L(\lambda, x) = f(x) + \lambda g(x)$. Posons $\Lambda = \{\lambda \geqslant 0\}$.
Un point $(\overline{\lambda}, \overline{x}) \in \Lambda \times S$ est dit **point-col** si

$$\forall (\lambda, x) \in \Lambda \times S : \quad L(\lambda, \overline{x}) \leqslant L(\overline{\lambda}, \overline{x}) \leqslant L(\overline{\lambda}, x).$$

L'inégalité à droite est équivalent à $L(\overline{\lambda}, \overline{x}) = w(\overline{\lambda})$. Celle à gauche équivaut $g(\overline{x}) \leqslant 0$ ($\Longleftrightarrow \overline{x}$ est réalisable pour (P)), et $\overline{\lambda} g(\overline{x}) = 0$ ($\Longleftrightarrow L(\overline{\lambda}, \overline{x}) = f(\overline{x})$).

Comme de plus $w(\lambda) \leqslant L(\lambda, \overline{x})$ par construction, on vient de démontrer :
$(\overline{\lambda}, \overline{x}) \in \Lambda \times S$ point-col $\Longleftrightarrow$

$\overline{x}$ et $\overline{\lambda}$ sont solutions optimales de (P), et de (D), respectivement, avec $f(\overline{x}) = w(\overline{\lambda})$,

c'est-à-dire, un gap de dualité zéro. Mais on n'a pas forcément $\overline{x} = x(\overline{\lambda})$ (car $w(\lambda)$ peut avoir plusieurs solutions optimales).

Quelques propriétés théoriques (sans preuves)

Soit $E \subset \mathbb{R}^m$ convexe, et $h : E \mapsto R$ concave:
condition de la sécante
$\forall t \in [0,1] \, \forall y, \widetilde{y} \in E : h(ty + (1-t)\widetilde{y}) \geqslant th(y) + (1-t)h(\widetilde{y})$
• $\gamma \in \partial h(\overline{y})$ (ensemble de **sous-gradients**) si
$\forall y \in E : h(y) \leqslant h(\overline{y}) + \gamma(y - \overline{y})$.
• h est dérivable en $\overline{y}$ avec gradient γ si et seulement si $\partial h(\overline{y}) = \{\gamma\}$.
Ici on travaille avec les objets transposés!

Propriété 1: w est concave sur $\Lambda = \{\lambda \geqslant 0\}$, généralement pas de classe $\mathcal{C}^1$.
Propriété 2: $g(x(\lambda)) \in \partial w(\lambda)$.
Propriété 3 (généralisation de Kuhn et Tucker pour les fonctions non différentiables mais concaves): $\overline{\lambda} \in \Lambda$ est solution optimale de (D) ssi

$$\exists \gamma \in \partial w(\lambda) \text{ avec } \forall \lambda \geqslant 0 : (\lambda - \overline{\lambda})\gamma \leqslant 0 \quad (\Longleftrightarrow \gamma \leqslant 0, \overline{\lambda}\gamma = 0).$$

Propriété 4: Soit $\overline{\lambda}$ une solution optimale de (D). Si w est différentiable en $\overline{\lambda}$ alors le gap de dualité vaut zéro, et $x(\overline{\lambda})$ est solution optimale de (P).

Pour S une partie continue du $\mathbb{R}^n$, il existent d'autres conditions suffisantes pour assurer gap de dualité$= 0$, par exemple f, g_j convexes, S convexe fermé, et $((\exists \widetilde{x}$ t.q.$\forall j, g_j(\widetilde{x}) < 0)$ ou
$(\forall j \exists x(j)$réalisable pour (P) t.q. $g_j(x(j)) < 0))$.

Moralité provisoire (pour cas de PLNE)

Pour le problème

$$(P)\text{: } \min\{fx : x \in \mathbb{R}^n \cap S, g(x) = b - Bx \leqslant 0\} \text{ avec } S \text{ fini,}$$

il assez peu probable d'avoir un gap de dualité nul. Notons que

$$w(\lambda) = \lambda b + \min\{(f - \lambda B)x : x \in \mathbb{R}^n \cap S\}$$

ne change pas de valeur si on remplace S par son enveloppe convexe $[S]$ (on obtient un programme linéaire avec solution optimale en un point extrême). Par conséquent, le problème relâché $(\widetilde{P})$ obtenu en remplaçant S dans (P) par $[S]$ (ce qui est fait dans le branch & bound) aura le même problème dual, mais cette fois avec un gap de dualité $= 0$.

On utilise alors la valeur optimale de (D) et une solution optimale λ_* (ou tout simplement une "bonne" approximation en arrêtant un algorithme de résolution avant la fin) pour obtenir une sous-estimation (évaluation rapide) de $(\widetilde{P})$, et en plus des "bonnes" pénalités $f - \lambda_* A$.

Comment alors résoudre (D), un programme convexe avec objectif non différentiable??

La méthode de Polyak/Uzawa pour (D)

Rappel: pour un convexe fermé C, et $y \in \mathbb{R}^n$, $y_0 := \Pi_C(y)$ est l'élément le plus proche de y dans C ssi, pour tout $y_1 \in C$, $\langle y_0 - y, y_1 - y_0 \rangle \leqslant 0$.
Exemple: $C = \{y \geqslant 0\}$, $\Pi_C(y)_j = \max\{0, (y)_j\}$.
d est une **direction de descente** en y pour la distance de y à C $(= \|y - y_0\|$ avec $y_0 = \Pi_C(y))$ si, pour $t > 0$ suffisamment petit,
$\|y + td - y_0\| < \|y - y_0\| \Longleftrightarrow \langle y - y_0, d \rangle < 0$.
Dans notre cas, $\Lambda^* := \{\lambda$ solution optimale de $(D)\} \subset \Lambda = \{\lambda \geqslant 0\}$ sont des convexes fermés. Si $\lambda_0 := \Pi_{\Lambda^*}(\lambda)$ et $\lambda \notin \Lambda^*$, alors,

$\forall \gamma \in \partial w(\lambda)$: $\quad \gamma^t$ est une direction de descente en λ pour la distance de λ à Λ^*,

car $0 < w(\lambda_0) - w(\lambda) \leqslant (\lambda_0 - \lambda)\gamma$. Par contre, nous ne savons pas optimiser le pas t, car γ^t n'est pas forcement de pente en λ pour w (sous-gradient $\neq$ gradient).

Algorithme de Polyak (Uzawa pour $t_k = t$):
donner $(t_k) \subset (0, +\infty)$, $t_k \to 0$, $\sum_k t_k = \infty$.
Itération: on dispose de $\lambda_k \in \Lambda$.
Trouver $x^k = x(\lambda_k) \in S$ avec $w(\lambda_k) = L(\lambda_k, x^k)$,
poser $\gamma = g(x^k)$, et $\lambda_{k+1} = \Pi_\Lambda(\lambda_k + t_k \gamma^t)$.
Sortie: $\lambda_k \to \overline{\lambda}$ solution de (D).

La méthode de Benders pour (D)

On supposera $x^1, x^2, ..., x^K$ une énumération de S (encore à construire), et pour $k = 1, ..., K$ on considère le problème auxiliaire

$$(D_k) : w_{k+1} := \max_{\lambda \geqslant 0} \min_{j=1,...,k} L(\lambda, x^j) = \max\{w : \lambda \geqslant 0, \forall j = 1, ..., k, L(\lambda, x^j) \geqslant w\}.$$

Clairement, $(D_K) = (D)$, et $w_k \geqslant w_{k+1} \geqslant$ valeur optimale de (D). Le pari est que l'on obtient égalité pour $k \ll K$, au moins si on choisit bien les x^j (et une condition d'arrêt?).

Algorithme de Bender :
Itération: on dispose de $x^1, ..., x^k \in S$ distincts.
Trouver solution optimale (w_{k+1}, λ_{k+1}) du programme linéaire (D_k)
$\max\{w : (w, \lambda) \in \mathbb{R}^{m+1}, \lambda \geqslant 0, \forall j = 1, ..., k, w - \lambda g(x^j) \leqslant f(x^j)\}$
Trouver $x^{k+1} = x(\lambda_{k+1})$ avec $w(\lambda_{k+1}) = L(\lambda_{k+1}, x^{k+1}) = \min_{x \in S} L(\lambda_{k+1}, x)$
Arrêt si $w(\lambda_{k+1}) = w_{k+1}$ (ou $k = K - 1$ ou $w_{k+1} - w(\lambda_{k+1})$ petit).

Par construction, $w(\lambda_{k+1}) \leqslant$valeur optimale de $(D) \leqslant w_{k+1}$. Si on trouve égalité $w(\lambda_{k+1}) = w_{k+1}$ alors w_{k+1} est valeur optimale de (D).
Sinon, on ajoute la nouvelle contrainte $w - \lambda g(x^{k+1}) \leqslant f(x^{k+1})$ pas valable pour (w_{k+1}, λ_{k+1}) pour obtenir (D_{k+1}) (résolution par post-optimisation en partant de la solution de (D_k)).

Bien que $w_k \searrow$, on n'a généralement pas de monotonie pour $w(\lambda_k)$.

Retour a l'exemple 3 (affectation généralisée) :

$$\min\{\sum_{i,j} c_{i,j}x_{i,j} : x \in S, \forall i, \sum_{j} a_{i,j}x_{i,j} \leqslant b_i\}, \quad S = \{x = (x_{i,j})_{i,j} : \forall i, j, x_{i,j} \in \{0,$$

ici $A = \begin{bmatrix} 1 & 3 \\ 5 & 2 \end{bmatrix}, b = \begin{bmatrix} 6 \\ 6 \end{bmatrix}, C = \begin{bmatrix} 4 & 3 \\ 1 & 2 \end{bmatrix}$, et donc

$$L(\lambda, x) = -6(\lambda^1+\lambda^2)+\Big[(4+\lambda^1)x_{1,1}+(1+5\lambda^2)x_{2,1}\Big]+\Big[(3+3\lambda^1)x_{1,2}+(2+2\lambda^2)x_{2,2}\Big].$$

Choisir $\lambda_1 = (1,1)$ (de sorte que le premier polyèdre soit borné) et résoudre $w(\lambda_1) = -3$, atteint en $x^1 = x(\lambda_1) = (x_{1,1} = 1 = x_{2,2}, x_{1,2} = 0 = x_{2,1})$, c-à-d, $i_1 = 1, i_2 = 2$, satellite

$$\max\{w : \lambda \geqslant 0, w \leqslant L(\lambda, x^1) = 6 - 5\lambda^1 - 4\lambda^2\}$$

solution optimale $w_2 = 6$, $\lambda_2 = (0,0)$. Résoudre $w(\lambda_2) = 3 < w_2$, atteint en $x^2 = x(\lambda_2) = (x_{2,1} = 1 = x_{2,2}, x_{1,2} = 0 = x_{1,1})$, c-à-d, $i_1 = i_2 = 2$, satellite

$$\max\{w : \lambda \geqslant 0, w \leqslant L(\lambda, x^1) = 6 - 5\lambda^1 - 4\lambda^2, w \leqslant L(\lambda, x^2) = 3 - 6\lambda^1 + \lambda^2\}$$

solution optimale $w_3 = 18/5$, $\lambda_3 = (0, 3/5)$.

Résoudre $w(\lambda_3) = 17/5 < w_3$, atteint en
$x^3 = x(\lambda_3) = (x_{2,1} = 1 = x_{1,2}, x_{2,2} = 0 = x_{1,1})$, c-à-d, $i_1 = 2, i_2 = 1$, satellite

$$\max\{w : \lambda \geqslant 0, w \leqslant L(\lambda, x^1), w \leqslant L(\lambda, x^2), w \leqslant L(\lambda, x^3) = 4 - 3\lambda^1 - \lambda^2\}$$

solution optimale $w_4 = 7/2$, $\lambda_4 = (0, 1/2)$. Résoudre $w(\lambda_4) = 7/2 = w_4$, atteint en $x^4 = x^3$, arrêt avec solution optimale de (D) donnée par $\lambda = \lambda_4$.
GAP de dualité $f(x^4) - w(\lambda_4) > 0$.

Application au VdC non orienté

Soit $G = (L, J, d)$ un graphe à poids non orienté, $L = \{1, 2, ..., m\}$.
Un cycle hamiltonien est un sous-graphe $\overline{G} = (L, I)$ avec les propriétés

(a) $\forall \ell = 1, ..., m :$ $\deg_{\overline{G}}(\ell) \leqslant 2$

(b) $\deg_{\overline{G}}(1) = 2,$ le sous-graphe de $\overline{G}$ induit par $L' = \{2, ..., m\}$ est un arbre.

Introduisons le vecteur caractéristique x de $\overline{G}$ dans G, c'est-à-dire, les variables décisionnelles $x_j \in \{0, 1\}$ pour $j \in J$ avec $x_j = 1$ ssi $j \in I$. La condition (a) devient $Ax - b \leqslant 0$, $b = (2, ..., 2)$, et A la matrice incidence sommets-arêtes de G. Rapportant la contrainte (b) dans S, nous obtenons le Lagrangien

$$L(\lambda, x) = -\lambda b + (\lambda A + f) = -2\sum_{\ell=1}^{m} \lambda^{\ell} + \sum_{j=\{\ell,k\}\in J} (\lambda^{\ell} + \lambda^{k} + d(j))x_j$$

avec la deuxième somme la somme des poids modifiés
$\widetilde{d}(\{\ell, k\}) = \lambda^{\ell} + \lambda^{k} + d(\{\ell, k\}) \geqslant 0$ des arêtes sélectionnés dans I.
Pour trouver $w(\lambda)$ et $x(\lambda) \in S$ il nous faut alors trouver un arbre (L', I') de poids modifié minimum dans le sous-graphe de G induit par L', et finalement ajouter deux arêtes de J de poids modifié minimum avec une extrémité 1 et une autre dans L', pour obtenir I et son vecteur caractéristique $x(\lambda)$.

Algo de Prim pour la recherche d'un arbre de poids minimum

Il s'agit d'un algorithme presque identique à Dijkstra, sauf que le score $\pi(k)$ pour un sommet k colorié au crayon (c'est-à-dire, accessible depuis un sommet marqué à l'encre) est le plus petit poids des arêtes $\{\ell, k\}$ avec ℓ marqué à l'encre.

Initialisation: $\mathcal{A} = \{\}$, écrire $\pi(s) = 0$, $\forall k \neq s : \pi(k) = +\infty$ au crayon.
Itération : Tant qu'il existe encore une valeur $\pi(\ell)$ écrite au crayon
parmi tous les $\pi(k)$ écrit au crayon, trouver $\pi(\ell)$ de valeur minimum
marquer $\pi(\ell)$ à l'encre et, si $\ell \neq s$, ajouter $\{pere(\ell), \ell\}$ à $\mathcal{A}$.
pour tout k voisin de ℓ non marqué à l'encre
si $d(\{\ell, k\}) < \pi(k)$ alors
$\pi(k) = d(\{\ell, k\})$, $pere(k) = \ell$.
Sortie: $\mathcal{A}$ les arêtes d'un arbre de poids minimum.

Exemple pour Prim
VdC avec Benders
VdC avec Polyak