Cours n°5: Modélisation PLNE de problèmes d'optimisation dans les graphes

Nicolas DUPIN

https://github.com/ndupin/ORteaching
http://nicolasdupin2000.wixsite.com/research

version du 20 mars 2022

Issu et étendu d'enseignements donnés à l'ENSTA Paris, Polytech' Paris-Saclay, et à l'université Paris-Saclay

Motivations à parler de théorie des graphes dans ce cours

- La Théorie des Graphes est très présente dans notre société moderne, elle intervient naturellement dans de nombreux domaines : réseaux informatiques, de transport de personnes ou de marchandises, réseaux sociaux ...
- Analyser les données d'un réseau, ou prendre des décisions sur l'infrastructure d'un réseau fait intervenir une large gamme de problèmes d'optimisation dans des graphes.
- Les problèmes d'optimisation dans les graphes se modélisent en PLNE, c'est au moins un bon exercice pour illustrer les techniques et variantes de modélisation PLNE, tout en définissant des variantes de problèmes classiques que vous ne connaissez pas encore, pour modéliser des problèmes réels.
- Modéliser en PLNE les problèmes d'optimisation dans les graphes permet d'avoir une résolution générique, vous l'implémenterez sous GLPK.

Est-ce efficace de résoudre les problèmes d'optimisation dans les graphes par modélisation et résolution PLNE ?

- On avait bien dit que résoudre un PLNE général, et même en trouver une solution réalisable, est un pb NP-complet.
- Quand on a un algo spécifique et polynomial de résolution, à privilégier bien sûr ! (cela peut être sur un sous-ensembles d'instance avec des propriétés spécifiques)
- Dans certains pb NP-complets, des techniques de modélisation et résolution PLNE fournissent les méthodes exactes les plus efficaces en pratique. (ex : coloration)
- Dans certains pb NP-complets, des algorithmes de résolution spécifiques avec une complexité exponentielle présentent des similarités avec la résolution PLNE. (ex : max stable/clique)
- N.B : dans SageMath, la PLNE est utilisée pour l'optimisation dans les graphes, permet d'avoir un code concis et très maintenable. (et qui suit les progrès continuels des solveurs de PLNE)

Quel apport de la PLNE à l'optimisation dans les graphes ?

- Même si on a un problème avec un algo spécifique connu, connaître la modélisation PLNE est utile lorsqu'on considère des problèmes hybrides ou des variantes, la modélisation PLNE est flexible à des variantes, contrairement aux algorithmes spécifiques en général.
- Recherche dans l'équipe ROCS du LISN : la modélisation et résolution PLNE s'appliquent sur des extensions des problèmes classiques, prenant en compte des contraintes spécifiques supplémentaires.
- Apport plus théoriques que l'on va essayer d'illustrer :
  - La théorie de la Programmation Linéaire peut permettre de prouver que des problèmes d'optimisation sont polynomiaux.
  - La Programmation Linéaire peut fournir des algorithmes d'approximation. (on va le définir)
  - La dualité de la Prog. Linéaire s'applique également pour des graphes !

Ouvrage de référence (en accès libre) : Williamson, D. P., & Shmoys, D. B. (2011). The design of approximation algorithms. Cambridge university press. `https://www.designofapproxalgs.com/book.pdf`

L'optimisation dans les graphes, brique essentielle de la RO

- Important de connaître ses classiques de théorie des graphes, les cas polynomiaux d'optimisation dans les graphes, les pb qui se résolvent assez bien en pratique.
- Résoudre un problème de RO peut se modéliser dans un graphe comme un plus court chemin (avec ressources), ou un problème de coloration (ex : emploi du temps cf cours GOL Graphes et Outils Logiques), et se résoudre ave un algo/outil dédié.
- Pour des méthodes de décomposition : important de savoir détecter qu'un sous problème se modélise et se résout avec un algorithme spécifique efficace (flot, plus court chemins)
- Un théorème fondamental fera le lien : thm de Grötschel-Lovász-Schrijver.
- Groupe du GDR RO : "Graphes, Polyèdres et Optimisation Combinatoire"

`https://www.lamsade.dauphine.fr/poc/`
`https://www.lamsade.dauphine.fr/~poc/jpoc9/Chapitre-Polyedres.pdf`

Approches exactes, heuristiques, approximation

- Jusqu'ici, on a parlé de méthodes d'optimisation exacte (prog.dynamique, gloutons dans cas spécifiques, résolution PL/PLNE générique). But : converger le plus rapidement vers une solution prouvée optimale.
- Autre motivation : à une taille d'instances donnée, comment trouver les meilleures possibles solutions en temps imparti (faible) ?
- Variante : Peut-on avoir des algorithmes légers pour résoudre des problèmes d'optimisation dans des calculs embarqués (ex : robotique) ?
- Heuristiques/méta-heuristique : algorithmes plus légers, sans garantie d'optimalité, avec validation expérimentale.
- Méta-heuristique : une heuristique est spécifique à un problème, méta-heuristique désigne une famille large d'heuristiques s'adaptant à différents pb d'optim, avec des opérateurs génériques à définir.
- Algorithme d'approximation : heuristiques avec une garantie de qualité. On va le redéfinir.

$\Longrightarrow$ On présentera des exemples d'heuristiques et d'approximation, pour un panorama plus complet de l'optimisation dans les graphes.

Notations générales pour les graphes dans ce cours (1)

Un graphe orienté ou non-orienté sera toujours noté $G = (V, E)$, V désigne les sommets ("vertex" en anglais), $E \subset V^2$ désigne les arêtes ("edges" en anglais).

N.B : outre l'anglais, les notations sont différentes du cours GOL de LDD1. Ici, on ne considérera pas de multi-arêtes entre deux sommets, on parle de "multi-graphe".

$E \subset V^2$, une arête e s'écrit $e = (v_1, v_2)$ avec $v_1, v_2 \in V$.

Dans un graphe orienté, $e = (v_1, v_2)$ désigne l'arête qui part de v_1 et arrive à v_2.

Pour un graphe non orienté, on peut supposer $e = (v_1, v_2) = (v_2, v_1)$ pour tout $e \in E$. On pourra toujours inclure les cas non-orienté en considérant un graphe orienté après la duplication des arêtes.

On ne considérera pas d'arête entre un sommet et lui même :
$E \cap \{(v, v)\}_{v \in V} = \emptyset$. (cela n'aura pas de sens dans les problèmes de ce cours)

Dans un graphe non orienté, $\delta(v)$ désigne pour tout sommet $v \in V$, l'ensemble des voisins de v : $\delta(v) = \{v_1 \in V, (v, v_1) \in E\}$

Dans un graphe orienté, $\delta_+(v)$ et $\delta_-(v)$ désignent respectivement l'ensemble des voisins sortants de v et entrant dans v :
$\delta_+(v) = \{v_1 \in V, (v, v_1) \in E\}$ et $\delta_-(v) = \{v_1 \in V, (v_1, v) \in E\}$

Notations générales pour les graphes dans ce cours (2)

Si X est en ensemble (d'arêtes ou de sommets par exemple), $A, B \subset X$ deux sous-ensemble de X, on notera avec $-$ la différence ensembliste :

$$A - B = \{x \in A \mid x \notin B\} \tag{1}$$

Ainsi $\delta(v) - \{v\}$ permet de lever l'ambiguité sur les voisins stricts de v sans l'hypothèse $E \cap \{(v, v)\}_{v \in V} = \emptyset$, on gardera cette écriture lorsque cela n'alourdit pas trop les expressions.

Le graphe complémentaire d'un graphe orienté $G = (V, E)$ sera noté $\overline{G} = (V, \overline{E})$ avec $\overline{E} = V^2 - E - \{(v, v)\}_{v \in V}$

Un graphe orienté $G = (V, E)$ est dit complet si $E = V^2 - \{(v, v)\}_{v \in V}$

Le sous graphe induit par $V' \subset V$ d'un graphe orienté $G = (V, E)$ est défini par $G' = (V', E')$ où $E' = \{(v_1, v_2) \in E \mid v_1 \in V' \text{ ET } v_2 \in V'\}$

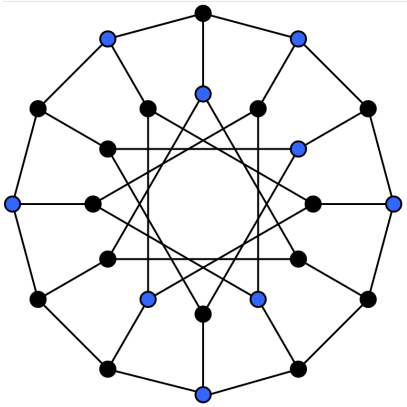
Ces définitions s'étendent aux graphes non orientés, en considérant le graphe orienté induit. ie. pour tout $e \in E$ qui s'écrit $e = (v_1, v_2) = (v_2, v_1)$, on considère deux arêtes dans le graphe orienté induit : (v_1, v_2) et (v_2, v_1)

Plan

Plan

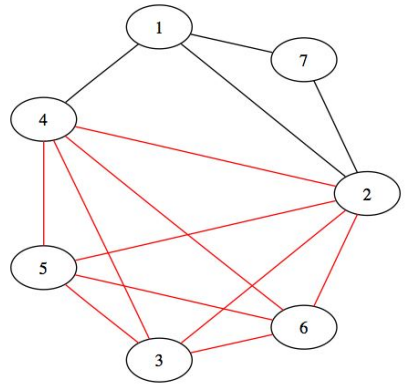
Stables et cliques, définition

Dans un graphe non orienté $G = (V, E)$, un sous-ensemble de sommets $V' \subset V$ est un stable ("independent set" en anglais) ou une clique si :



V' est un stable ssi le sous graphe induit par V' dans G n'a pas d'arête :

$$\forall v_1, v_2 \in V', \quad (v_1, v_2) \notin E$$



V' est une clique ssi le sous graphe induit par V' dans G est complet :

$$\forall v_1, v_2 \in V', \quad (v_1, v_2) \in E$$

Dans le graphe complémentaire, une clique (resp un stable) du graphe originel est un stable (resp une clique)

Stable maximal ou maximum ?

(a) Un stable maximal (b) Un stable maximum

Figure extraite de la thèse de G. Morel (2011)

Un stable $S \subset V$ d'un graphe $G = (V, E)$ peut être dit maximal ou maximum :

- S est un stable MAXIMAL : on ne peut plus ajouter de sommets dans S et avoir toujours un stable. C'est un max "au sens de l'inclusion"
- S est un stable MAXIMUM : il n'existe pas de stable dans G ayant un cardinal strictement plus élevé que S. C'est un max "de cardinalité"
- Un stable maximum est donc forcément maximal.

Idem pour les cliques, maximums et maximales se définissent de même.

Une clique maximum (resp maximale) dans G est un stable maximum (resp maximal) dans $\overline{G}$

Un stable maximum (resp maximal) dans G est une clique maximum (resp maximale) dans $\overline{G}$

Au fait, des applications aux stables et cliques de grande taille ?

- Une clique : une communauté dans un graphe de réseau social.
- Une clique, sous graphe le plus dense, des définitions moins fortes sont utilisées pour l'analyse de réseaux sociaux.
- Stable : une couleur est un stable dans un graphe coloré.
- Des structures très fondamentales, qui auront leur application très spécifique en PLNE. On en entrevoit une dès mtnt
- En préprocessing de la résolution PLNE, les solveurs font des calculs de grandes cliques, on la verra au cours 9 !

Questions algorithmiques qui se posent :

- Comment déterminer qu'un stable est maximal ?
- Comment construire un stable maximal ?
- Comment construire un stable maximal a priori assez grand ?
- Combien peut on trouver de stables maximaux dans un graphe ?
- Quelle complexité à énumérer tous les stables maximaux, pour calculer la taille maximum d'un stable ?
- Comment construire un stable maximum à l'aide de la PLNE ?

N.B : avec le passage au complémentaire, il suffit de traiter les questions sur les stables pour avoir les réponses analogues sur les cliques

Comment déterminer qu'un stable est maximal ?

S est un stable maximal d'un graphe non orienté $G = (V, E)$ ssi

$$\forall v \in V - S, \exists s \in S, v \in \delta(s) \tag{2}$$

Vérification naïve en $O(SV)$

C est une clique maximale d'un graphe non orienté $G = (V, E)$ ssi

$$\forall v \in V - C, \exists c \in C, c \notin \delta(c) \tag{3}$$

Comment construire un stable maximal ?

- On part de $S = \emptyset$, stable, mais non maximal
- On parcourt tous les sommets $v \in V$.
- Si $S \cup \{v\}$ est toujours un stable, ie v n'est lié à aucun sommet de S, on ajoute v à S.
- Si un sommet v n'a pas été sélectionné, c'est parce qu'un sommet précédent est relié à v.
- Au bout d'un parcours, on a construit un stable maximal
- Complexité en $O(V^2)$

$\Longrightarrow$ Comment peut on espérer avoir un stable de grande taille ?

Comment construire un grand stable maximal ?

- Si on sélectionne un sommet avec un grand degré, cela élimine bcp de sommets d'un coup.
- Dans l'algorithme précédent, l'ordre de parcours n'a pas d'importance sur la validité de l'algorithme pour fournir un stable maximal.
- Si on commençait par les noeuds de plus petit degré ?

Algorithmes gloutons

- De manière générale : construction itérative d'une solution suivant une optimisation locale sans jamais remettre en cause une décision précédemment fixée.
- Algorithme glouton le plus simple : on prend les décisions selon l'ordre d'un critère initialement choisi.
- Exemples Sac à dos : mettre dans le sac à dos l'objet avec le meilleur rapport coût/volume parmi les objets pouvant rentrer
- Glouton adaptatif : le critère de choix local est adapté au cours des itérations.

Algorithme glouton pour stableMax

- Quels éléments a t'on a priori dans un stable ?
- Des noeuds de faible degrés !
- On calcule le tableau des degrés $O(|E|)$, et on trie initialement les sommets par degré minimal. $O(|V| \log |V|)$
- On parcourt les sommets dans cet ordre. La liste I des stables est initialement vide.
- Si un sommet considéré peut être ajouté dans I, i.e., voisin d'aucun sommet de I, on l'ajoute dans I.
- Une fois tous les sommets parcourus, on renvoie les sommets de I.
- Heuristique gloutonne en $O(|V|^2)$, comme $|E| + |V| \log |V| \in O(|V|^2)$

Algorithme glouton adaptatif pour stableMax

- Dans l'algorithme précédent, le critère de choix des sommets à ajouter est uniquement basé sur le tri initial des degrés
- Version adaptative : on recalcule les "degrés induits".
- Une fois un sommet v choisi pour le stable, cela revient à considérer un stable dans le graphe où on a retiré les voisins de v.
- Approche "destructive" : on recopie le graphe $G = (V', E')$ (ou un système de marqueurs pour sommets et arêtes éliminés). On stocke le stable courant dans une liste I, initialement vide.
- Tant que $V' \neq \emptyset$:
  - On considère un sommet v de V' de degré minimal dans (V', E').
  - On ajoute v dans I et on actualise (V', E') en enlevant le sommet v et ses voisins et les arêtes contenant v et ses voisins.
- A la fin, on renvoie les sommets de I, un stable.
- Heuristique gloutonne en $O(|V||E| + |V|^2 \log |V|)$, mais on a adapté les choix gloutons aux perturbations induites par les choix précédents.

Algorithmes gloutons et optimalité

- Il existe des problèmes d'optimisation où des algorithmes gloutons fournissent des solutions optimales : sac à dos continu, arbre couvrant de poids minimum (algos de Prim et Kruskal).
- L'algorithme glouton, s'il a une complexité faible, peut fournir de très mauvaises solutions pour la recherche de stable maximum.
- N.B : il existe une théorie pour donner des conditions pour que des algorithmes gloutons fournissent des solutions optimales : la théorie des matroïdes

`https://www.gerad.ca/~alainh/Matroides.pdf`

`https://www.lirmm.fr/~gioan/telecharger/2013%20Gioan%20Ramirez-Alfonsin%20-%20Cours%20sur%20les%20matroides%20(orientes)%20-%20GDRIM2013.pdf`

Algorithme glouton randomisé : GRASP

- Extension GRASP : Glouton adaptatif avec aléatoire.
- Phase gloutonne randomisée : on prend aléatoirement une des toutes meilleures de l'optimisation locale à chaque itération.
- Paramètre glouton randomisé : loi de probabilité, et nombre de meilleures solutions potentielles pouvant être considérées
- "multi-start" : phase gloutonne randomisée peut être répétée pour fournir des solutions différentes, permet de prendre la meilleure solution (ou d'avoir plus de chances d'éviter des infaisabilités de contraintes)
- GRASP : pour chaque solution construite par glouton randomisé, on cherche à améliorer localement la solution obtenue (recherche locale, cf section TSP).

Resende, M. G., & Ribeiro, C. C. (2014). GRASP : Greedy randomized adaptive search procedures. In Search methodologies (pp. 287-312). Springer, Boston, MA.

Enumérer tous les stables maximaux ?

Tout graphe de n sommets a au plus $3^{n/3} \approx 1.4422^n$ stables maximaux.

Moon, J. W. ; Moser, L. (1965), "On cliques in graphs", Israel Journal of Mathematics, 3 : 23–28.

C'est toujours mieux que $O(n^2 2^n)$ avec une énumération naïve "brute force".

Le calcul d'un stable maximum se fait en temps $O(1.1996^n)$ avec un espace mémoire polynomial. (on le verra plus tard)

Xiao, M. et Nagamochi, H. (2017), "Exact algorithms for maximum independent set", Information and Computation, 255 : 126–146.

Sur des graphes de degré maximum 3, la complexité temporelle passe à $O(1.0836^n)$.

Xiao, M. et Nagamochi, H. (2013), "Confining sets and avoiding bottleneck cases : A simple maximum independent set algorithm in degree-3 graphs", Theoretical Computer Science, 469 : 92–104.

Problème polynomial lorsque le degré maximal est 2 et sur des graphes spécifiques "claw-free graphs", "P5-free graphs", graphes parfaits et graphes cordaux (complexité linéaire dans ce cas).

Stables/cliques maximums, formulation PLNE

On utilise des variables binaires $z_v \in \{0,1\}$ pour tout $v \in V$, où $z_v = 1$ si le sommet v est considéré dans le stable (ou la clique).
En cherchant à maximiser le cardinal d'un sous ensemble de V, on maximise l'expression linéaire $\sum_{v \in V} z_v$.

- Les contraintes sur les variables z_v définissant que le sous-ensemble est un stable s'écrivent :

$$\forall e = (v_1, v_2) \in E, z_{v_1} + z_{v_2} \leqslant 1$$

  Si il y a une arête entre v_1 et v_2 alors on a au plus un des 2 sommets dans le stable.

- Les contraintes sur les variables z_v définissant que le sous-ensemble est une clique s'écrivent :

$$\forall e = (v_1, v_2) \notin E, z_{v_1} + z_{v_2} \leqslant 1$$

  Si il n'y a pas d'arête alors on a au plus un des 2 sommets dans la clique.

Stables/cliques maximums, formulation PLNE

$z_v \in \{0,1\}$, où $z_v = 1$ si le sommet v est considéré dans le stable (ou la clique).

Stable max :

$$\begin{array}{lll} & & \max\limits_{z \in \{0,1\}^{|V|}} \sum\limits_{v \in V} z_v \\ s.c: & \forall e = (v_1, v_2) \in E, & z_{v_1} + z_{v_2} \leqslant 1 \\ & \forall v \in V & z_v \in \{0,1\} \end{array} \tag{4}$$

Clique max :

$$\begin{array}{lll} & & \max\limits_{z \in \{0,1\}^{|V|}} \sum\limits_{v \in V} z_v \\ s.c: & \forall e = (v_1, v_2) \notin E, & z_{v_1} + z_{v_2} \leqslant 1 \\ & \forall v \in V & z_v \in \{0,1\} \end{array} \tag{5}$$

Versions pondérées, formulation PLNE

On peut associer un poids c_v à la sélection d'un sommet $v \in V$ et minimiser la somme des pondérations sélectionnées

Stable max pondéré (on reverra ce problème) :

$$\begin{array}{lll} & & \max\limits_{z \in \{0,1\}^{|V|}} \sum\limits_{v \in V} c_v z_v \\ s.c: & \forall e = (v_1, v_2) \in E, & z_{v_1} + z_{v_2} \leqslant 1 \\ & \forall v \in V & z_v \in \{0,1\} \end{array} \tag{6}$$

Clique max pondérée :

$$\begin{array}{lll} & & \max\limits_{z \in \{0,1\}^{|V|}} \sum\limits_{v \in V} c_v z_v \\ s.c: & \forall e = (v_1, v_2) \notin E, & z_{v_1} + z_{v_2} \leqslant 1 \\ & \forall v \in V & z_v \in \{0,1\} \end{array} \tag{7}$$

Remarque : l'ajout de pondérations modifie les perspectives d'un algo glouton : doit on prendre un sommet de degré minimal ou de poids maximal ?

Rappels : relaxation continue et variables incompatibles

Si on considère le PLNE suivant, avec 3 variables binaires incompatibles $x_1, x_2, x_3 \in \{0, 1\}$:

$$\max_{x_1,x_2,x_3 \in \{0,1\}} \quad 1.1x_1 + 1.2x_2 + 1.3x_3$$

$$x_1 + x_2 \leqslant 1 \quad \text{(a)}$$
$$x_2 + x_3 \leqslant 1 \quad \text{(b)}$$
$$x_1 + x_3 \leqslant 1 \quad \text{(c)}$$

Le relâché continu de ce PLNE donne (par un algo de type simplexe) pour optimum $x_1 = x_2 = x_3 = \frac{1}{2}$, et une valeur de borne supérieure 1.8

Si on écrit la contrainte : $x_1 + x_2 + x_3 \leqslant 1$, la relaxation continue donne une solution entière, $x_1 = x_2 = 0$ et $x_3 = 1$, c'est l'optimum entier, de valeur 1.3

La contrainte $x_1 + x_2 + x_3 \leqslant 1$ est plus forte que (a), (b) et (c), elle les implique avec la positivité des variables, et contient moins de points continus réalisables.

A RETENIR : Regrouper les contraintes d'incompatibilité améliore la qualité de la relaxation continue. (on le formalisera et le généralisera)

Premiers éléments d'analyse polyédrale

Soit $\mathcal{P}$ un problème d'optimisation discret portant sur des variables $x \in \mathbb{N}^m$ avec $c.x$ à minimiser.

On suppose avoir deux descriptions linéaires du problème, $Ax \geqslant a$ et $Bx \geqslant b$ pour $x \geqslant 0$: $\{x \geqslant 0 |\ Ax \geqslant a\} \cap \mathbb{N}^m = \{x \geqslant 0 |\ Bx \geqslant b\} \cap \mathbb{N}^m$ définissent les solutions réalisables de $\mathcal{P}$.

La formulation $Ax \geqslant a$ est plus forte si elle contient moins de solutions continues : $\{x \geqslant 0 |\ Ax \geqslant a\} \subset \{x \geqslant 0 |\ Bx \geqslant b\}$

On a alors pour toute fonction objectif définie par des valeurs de c :
$\min_{x \geqslant 0, Ax \geqslant a} c.x \geqslant \min_{x \geqslant 0, Bx \geqslant b} c.x$ même si $\min_{x \in \mathbb{N}^m, Ax \geqslant a} c.x = \min_{x \in \mathbb{N}^m, Bx \geqslant b} c.x$

La relaxation continue est de meilleure qualité avec des contraintes plus fortes.

$\Longrightarrow$ Lien entre inclusions de polyèdres et qualité de relaxation continue

Premiers éléments d'analyse polyédrale, sur l'exemple

Dans l'exemple précédent, on avait comme solutions réalisables :
$(0,0,0),(1,0,0),(0,1,0),(0,0,1)$.

La description linéaire parfaite (= définissant l'enveloppe convexe des points entiers réalisables) est donnée par les contraintes $x_1, x_2, x_3 \geqslant 0$ et $x_1 + x_2 + x_3 \leqslant 1$

Une description linéaire moins bonne est donnée par les contraintes $x_1, x_2, x_3 \geqslant 0$, $x_1 + x_2 + \leqslant 1$, $x_1 + x_3 \leqslant 1$ et $x_2 + x_3 \leqslant 1$

Si x vérifie $x_1, x_2, x_3 \geqslant 0$ et $x_1 + x_2 + x_3 \leqslant 1$, en utilisant par exemple $x_3 \geqslant 0$, on a $x_1 + x_2 \leqslant x_1 + x_2 + x_3 \leqslant 1$

$\Longrightarrow$ La contrainte $x_1 + x_2 + x_3 \leqslant 1$ est plus forte, elle contient les solutions continues définies par $x_1 + x_2 + \leqslant 1$, $x_1 + x_3 \leqslant 1$ et $x_2 + x_3 \leqslant 1$, en supposant $x_1, x_2, x_3 \geqslant 0$.

Il y a stricte inclusion des polyèdres des contraintes des deux formulations, $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ est réalisable sur la seconde formulation linéaire, non réalisable avec $x_1 + x_2 + x_3 \leqslant 1$.

Plus généralement

Si on a un système de contraintes $x_1, \ldots, x_N \geqslant 0$ et $\sum_{n=1}^{N} x_n \leqslant 1$.

Cela implique pour tout $C \subset [\![1; N]\!]$, qu'on a $\sum_{c \in C} x_c \leqslant 1$

L'intérêt est à partir des contraintes déjà écrites d'en obtenir d'autres plus fortes, ie rajouter des éléments positifs avant l'inégalité.

Un procédé général à la planche suivante, on verra d'autres exemples dans ce cours.

Graphe de dépendance, contraintes regroupées sur des cliques

Pour tout PLNE, on peut associer un graphe de dépendance :

Sommets : les variables binaires x et leurs complémentaires $\bar{x} = 1 - x$ du PLNE.

Arêtes : 2 sont reliés si les variables associées sont incompatibles.

Par exemple :
$x_1 + x_2 \leqslant 1$, incompatibilité entre x_1 et x_2 ;
$x_1 \leqslant x_2 \Longrightarrow x_1 + 1 - x_2 \leqslant 1 \Longrightarrow x_1 + \bar{x_2} \leqslant 1$, incompatibilité entre x_1 et $\bar{x_2}$

Dans le graphe de dépendance, une clique $\mathcal{C} = \left((x_i)_{i \in I}, (\bar{x_j})_{j \in J}\right)$ induit la contrainte valide suivante :

$$\sum_{x \in \mathcal{C}} x \leqslant 1 \Longleftrightarrow \sum_{i \in I} x_i + \sum_{j \in J} (1 - x_j) \leqslant 1$$

Exemple de coupe de Clique (1)

max $1.1x_1 + 1.2x_2 + 1.3x_3$

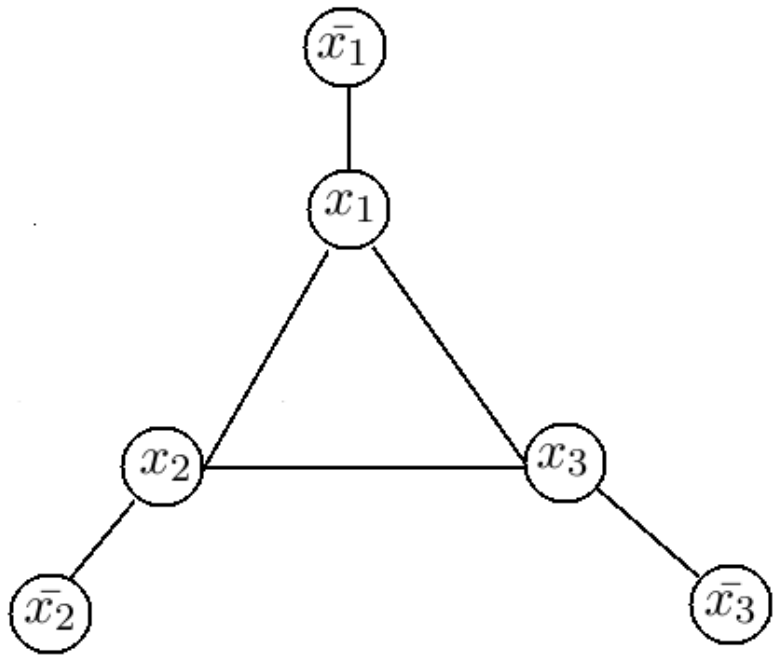
s.t :

$x_1 + x_2 \leqslant 1$

$x_2 + x_3 \leqslant 1$

$x_1 + x_3 \leqslant 1$

$x_1, x_2, x_3 \in \{0, 1\}$



Coupe de clique : $x_1 + x_2 + x_3 \leqslant 1$

Exemple de coupe de Clique (2)

max $1.1x_1 + 1.2x_2 + 1.3x_3 + 1.4x_4$

s.t :

$x_1 + x_2 \leqslant 1$

$x_2 + x_3 \leqslant 1$

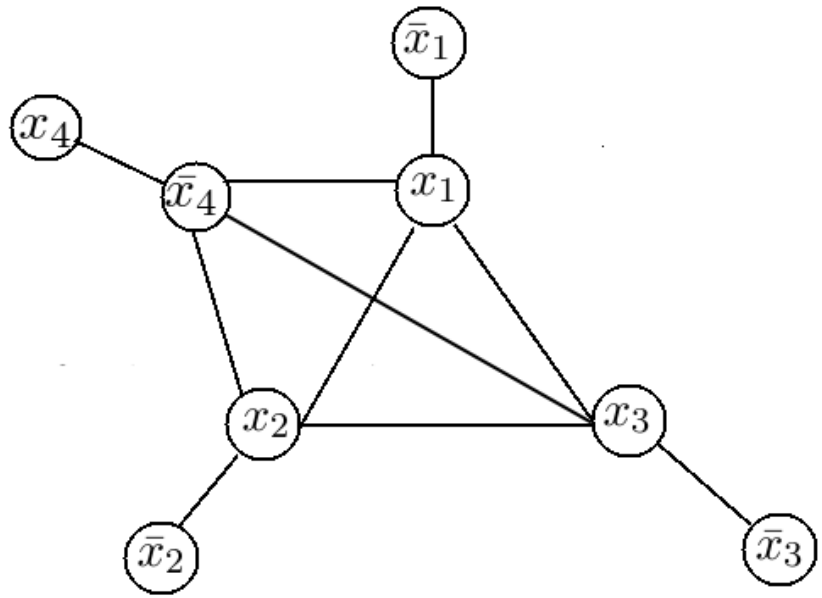
$x_1 + x_3 \leqslant 1$

$x_1 \leqslant x_4$

$x_2 \leqslant x_4$

$x_3 \leqslant x_4$

$x_1, x_2, x_3, x_4 \in \{0, 1\}$



Coupe de clique : $x_1 + x_2 + x_3 + (1 - x_4) \leqslant 1$ cad $x_1 + x_2 + x_3 \leqslant x_4$

Exemple : "min-up/min down" polytope

Contraintes : une machine fonctionne sur au moins L_u périodes discrétisées

Variables :
- $x_t^u \in \{0,1\}$ de "set up", $x_t^u = 1$ ssi la machine u est en fonctionnement à t.
- $y_t^u \in \{0,1\}$ de "start up", $y_t^u = 1$ ssi la machine u démarre à la période t.

$y_t^u = \max(0, x_t^u - x_{t-1}^u) \in \{0,1\}$, se linéarise avec $y_t^u \geqslant 0$, $x_t^u - x_{t-1}^u \leqslant y_t^u$.

On peut écrire les contraintes suivantes, pour qu'un démarrrage à t' implique le fonctionnement pour tout $t \in [\![t', t' + L_u - 1]\!]$:

$$\forall u, \forall (t, t') \ \ tq \ \ 0 \leqslant t - t' < L_u, \ \ y_{t'}^u \leqslant x_t^u \tag{8}$$

Sur L_u pas de temps consécutifs, il ne peut y avoir qu'un seul démarrage :

$$\forall u, \forall t \in [\![L^u, T]\!], \quad \sum_{t'=t-L^u+1}^{t} y_{t'}^u \leqslant 1 \tag{9}$$

Les binaires $y_{t'}^u$ et $1 - x_t^u$ sont incompatibles 2 à 2, la coupe de clique plus forte correspond à la formulation déjà mentionnée :

$$\forall u, \forall t \in [\![L^u, T]\!], \quad \sum_{t'=t-L^u+1}^{t} y_{t'}^u \leqslant x_t^u \tag{10}$$

Regrouper stableMax sur des cliques ?

En partant de la formulation PLNE du Stable max pondéré :

$$\begin{aligned} & \max_{z \in \{0,1\}^{|V|}} \sum_{v \in V} c_v z_v \\ s.c: \quad & \forall e = (v_1, v_2) \in E, \quad z_{v_1} + z_{v_2} \leqslant 1 \end{aligned} \tag{11}$$

Soit $\mathcal{C}$ l'ensemble des cliques maximales du graphe de dépendance : On a la formulation PLNE suivante, qui améliore la relaxation continue de la formulation précédente :

$$\begin{aligned} & \max_{z \in \{0,1\}^{|V|}} \sum_{v \in V} c_v z_v \\ s.c: \quad & \forall c \in \mathcal{C}, \quad \sum_{v \in c} z_v \leqslant 1 \end{aligned} \tag{12}$$

On se ramène à un PLNE avec un nombre potentiellement exponentiel de contraintes, on verra comment ça peut se gérer.

Regrouper stableMax sur des cliques ? (2)

Soit $\mathcal{C}$ un sous ensemble des cliques maximales du graphe de dépendance, qui recouvre toutes les arêtes : pour tout $(v_1, v_2) \in E$, il existe $c \in \mathcal{C}$ tq $v_1, v_2 \in c$
On a la formulation PLNE suivante valide, qui améliore la relaxation continue de la première formulation, en ayant moins de contraintes :

$$\begin{aligned} & \max_{z \in \{0,1\}^{|V|}} \sum_{v \in V} c_v z_v \\ s.c: \quad & \forall c \in \mathcal{C}, \quad \sum_{v \in c} z_v \leqslant 1 \\ & \forall v \in V \quad z_v \in \{0,1\} \end{aligned} \tag{13}$$

Un tel calcul de recouvrement peut se faire avec des successions de phases gloutonnes, en calculant des cliques maximales contenant une arête non marquée précédemment, et en marquant toutes les arêtes d'une clique maximale trouvée.

On verra comment bien générer de telles cliques maximales dans les algorithmes de résolution.

Heuristiques primales et duales

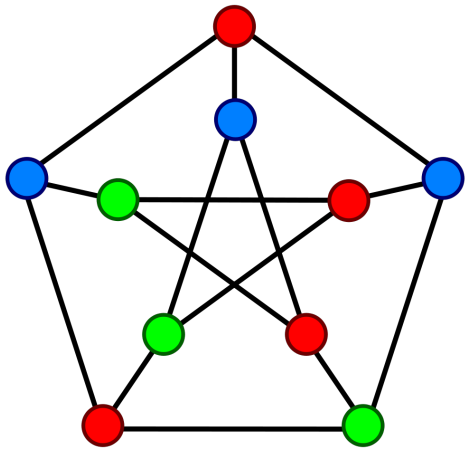
Si on a une bonne heuristique (GRASP par exemple) pour calculer des cliques maximales de grande taille, on peut l'appliquer au graphe complémentaire pour avoir des stables solutions, bornes primales, ici inférieures, du problème. C'est l'application heuristique primale. (le plsu classique)

Le regroupement par cliques sur le graphe de dépendance permet d'améliorer la qualité de la relaxation continue, donc de la borne duale du problème.

Une heuristique de recherche de cliques/stables de grande taille, permet d'améliorer les bornes inférieures et supérieures où situer l'optimum.

Plan

Problème de coloration des sommets d'un graphe



A partir d'un graphe donné, minimiser le nombre de couleurs à utiliser pour colorer tous les sommets du graphe, tq deux sommets voisins soient colorés avec des couleurs différente.

Ce minimum est le nombre chromatique du graphe G, noté $\chi(G)$.

Un problème NP-complet, très célèbre

Lien avec les stables et cliques :

- Chaque couleur définit un stable de G

- Dans une clique, tous les sommets doivent avoir une couleur différente.

$\omega(G) \leqslant \chi(G)$, la taille de clique maximum $\omega(G)$ est une borne inférieure du nombre chromatique $\chi(G)$.

Problème de coloration des arêtes

Variante : coloration des arêtes avec un nombre minimal de couleurs sachant que sachant que deux arêtes partageant un sommet ont une couleur différente.

Application : dans un réseau télécom (ex : satellite), pour éviter les interférences, des couleurs désignent des gammes de fréquences qui n'ont pas d'interférences.

En fait, ça s'écrit comme un problème de coloration des sommets du graphe $G' = (E, E')$

Le sommets du nouveau graphe sont les arêtes du graphe de départ.

Les arêtes du nouveau graphe définissent les incompatibilité entre arêtes de départ.

$e' = (e_1, e_2) \in E'$ avec $e_1, e_2 \in E$ ssi e_1 et e_2 ont un sommet en commun.

Problème de coloration des sommets, heuristiques

De nombreuses heuristiques ont été étudiées pour résoudre le problème de manière heuristique, l'état de l'art est conséquent sur ce pb très étudié.

On peut définir assez naturellement des constructions heuristiques de type gloutonnes.

Diverses méta-heuristiques classiques ont été étudiées sur ce problème, état de l'art récent et encore très actif :

Brélaz, D. (1979). New methods to color the vertices of a graph. Communications of the ACM, 22(4), 251-256.
Laguna, M., & Martí, R. (2001). A GRASP for coloring sparse graphs. Computational optimization and applications, 19(2), 165-178.
Galinier, P., Hamiez, J. P., Hao, J. K., & Porumbel, D. (2013). Recent advances in graph vertex coloring. Handbook of optimization, 505-528.
Jin, Y., Hamiez, J. P., & Hao, J. K. (2017). Algorithms for the minimum sum coloring problem : a review. Artificial Intelligence Review, 47(3), 367-394.
Mostafaie, T., Khiyabani, F. M., & Navimipour, N. J. (2020). A systematic study on meta-heuristic approaches for solving the graph coloring problem. Computers & Operations Research, 120, 104850.

Heuristique de Welsh et Powell

1. Calculer le degré de chaque sommet.
2. Trier les sommets par ordre de degrés décroissants.
3. Attribuer au premier sommet (A) de la liste une couleur.
4. Suivre la liste en attribuant la même couleur au premier sommet (B) qui ne soit pas adjacent à (A).
5. Suivre (si possible) la liste jusqu'au prochain sommet (C) qui ne soit adjacent ni à A ni à B.
6. Continuer jusqu'à ce que la liste soit finie.
7. Prendre une autre couleur pour le premier sommet (D) non encore coloré de la liste (s'il en reste) et répéter les opérations 3 à 7 tant que tous les sommets ne sont pas colorés.

Cette méthode peut aboutir à des piètres colorations. Si G est une couronne à 2n sommets, on peut avoir une coloration utilisant n couleurs, alors que le nombre chromatique est 2.

D. J. A. Welsh, M. B. Powell. An upper bound for the chromatic number of a graph and its application to 565 timetabling problems, The Computer Journal 10 (1) : pp. 85–86, (1967).

Problème de coloration, première formulation PLNE

Soit C un majorant du nombre de couleurs.($C = |V|$ convient, on a intérêt en pratique à initialiser avec une bonne valeur, par exemple obtenue par une heuristique)

On utilise des variables binaires $z_{v,c} \in \{0,1\}$, $y_c \in \{0,1\}$.
Pour $v \in V$ et $c \in [\![1; C]\!]$, $z_{v,c} = 1$ ssi le sommet v est coloré avec la couleur c.
$y_c = 1$ si la couleur c est utilisée.

En cherchant à minimiser le nombre de couleurs utilisées, on minimise l'expression linéaire $\sum_{c=1}^{C} y_c$.

On a donc un lien entre $z_{v,c}$ et y_c, $y_c = 0 \Longrightarrow z_{v,c} = 0$ pour tout v : Cela correspond a de la linéarisation de contraintes logiques vu au dernier cours :

$$\forall c \in [\![1; C]\!], \forall v \in V, z_{v,c} \leqslant y_c$$

Ensuite on ajoute la contrainte de stable, des couleurs différentes pour des voisins :

$$\forall c \in [\![1; C]\!], \forall e = (v_1, v_2) \in E, z_{v_1,c} + z_{v_2,c} \leqslant 1$$

Vertex coloring, formulation PLNE compacte

Soit C un majorant du nombre de couleurs.

On utilise des variables binaires $z_{v,c} \in \{0,1\}$, $y_c \in \{0,1\}$.

$$
\begin{array}{lll}
 & & \min \sum_{c=1}^{C} y_c \\
s.c: & \forall v \in V, & \sum_{c=1}^{C} z_{v,c} = 1 \\
 & \forall c \in [\![1;C]\!], \forall v \in V, & z_{v,c} \leqslant y_c \\
 & \forall c \in [\![1;C]\!], \forall e = (v_1, v_2) \in E, & z_{v_1,c} + z_{v_2,c} \leqslant 1 \\
 & \forall c \in [\![1;C]\!], \forall v \in V, & z_{v,c} \in \{0,1\} \\
 & \forall c \in [\![1;C]\!], & y_c \in \{0,1\}
\end{array}
\tag{14}
$$

Peut on l'améliorer avec des cliques ?

Vertex coloring, formulation PLNE compacte

Soit C un majorant du nombre de couleurs.

On utilise des variables binaires $z_{v,c} \in \{0,1\}$, $y_c \in \{0,1\}$.

$$\begin{array}{lll} & & \min \sum_{c=1}^{C} y_c \\ s.c: & \forall v \in V, & \sum_{c=1}^{C} z_{v,c} = 1 \\ & \forall c \in [\![1;C]\!], \forall e = (v_1, v_2) \in E, & z_{v_1,c} + z_{v_2,c} \leqslant y_c \\ & \forall c \in [\![1;C]\!], \forall v \in V, & z_{v,c} \in \{0,1\} \\ & \forall c \in [\![1;C]\!], & y_c \in \{0,1\} \end{array} \tag{15}$$

En effet, on a un regroupement par cliques avec $z_{v_1,c} + z_{v_2,c} \leqslant 1$, $z_{v_1,c} \leqslant y_c$ et $z_{v_2,c} \leqslant y_c$

Comme pour maxStable, on peut aussi regrouper des contraintes $z_{v_1,c} + z_{v_2,c} \leqslant y_c$ sur des cliques du graphe $G = (V, E)$

Problèmes de colorations, symétries

- Sur les formulations précédentes, de nombreuses solutions symétriques existent.
- A partir d'une solution optimale, une permutation des couleurs sans changer l'affectation en sous-ensembles reste optimale.
- Si C^{opt} est le nombre de couleur optimal, pour une solution exprimée en variables $z_{v,c}$, on a $C^{opt}! \times \binom{C}{C^{opt}} = \frac{C!}{(C - C^{opt})!}$
- Même avec l'initialisation optimale $C = C^{opt}$, il y a $C!$ solutions équivalentes
- N.B : ces symétries sont un facteur d'inefficacité de résolution PLNE, on l'expliquera

But : limiter ces symétries. Comment faire ?

Problèmes de colorations, symétries

1. On peut imposer qu'on utilise des couleurs de 1 à c_{max} et que les couleurs de c_{max+1} à C ne sont pas utilisées.

   Contraintes de monotonie dans l'utilisation des couleurs
   $y_c = 0 \Longrightarrow y_{c+1} = 0$:

   $$\forall c \in [\![1; C-1]\!], y_{c+1} \leqslant y_c$$

   Pour une solution exprimée en variables $z_{v,c}$, on a $C^{opt}!$ solutions symétriques par permutation des couleurs.

2. On peut commencer par choisir un sommet v et lui imposer sa couleur $z_{v,1} = 1$

   En fixant une seule couleur, il reste $(C^{opt} - 1)!$ permutations symétriques. Comment faire mieux ?

Problèmes de colorations des sommets, symétries

3. On peut choisir une arête $e = (v_1, v_2)$, et affecter 2 couleurs différentes à v_1, v_2, ex $z_{v_1,1} = 1$, $z_{v_2,2} = 1$. Il reste $(C^{opt} - 2)!$ permutations symétriques.

4. On cherche une grande clique dans le graphe, la plus grande qu'on puisse trouver, de cardinal K. (cf heuristiques, GRASP notamment)

   Dans une clique, toutes les couleurs sont différentes.

   On définit les couleurs de 1 à K pour tous les sommets de la clique.

   Il reste alors $(C^{opt} - K)!$ permutations symétriques, les permutations possibles pour les couleurs qui n'ont pas été fixées dans la clique

$\Longrightarrow$ un point clé de l'efficacité de la résolution PLNE pour vertex coloring, un point motivant les variantes de formulation suivantes

Formulation PLNE compacte "par représentants"

Formulation par représentants : on utilise des variables binaires $z_{v,v'} \in \{0,1\}$, pour tout $v, v' \in V$ avec $v \leqslant v'$. $z_{v,v'} = 1$ ssi v et v' sont de la même couleur, de "représentant" v, v est l'indice minimal de sa couleur.

N.B : formulation par représentants déjà utilisée pour le clustering p-median

$$
\begin{array}{lll}
\min\limits_{z} \sum\limits_{v \in V} z_{v,v} & & \\
s.c : \quad \sum\limits_{v' \leqslant v} z_{v',v} \geqslant 1 \quad (\text{ou} \quad = 1\,) & \forall v \in V & \\
z_{v,v_1} + z_{v,v_2} \leqslant z_{v,v} & \forall e = (v_1, v_2) \in E, \forall v \leqslant \min(v_1, v_2), & \\
z_{v,v'} \in \{0,1\} & \forall v \leqslant v', &
\end{array}
\tag{16}
$$

Formulation avec $\Theta(|V|^2)$ variables, mais pas de symétrie en permutant les couleurs. Formulation compacte efficace (vous pourrez l'expérimenter).

égalité ou inégalité ? A tester si cela fait une différence avec GLPK et Cplex !

Vertex coloring, formulation PLNE étendue

Soit $\mathcal{S}$ l'ensemble des stables de G. Pour tout $v \in V$, on note $\mathcal{S}_v$ l'ensemble des stables de G contenant v.

On utilise des variables binaires $z_s \in \{0,1\}$ pour $s \in \mathcal{S}$ indiquant si le stable est choisi pour définir une couleur.

$$\begin{aligned} & & \min & \sum_{s \in \mathcal{S}} z_c \\ s.c: & \quad \forall v \in V, & & \sum_{s \in \mathcal{S}_v} z_s = 1 \\ & \quad \forall s \in \mathcal{S}, & & z_s \in \{0,1\} \end{aligned} \tag{17}$$

Cette fois, on a une formulation avec un nombre potentiellement exponentiel de variables ! L'algorithme de génération de colonnes permettra de s'en sortir

Vertex coloring, autre formulation PLNE étendue

Soit $\mathcal{S}$ l'ensemble des stables maximaux de G . Pour tout $v \in V$, on note $\mathcal{S}_v$ l'ensemble des stables maximaux de G contenant v.

On utilise des variables binaires $z_s \in \{0,1\}$ pour $s \in \mathcal{S}$ indiquant si le stable est choisi pour définir une couleur.

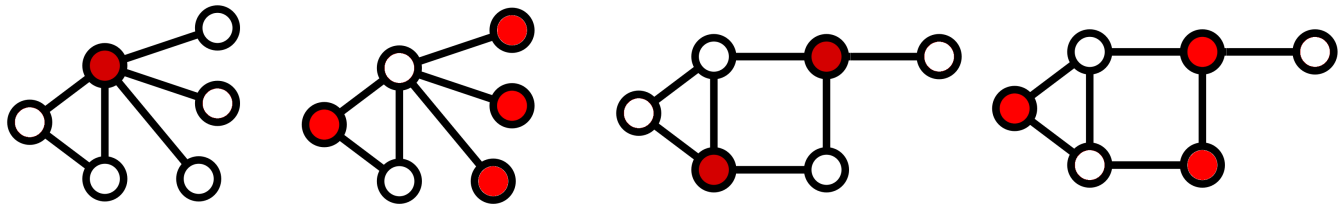
$$\begin{aligned} & \min \sum_{s \in \mathcal{S}} z_c \\ s.c: \quad & \forall v \in V, \quad \sum_{s \in \mathcal{S}_v} z_s \geqslant 1 \\ & \forall s \in \mathcal{S}, \quad z_s \in \{0,1\} \end{aligned} \tag{18}$$

Formulation mieux adaptée à la résolution de la relaxation continue par l'algorithme de génération de colonnes

Plan

Couvertures “set covering” dans un graphe non orienté

Dans un graphe $G = (V, E)$, une couverture “set covering” est un sous ensemble de sommets tel que tout sommet du graphe est relié à un sommet de la couverture, ou appartient à la couverture. ex :



Exemple trivial : V est une couverture “set covering” de $G = (V, E)$.

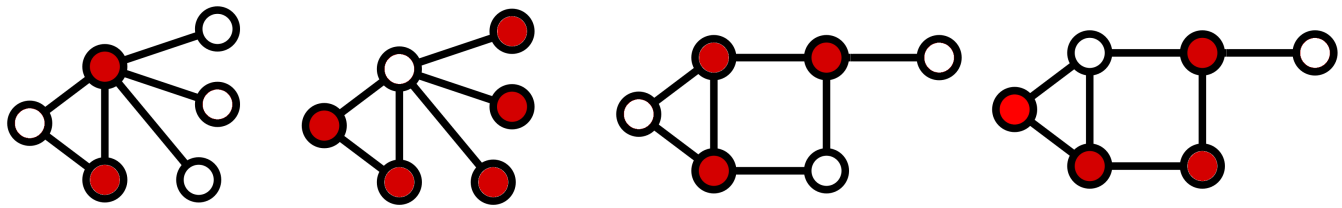
Exemple d’application (classique) : une couverture peut être un placement d’antennes en télécommunications, permettant d’atteindre tous les terminaux ? Une arête désigne alors la possibilité de capter le signal.

Autre type d’application : dans un réseau social, cerner des ”influenceurs” qui pourront diffuser une information/publicité, à un grand nombre de ”followers”.

Application sécurité informatique : où placer des des cyber-attaques pour attaquer un réseau/propager un virus ?

Couvertures “vertex covering” dans un graphe non orienté

Dans un graphe $G = (V, E)$, une couverture par les sommets,“vertex covering” est un sous ensemble de sommets tel que toute arête du graphe est contient un sommet de la couverture :



Exemple trivial : V est une couverture par les sommets de $G = (V, E)$.

Il existe aussi des couverture par les arêtes, ”edge covering”, un sous ensemble d’arêtes tel que tout sommet du graphe est présent dans les sommets des arêtes sélectionnées.

Problème de couvertures minimums ou minimales ?

Comme pour les cliques et les stables, on a des couvertures minimums ou minimales

Couverture minimum par les sommets ("MINIMUM vertex/set cover") : couverture de cardinal minimum.

Couverture minimale par les sommets ("MINIMAL vertex/set cover") : couverture qui ne contient pas de sous-ensemble couvrant, minimal au sens de l'inclusion.

Toute couverture minimum est minimale, la réciproque est fausse en général (cf illustration)

MINIMUM vertex/set covering sont des problèmes NP-complets (tous deux parmi les 21 problèmes NP-complets de Karp), MINIMUM edge covering admet une résolution polynomiale.

Richard M. Karp, *Reducibility Among Combinatorial Problems*, Complexity of Computer Computations, Plenum, 1972, p. 85-103

Minimum vertex covering, formulation PLNE

Le problème de couverture minimale s'écrit aisément en PLNE :

On utilise des variables binaires $z_v \in \{0,1\}$ pour tout $v \in V$, où $z_v = 1$ si le sommet v est considéré dans la couverture par les sommets. On a :

$$\begin{array}{lll} & & \min\limits_{z \in \{0,1\}^{|V|}} \sum\limits_{v \in V} c_v z_v \\ s.c: & \forall e = (v_1, v_2) \in E, & z_{v_1} + z_{v_2} \geqslant 1 \\ & \forall v \in V & z_v \in \{0,1\} \end{array} \tag{19}$$

On cherche à minimiser l'expression linéaire $\sum_{v \in V} z_v$, le cardinal des éléments sélectionnés, ou $\sum_{v \in V} c_v z_v$ lorsqu'une pondération des sommets a été définie.

Pour toute arête de E, au moins un de deux sommets aux extrémités est sélectionné.

Particularité des solutions optimales de relaxation continue

$$\begin{array}{lll} & & \min\limits_{z \geqslant 0} \sum\limits_{v \in V} c_v z_v \\ s.c: & \forall e = (v_1, v_2) \in E, & z_{v_1} + z_{v_2} \geqslant 1 \\ & \forall v \in V & z_v \leqslant 1 \end{array} \tag{20}$$

On peut prouver que tout sommet du polyèdre des contraintes, et donc une solution optimale de la relaxation continue obtenue par un algo de type simplexe, vérifie $z_v^* \in \{0, \frac{1}{2}, 1\}$

Preuve : on aura un élément plus générique, ici, on peut voir que partant de $z_v = 1$ et en appliquant l'algo du simplexe primal, on a toujours des solutions demi-entières. ça peut se prouver par récurrence/induction, et la propriété est vraie, à la convergence du simplexe primal, on trouve une solution demi-entière.

Qu'est ce que cela implique ?

2-Approximation pour vertex covering

Soit $\tilde{z}_v \in \{0, \frac{1}{2}, 1\}$ une solution optimale de la relaxation continue, (calculée en temps polynomial).

Soient $V_{0.5}$ et V_1 les sous-ensemble de V tels que respectivement $\tilde{z}_v = \frac{1}{2}$ et $\tilde{z}_v = 1$

$$\sum_{v \in V_{0.5}} c_v \frac{1}{2} + \sum_{v \in V_1} c_v = \sum_{v \in V} c_v \tilde{z}_v \leqslant \text{OPT}$$

Si on définit $\overline{z}_v \in \{0, 1\}$ tq $\overline{z}_v = 1$ ssi $v \in V_{0.5} \cup V_1$, $\overline{z}$ est une solution réalisable du problème, de coût $\sum_{v \in V_{0.5}} c_v + \sum_{v \in V_1} c_v$. Alors :

$$\text{OPT} \leqslant \sum_{v \in V_{0.5}} c_v + \sum_{v \in V_1} c_v \leqslant 2 \left(\sum_{v \in V_{0.5}} c_v \frac{1}{2} + \sum_{v \in V_1} c_v \right) \leqslant 2 \text{ OPT}$$

$c\overline{z}$ ainsi calculé est une 2-approximation, se calcule par PL, et la preuve de la 2-approximation utilise également la théorie de l'optimisation linéaire

Approximation polynomiale : α-approximation

Notations

- I désigne une instance d'un problème d'optimisation discrète π (minimisation dans la suite).
- $OPT(I)$ est le valeur optimale du problème π sur l'instance I.
- Soit H un algorithme de résolution de π, complexité polynomiale.
- $H(I)$ est la valeur de la solution générée par l'algo H sur l'instance I

Définition (α-approximation)

Soit H un algorithme de résolution d'un problème de minimisation π, soit $\alpha \geqslant 1$. H est une α-approximation si, pour toute instance I, on a

$$OPT(I) \leqslant H(I) \leqslant \alpha OPT(I) \tag{21}$$

Remarque : sur une instance I, le calcul de $H(I)$ donne alors une borne inférieure à l'optimum avec $\frac{1}{\alpha} H(I)$.

Borne dite "a priori", les bornes de relaxation continue sont des bornes "a posteriori", utilisant des spécificités de l'instance I.

Classes de complexité si $\mathcal{P} \neq \mathcal{NP}$

Classes de complexités pour des algorithmes suivant l'existence d'algorithmes polynomiaux avec différents types de ratios d'approximation :

- Ratios dépendant de la taille de I ($|I|$) :
  - Exp-APX (Travelling Salesman Problem)
  - Poly-APX (Graph Coloring Problem)
  - Log-APX (Set Covering Problem)

- Ratios constants, (indépendants de $|I|$), existence d'une α-approximation : APX , ex : Vertex Covering Problem

- Ratios en $1+\varepsilon$, pour tout $\varepsilon > 0$ (indépendants de $|I|$) :
  - Polynomial time approximation scheme (PTAS) (complexité polynomiale en $|I|$ mais peut être exponentielle en $1/\varepsilon$). ex : $(Pm||Cmax)$
  - Fully polynomial time approximation scheme (complexité polynomiale en $|I|$ et en $1/\varepsilon$). ex $(2||Cmax)$

Minimum set covering dans un graphe, formulation PLNE

Le problème de couverture minimale s'écrit aisément en PLNE :

- On utilise des variables binaires $z_v \in \{0,1\}$ pour tout $v \in V$, où $z_v = 1$ si le sommet v est considéré dans la couverture.
- On a :

$$\begin{aligned} & \min_{z \in \{0,1\}^{|V|}} \sum_{v \in V} z_v \\ s.c : \quad \forall v \in V, \quad & z_v + \sum_{v_2 \in \delta(v) - \{v\}} z_{v_2} \geqslant 1 \\ \forall v \in V \quad & z_v \in \{0,1\} \end{aligned} \tag{22}$$

- On cherche à minimiser l'expression linéaire $\sum_{v \in V} z_v$, le cardinal des éléments sélectionnés.
- Soit un sommet est sélectionné, soit un de ses voisins est sélectionné.

Minimum set covering, version plus robuste

Une solution optimale de Minimum set covering n'est pas forcément robuste : si une antenne est en panne, le réseau est dysfonctionnel. Pour plus de robustesse, on souhaite que chaque sommet soit couvert par au moins deux sommets colorés. Cette redondance induit un niveau de robustesse garanti. Qu'est ce que cela change pour la PLNE ?

Seule une des contraintes change. On a :

$$\begin{aligned} & \min_{z \in \{0,1\}^{|V|}} \sum_{v \in V} z_v \\ s.c: \quad & \forall v \in V, \quad z_v + \sum_{v_2 \in \delta(v) - \{v\}} z_{v_2} \geqslant 2 \\ & \forall v \in V \quad z_v \in \{0,1\} \end{aligned} \tag{23}$$

Versions couverture partielles de Minimum Set Covering

Variante : on s'autorise ici à ne pas couvrir M sommets que l'on peut choisir. (par exemple, $M \approx 0.01|V|$)

Comment adapter la formulation PLNE ?

On utilise des variables binaires $z_v \in \{0,1\}$ pour tout $v \in V$, où $z_v = 1$ si le sommet v est considéré dans la couverture.

On utilise des variables binaires $y_v \in \{0,1\}$ pour tout $v \in V$, où $y_v = 1$ si le sommet v n'est pas couvert dans la couverture partielle.

$$
\begin{aligned}
& \min_{y,z \in \{0,1\}^{|V|}} \sum_{v \in V} z_v \\
s.c: \quad \forall v \in V, \quad & z_v + \sum_{v_2 \in \delta(v) - \{v\}} z_{v_2} \geqslant 1 - y_v \\
& \sum_{v \in V} y_v \leqslant M \\
\forall v \in V \quad & z_v, y_v \in \{0,1\}
\end{aligned}
\tag{24}
$$

Versions couverture partielle robuste

Variante : on s'autorise ici à ne pas couvrir M sommets que l'on peut choisir avec un niveau de robustesse 2, tous les sommets doivent être couverts.

Comment adapter la formulation PLNE ?

On utilise des variables binaires $z_v \in \{0,1\}$ pour tout $v \in V$, où $z_v = 1$ si le sommet v est considéré dans la couverture.

On utilise des variables binaires $y_v \in \{0,1\}$ pour tout $v \in V$, où $y_v = 1$ si le sommet v n'est pas couvert dans la couverture robuste.

$$
\begin{aligned}
& \min_{y,z \in \{0,1\}^{|V|}} \sum_{v \in V} z_v \\
s.c: \quad \forall v \in V, \quad & z_v + \sum_{v_2 \in \delta(v) - \{v\}} z_{v_2} \geqslant 2 - y_v \\
& \sum_{v \in V} y_v \leqslant M \\
\forall v \in V \quad & z_v, y_v \in \{0,1\}
\end{aligned}
\tag{25}
$$

Version couverture maximale

Variante : on a ici exactement P antennes et l'on souhaite couvrir le plus de sommets du graphe

Comment adapter la formulation PLNE ?

On utilise des variables binaires $z_v \in \{0,1\}$ pour tout $v \in V$, où $z_v = 1$ si le sommet v est considéré dans la couverture.

On utilise des variables binaires $y_v \in \{0,1\}$ pour tout $v \in V$, où $y_v = 1$ si le sommet v est couvert .

$$
\begin{aligned}
& \max_{y,z \in \{0,1\}^{|V|}} \sum_{v \in V} y_v \\
s.c: \quad \forall v \in V, \quad & z_v + \sum_{v_2 \in \delta(v) - \{v\}} z_{v_2} \geqslant y_v \\
& \sum_{v \in V} z_v \leqslant P \\
\forall v \in V \quad & z_v, y_v \in \{0,1\}
\end{aligned}
\tag{26}
$$

Version couverture maximale avec influenceurs

Variante : on a un budget B pour choisir des influenceurs $I \subset V$ sur un réseau social qui partageront un contenu publicitaire. Chaque influenceur négocie son prix b_i pour faire le partage. On souhaite que la publicité soit vue par le plus grand nombre, en respectant la contrainte de budget.

On utilise des variables binaires $z_v \in \{0,1\}$ pour tout $v \in V$, où $z_v = 1$ si v est un influenceur et qu'il a été est choisi pour le partage rémunéré.

On utilise des variables binaires $y_v \in \{0,1\}$ pour tout $v \in V - I$, où $y_v = 1$ si le sommet v est touché par la publicité.

$$
\begin{aligned}
& \max_{y,z} \sum_{v \in V-I} y_v \\
s.c : \quad \forall v \in V - I, \quad & \sum_{v_2 \in \delta(v)} z_{v_2} \geqslant y_v \\
& \sum_{i \in I} b_i z_i \leqslant B \\
\forall v \in V \qquad & y_v, z_v \in \{0,1\}
\end{aligned}
\tag{27}
$$

Plan

Problèmes de flots, contexte

1845 : lois de Kirchhoff (aka loi des noeuds) expriment la conservation de l'énergie et de la charge dans un circuit électrique.

Un circuit électrique peut être vu comme un graphe, le sommets étant des composants électriques (générateur, résistance, ...), une arête étant une connexion physique.

Analogie : écoulement d'un liquide à travers un réseau de canaux, ou en plomberie.

Récemment, dans des réseaux informatiques de type internet, le débit de connexion et l'utilisation de bande passante sotn des problèmes de flots.

Le trafic routier peut aussi s'analyser avec des problèmes de flots pour optimiser les tracés et infrastructures routières.

Problème de flot maximum, définition

On considère un graphe orienté $G = (V, E)$.

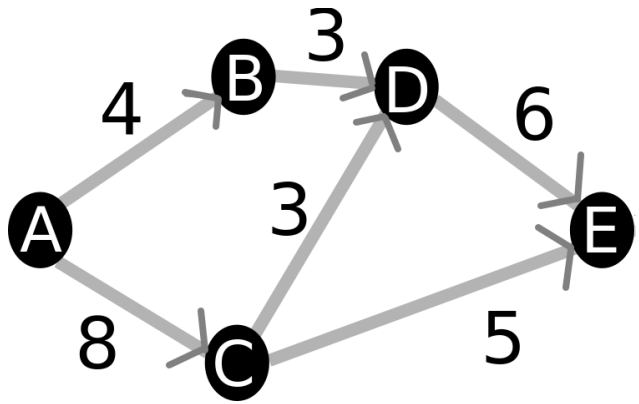
Toute arête $e \in E$ a une capacité maximale de flot, notée c_e, flot maximum traversant l'arête (orientée).

On a une source $s \in V$ et une destination $d \in V$, $s \neq d$.

Problème quel est le flot maximum total que le graphe peut supporter pour aller de s à t, et comment router le flot ?

Flots maximum, un exemple

Quel est le flot maximum entre A et E dans le graphe orienté suivant ?



On note $f_{v,v'}$ le flot de v vers v' pour toute arête (v, v'), écrite dans le format de graphe non orienté après duplications.

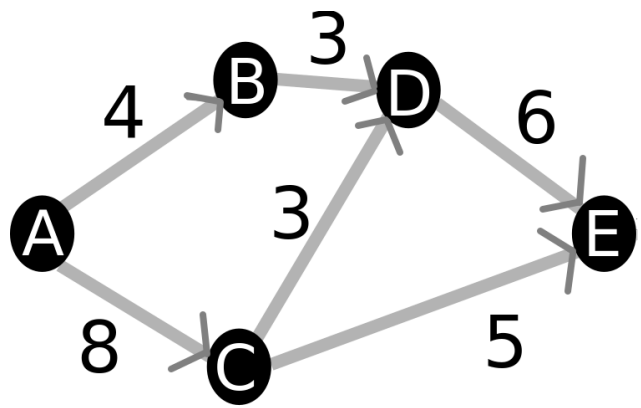
$f_{A,B} = f_{B,D} \leqslant 3$, (B, D) est limitant

$f_{A,C} = 8$ et après la séparation en C on peut avoir $f_{C,D} = 3$ et $f_{C,E} = 5$.
$f_{C,E} = 5$ est le flot maximum arrivant en E par C.

On peut avoir $f_{D,E} = 6$ grâce à après la jonction en D on a $f_{C,D} = 3$ et $f_{C,E} = 5$, c'est le flot maximum arrivant en E par D.

Cela définit bien un flot maximum de 11 entre A et E (on a eu de la chance ici sur cet exemple)

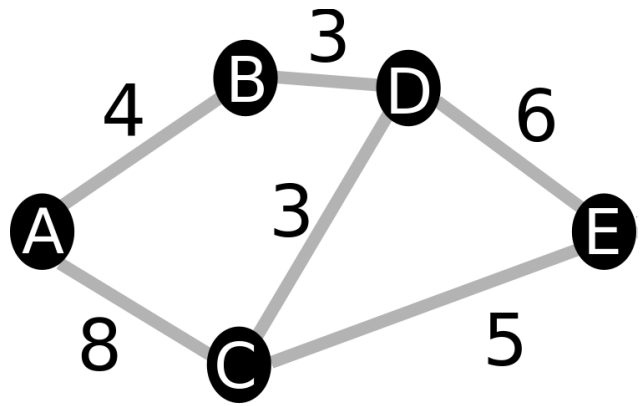
Flots maximaux, l'exemple comme un PL



On note $f_{v,v'}$ le flot de v vers v' pour toute arête (v, v'), écrite dans le format de graphe non orienté. On a le PL suivant :

$$
\begin{array}{lll}
\max & f_{A,B} + f_{A,C} & \\
s.c: & f_{A,B} + f_{A,C} = f_{C,E} + f_{D,E} & \text{(même flot en entrée et en sortie)} \\
 & f_{A,B} = f_{B,D} & \text{(loi de Kirchhoff en B)} \\
 & f_{A,C} = f_{C,D} + f_{C,E} & \text{(loi de Kirchhoff en C)} \\
 & f_{B,D} + f_{C,D} = f_{D,E} & \text{(loi de Kirchhoff en D)} \\
 & 0 \leqslant f_{A,B} \leqslant 4 & \\
 & 0 \leqslant f_{B,D} \leqslant 3 & \\
 & 0 \leqslant f_{A,C} \leqslant 8, \ldots &
\end{array}
$$

Flots maximaux, exemple non orienté comme un PL



On note $f_{v,v'}$ le flot de v vers v' pour toute arête (v, v'), écrite dans le format de graphe non orienté. On a le PL suivant :

$$
\begin{array}{lll}
\max & f_{A,B} + f_{A,C} & \\
s.c: & f_{A,B} + f_{A,C} = f_{C,E} + f_{D,E} & \text{(même flot en entrée et en sortie)} \\
 & f_{A,B} + f_{D,B} = f_{B,A} + f_{B,D} & \text{(loi de Kirchhoff en B)} \\
 & f_{A,C} + f_{D,C} + f_{E,C} = f_{C,A} + f_{C,D} + f_{C,E} & \text{(loi de Kirchhoff en C)} \\
 & f_{B,D} + f_{C,D} + f_{E,D} = f_{D,B} + f_{D,C} + f_{D,E} & \text{(loi de Kirchhoff en D)} \\
 & 0 \leqslant f_{A,B}, f_{B,A} \leqslant 4 & \text{(ou } 0 \leqslant f_{A,B} + f_{B,A} \leqslant 4 \text{ ?)} \\
 & 0 \leqslant f_{B,D}, f_{D,B} \leqslant 3 & \text{(ou } 0 \leqslant f_{B,D} + f_{D,B} \leqslant 3 \text{ ?)} \\
 & 0 \leqslant f_{A,C}, f_{C,A} \leqslant 8, \ldots &
\end{array}
$$

Flot maximum, formulation PL, cas général

Dans le cas général, on cherche le flot maximal entre s et d . On note $c_{v,v'}$ le flot maximal allant de v vers v' pour l'arête $e = (v, v')$. On a :

On note $f_{v,v'} \geqslant 0$ le flot de v vers v' pour toute arête (v, v')

$$\max_{f_{v,v'} \geqslant 0} \quad \sum_{v \in \delta_-(d)} f_{v,d}$$

$$s.c : \quad \sum_{v \in \delta_-(d)} f_{v,d} = \sum_{v \in \delta_+(s)} f_{s,v}$$

$$\sum_{v' \in \delta_-(v)} f_{v',v} = \sum_{v' \in \delta_+(v)} f_{v,v'} \quad \forall v \in V - \{s, d\},$$

$$0 \leqslant f_{v,v'} \leqslant c_{v,v'} \qquad \forall (v, v') \in E,$$

N.B : la première contrainte est en fait redondante, impliquée par les autres conservations du flot sur les sommets $v \in V - \{s, d\}$

Flot maximum entier, algorithmes polynomiaux

La formulation PL se résout en temps polynomial, donne des flots maximums continus.

Flot maximum entier : si f est une quantité discrète, ça se modéliser avec $f_{v,v'} \in \mathbb{N}$ quitte à changer l'échelle.

Résultat : si on a $c_{v,v'} \in \mathbb{N}$, le flot maximal défini par $f_{v,v'}$ est entier.

Preuve : on aura un élément plus générique, ici, on peut voir que partant de $f_{v,v'} = 0$ et en appliquant l'algo du simplexe primal, on a toujours des solutions entières. ça peut se prouver par récurrence/induction, et la propriété est vraie, à la convergence du simplexe primal, on trouve un optimum entier.

Le pb de flot maximum entier est alors polynomial, par un calcul de PL :

Algorithme "combinatoire" pour résoudre le pb de flot maximum entier sans utiliser la PL, l'algorithme de Ford-Fulkerson, avec la spécialisation d'Edmonds-Karp en temps $O(VE^2)$, des variantes plus rapides. A privilégier à la PL en pratique, des implémentations disponibles. (algos qui se voient en M1)

Dualité et flot maximum

Le pb de flot maximum (entier) comme un PL, programme primal :

$$\max_{f_{v,v'} \geqslant 0} \quad \sum_{v \in \delta_-(d)} f_{v,d}$$

$$s.c : \quad \sum_{v' \in \delta_-(v)} f_{v',v} = \sum_{v' \in \delta_+(v)} f_{v,v'} \quad \forall v \in V - \{s,d\}, \qquad (z_v)$$

$$f_{v,v'} \leqslant c_{v,v'} \qquad \forall (v,v') \in E, \quad (d_{v,v'})$$

Programme dual (même valeur par dualité forte), défini pour des variables $d_{v,v'} \geqslant 0$ pour $(v,v') \in E$ et $z_v \in \mathbb{R}$ pour $v \in V - \{s,d\}$:

$$\min_{d_{v,v'} \geqslant 0, z_v \in \mathbb{R}} \quad \sum_{(v,v') \in E} c_{v,v'} d_{v,v'}$$

$$s.c : \quad d_{u,v} - z_u + z_v \geqslant 0 \quad \forall (u,v) \in E : u \neq s, v \neq d, \quad (f_{u,v})$$

$$d_{s,v} + z_v \geqslant 0 \qquad \forall v \in \delta_+(s), \quad (f_{s,v})$$

$$d_{u,d} - z_u \geqslant 0 \qquad \forall u \in \delta_-(d), \quad (f_{u,d})$$

Dualité Coupe minimale/ flot maximal

Le problème dual . . .

$$\min_{d_{v,v'} \geqslant 0, z_v \in \mathbb{R}} \quad \sum_{(v,v') \in E} c_{v,v'} d_{v,v'}$$

$$\begin{array}{lll} s.c: & d_{u,v} - z_u + z_v \geqslant 0 & \forall (u,v) \in E : u \neq s, v \neq d, \quad (f_{u,v}) \\ & d_{s,v} + z_v \geqslant 1 & \forall v \in \delta_+(s), \quad (f_{s,v}) \\ & d_{u,d} - z_u \geqslant 0 & \forall u \in \delta_-(d), \quad (f_{u,d}) \end{array}$$

. . .est le problème de coupe minimale ("min-cut") entre s et d, les arêtes du graphe $G = (V, E)$ étant pondérées par c_e, quelles arêtes ôter dans le graphe, en minimisant la somme des c_e des arêtes choisies, pour qu'il n'existe plus de chemin allant de s à d. Cela revient à partitionner les sommets, ceux de S sont dans la composante connexe de la source, vs, ceux de D sont dans la composante connexe de la destination.

$d_{v,v'}, z_v$ sont binaires à l'optimum, $d_{v,v'} = 1$ ssi l'arête (v, v') est ôtée et $z_v = 1$ si $v \in S$

Problème de coupe minimale vs maximale

Théorème "max-flow/min-cut" : par dualité, le problème de la coupe minimale entre s et d se résout en temps polynomial.

On pourrait aussi considérer le problème "max-cut", de coupe maximale : c'est un problème NP-complet. (donc des cas spécifiques ont été prouvés polynomiaux, avec des méthodes qu'on présentera dans la suite)

Le flot minimal, c'est un flot nul, le problème n'a pas d'intérêt ...

Variante : capacités sur les sommets

Jusqu'ici, seules les arêtes avaient des capacités.

On pourrait aussi avoir des capacités maximales à chaque noeud : c_v.

$$\max_{f_{v,v'} \geqslant 0} \quad \sum_{v \in \delta_-(d)} f_{v,d}$$

$$s.c : \quad \sum_{v' \in \delta_-(v)} f_{v',v} = \sum_{v' \in \delta_+(v)} f_{v,v'} \leqslant c_v \quad \forall v \in V - \{s, d\},$$

$$f_{v,v'} \leqslant c_{v,v'} \qquad \forall (v, v') \in E,$$

Variante : capacités sur les sommets

$$\max_{f_{v,v'} \geqslant 0} \quad \sum_{v \in \delta_-(d)} f_{v,d}$$

$$s.c : \quad \sum_{v' \in \delta_-(v)} f_{v',v} = \sum_{v' \in \delta_+(v)} f_{v,v'} \leqslant c_v \quad \forall v \in V - \{s, d\},$$

$$f_{v,v'} \leqslant c_{v,v'} \quad \forall (v, v') \in E,$$

On se ramène au pb de flot maximal précédent, en transformant chaque sommet en un couple de sommets reliés par une arête, et en faisant porter la capacité du sommet sur l'arête nouvellement créée.

Transformation polynomiale au sens de la théorie de la complexité, le problème reste polynomial

Une généralisation du théorème "max-flow/min-cut" existe également.

Application : nombre maximal de chemins disjoints

On considère un graphe orienté $G = (V, E)$, un couple de source/destination s, d.

En affectant des capacités de 1 pour chaque arête, le problème max-flow calcule le nombre maximal de chemins disjoints (cad en passant par des arêtes différentes), donc en temps polynomial !

Avec la variante, on calcule le nombre maximal de chemins passant par des arêtes et des sommets différents.

Ce n'était pas trivial que ces problèmes sont polynomiaux et se calculent très bien !

Application : calcul de robustesse, l'optimum -1 du problème précédent représente le nombre maximal d'arêtes qui peuvent être retirées pour avoir toujours un chemin réalisable, "coupe minimum" apparaît ici naturellement

Extension : problèmes de multi-flots

Problèmes de multi-flots : on a un ensemble de K couples sources destinations : (s_k, d_k) pour $k \in [\![1; K]\!]$, on associe un flot F_k pour chaque $k \in [\![1; K]\!]$.

Objectif : maximiser $\sum_{k=1}^{K} F_k$

Ce problème se modélise bien avec des équations linéaires.

L'optimum en entier et en continu peut être différent, le multi-flot ebn nombres entiers est un problème continu.

L'optimum en entier et en continu peut être différent, le multi-flot maximum en nombres entiers est un problème NP-complet.

L'extension multi-coupe existe aussi.

Thèse de Lucas Létocart : *Problèmes de multicoupe et de multiflot en nombres entiers*, Paris, CNAM
`https://www-lipn.univ-paris13.fr/~letocart/These_LucasLetocart.pdf`

Multi-flots, modélisation PL(NE)

On note $f_{k,v,v'} \geqslant 0$ la part du flot k de v vers v' pour toute arête (v, v') et $k \in [\![1; K]\!]$.

$$\max_{f_{k,v,v'}, F_k \geqslant 0} \sum_{k=1}^{K} F_k$$

$$s.c : \quad \sum_{v \in \delta_+(v)} f_{k,v,v'} - \sum_{v' \in \delta_-(v)} f_{k,v',v} = \begin{cases} F_k & \text{si } v = s_k \\ -F_k & \text{si } v = d_k \\ 0 & \text{sinon.} \end{cases} \qquad \forall v \in V, k \in [\![1; K]\!]$$

$$0 \leqslant \sum_{k=1}^{K} f_{k,v,v'} \leqslant c_{v,v'} \qquad \forall (v, v') \in E,$$

$$F_k \geqslant 0, f_{k,v,v'} \in \mathbb{N} \text{ ou } \geqslant 0$$

Multi-flots avec débit commun minimum

Dans le problème précédent, à maximiser $\sum_{k=1}^{K} F_k$, on peut avoir des flots déséquilibrés.
Pour une application télécom, pour une question d'équité entre utilisateurs, on pourra chercher à avoir une solution la plus équitable, et garantir (maximiser) un flot minimal :

$$\max_{f_{k,v,v'},F_k \geqslant 0} \min_{k \in [\![1;K]\!]} F_k$$

$$s.c : \quad \sum_{v \in \delta_+(v)} f_{k,v,v'} - \sum_{v' \in \delta_-(v)} f_{k,v',v} = \begin{cases} F_k & \text{si } v = s_k \\ -F_k & \text{si } v = d_k \\ 0 & \text{sinon.} \end{cases} \qquad \forall v \in V, k \in [\![1;K]\!]$$

$$0 \leqslant \sum_{k=1}^{K} f_{k,v,v'} \leqslant c_{v,v'} \qquad \forall (v,v') \in E,$$

$$F_k \geqslant 0, f_{k,v,v'} \in \mathbb{N} \text{ ou } \geqslant 0$$

Multi-flots avec débit commun minimum, linéarisation

$$\max_{f_{k,v,v'},F_k \geqslant 0} C$$

$$s.c : \quad \sum_{v \in \delta_+(v)} f_{k,v,v'} - \sum_{v' \in \delta_-(v)} f_{k,v',v} = \begin{cases} F_k & \text{si } v = s_k \\ -F_k & \text{si } v = d_k \\ 0 & \text{sinon.} \end{cases} \qquad \forall v \in V, k \in [\![1;K]\!]$$

$$0 \leqslant \sum_{k=1}^{K} f_{k,v,v'} \leqslant c_{v,v'} \qquad \forall (v,v') \in E,$$

$$C \leqslant F_k \qquad \forall k \in [\![1;K]\!],$$

$$F_k \geqslant 0, f_{k,v,v'} \in \mathbb{N} \text{ ou } \geqslant 0$$

Variante télécom avec débit commun minimum

L'opérateur télécom garantit un débit minimum (ex 3G), défini à F_{min}. Et pour la qualité de service, essaie d'optimiser le nombre d'utilisateur ayant un excellent débit F_{QoS}.

Variables binaires supplémentaires : $x_k^{QoS} \in \{0,1\}$, avec $x_k^{QoS} = 1$ si l'utilisateur k a son débit au moins à F_{QoS}.

$$\max_{f_{k,v,v'},F_k \geqslant 0} \sum_{k=1}^{K} x_k^{QoS}$$

$$s.c : \quad \sum_{v \in \delta_+(v)} f_{k,v,v'} - \sum_{v' \in \delta_-(v)} f_{k,v',v} = \begin{cases} F_k & \text{si } v = s_k \\ -F_k & \text{si } v = d_k \\ 0 & \text{sinon.} \end{cases} \qquad \forall v \in V, k \in [\![1;K]\!]$$

$$0 \leqslant \sum_{k=1}^{K} f_{k,v,v'} \leqslant c_{v,v'} \qquad \forall (v,v') \in E,$$

$$F_{min} \leqslant F_k \qquad \forall k \in [\![1;K]\!],$$

$$F_k \geqslant F^{QoS} x_k^{QoS} \qquad \forall k \in [\![1;K]\!],$$

$$F_k \geqslant 0, f_{k,v,v'} \in \mathbb{N} \text{ ou } \geqslant 0$$

Plan

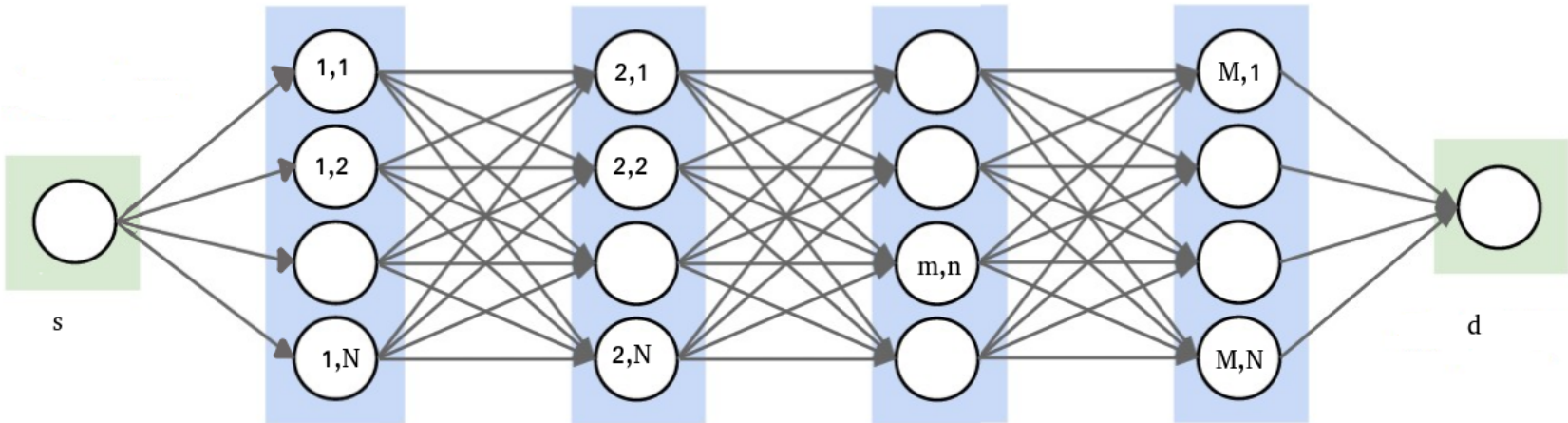
D'un problème de plus court chemin à un problème de flot

On peut voir le problème d'existence d'un chemin entre une source s et une destination t dans un graphe orienté graphe originel $G = (V, E)$ comme un problème de flot entier : existe-t'il un flot de valeur 1 entre s et t, en pondérant chaque arête d'une capacité de 1.

La formulation PLNE de problèmes de plus court chemin peut s'appuyer sur la formulation PLNE des flots.

Problème de plus court chemin spécifique

Structure spécifique de graphe orienté sans cycle à M couches, chaque couche ayant N noeuds :

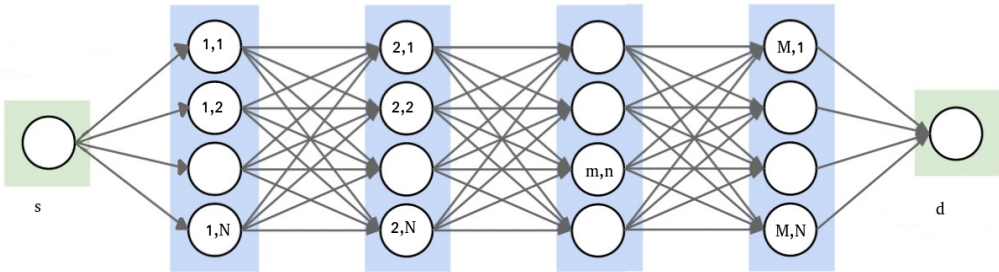


On note :

- $d_{m,n,n'}$ la distance entre le point m, n et le point $m+1, n'$ pour $m < M$
- a_n la distance entre l'origine et le point $1, n$
- b_n la distance entre le point M, n et la destination finale

On veut minimiser la distance d'un chemin entre la source s et la destination d

Formulation PLNE



- $d_{m,n,n'}$ la distance entre le sommet m, n et le sommet $m+1, n'$ pour $m < M$
- a_n la distance entre l'origine et le sommet $1, n$
- b_n la distance entre le sommet M, n et la destination finale

Définition des variables de la PLNE :

- $x_{m,n,n'} \in \{0,1\}$ indique (=1) ssi l'arc entre le sommet m, n et le sommet $m+1, n$ pour $m < M$ est utilisé
- $f_n \in \{0,1\}$ indique (=1) ssi la source et le sommet $(1, n)$ sont reliés.
- $l_n \in \{0,1\}$ indique (=1) ssi la destination et le sommet (M, n) sont reliés.

Fonction objectif à minimiser :

$$\sum_{n=1}^{N}(a_n f_n + b_n l_n) + \sum_{n=1}^{N}\sum_{n'=1}^{N}\sum_{m=1}^{M} d_{m,n,n'} x_{m,n,n'} \tag{28}$$

- $x_{m,n,n'} \in \{0,1\}$ indique (=1) ssi l'arc entre le sommet m,n et le sommet $m+1,n$ pour $m < M$ est utilisé
- $f_n \in \{0,1\}$ indique (=1) ssi la source et le sommet $(1,n)$ sont reliés.
- $l_n \in \{0,1\}$ indique (=1) ssi la destination et le sommet (M,n) sont reliés.

$$\min_{x,f,l \in \{0,1\}} \sum_{n=1}^{N} (a_n f_n + b_n l_n) + \sum_{n=1}^{N} \sum_{n'=1}^{N} \sum_{m=1}^{M} d_{m,n,n'} x_{m,n,n'}$$

$$\forall n \in [\![1; N]\!] \qquad \sum_{n=1}^{N} f_n = \sum_{n=1}^{N} l_n = 1$$

$$\forall n \in [\![1; N]\!], \qquad \sum_{n'=1}^{N} x_{1,n,n'} = f_n$$

$$\forall n \in [\![1; N]\!], \qquad \sum_{n'=1}^{N} x_{M,n',n} = l_n$$

$$\forall n \in [\![1; N]\!], \forall m \in [\![1; M-1]\!] \qquad \sum_{n'=1}^{N} x_{m,n',n} = \sum_{n'=1}^{N} x_{m+1,n,n'}$$

Ajout de contraintes de ressources ?

- On considère un ensemble $\mathcal{R}$ de ressources en quantités limitées, la ressource $r \in \mathcal{R}$ étant limitée à une quantité Q_r
- On associe à chaque arc la consommation d'une ressource $q_{r,m,n,n'}, q^f_{r,n}, q^l_{r,n} \geqslant 0$
- ex : minimiser le temps pour aller à la pompe à essence ou recharge de véhicule électrique et ne pas tomber en panne sur le chemin.
- On peut imaginer avoir plusieurs ressources (ex : avec des prix de péage)

$\Longrightarrow$ Comment adapter la formulation PLNE ?

Formulation PLNE, avec contraintes de ressource

- $x_{m,n,n'} \in \{0,1\}$ indique (=1) ssi l'arc entre le sommet m,n et le sommet $m+1,n$ pour $m < M$ est utilisé
- $f_n \in \{0,1\}$ indique (=1) ssi la source et le sommet $(1,n)$ sont reliés.
- $l_n \in \{0,1\}$ indique (=1) ssi la destination et le sommet (M,n) sont reliés.

$$\min_{x,f,l \in \{0,1\}} \sum_{n=1}^{N} (a_n f_n + b_n l_n) + \sum_{n=1}^{N} \sum_{n'=1}^{N} \sum_{m=1}^{M} d_{m,n,n'} x_{m,n,n'}$$

$$\forall n \in [\![1; N]\!] \qquad \sum_{n=1}^{N} f_n = \sum_{n=1}^{N} l_n = 1$$

$$\forall n \in [\![1; N]\!], \qquad \sum_{n'=1}^{N} x_{1,n,n'} = f_n$$

$$\forall n \in [\![1; N]\!], \qquad \sum_{n'=1}^{N} x_{M,n',n} = l_n$$

$$\forall n \in [\![1; N]\!], \forall m \in [\![1; M-1]\!] \qquad \sum_{n'=1}^{N} x_{m,n',n} = \sum_{n'=1}^{N} x_{m+1,n,n'}$$

$$\forall r \in \mathcal{R} \qquad \sum_{n=1}^{N} (q^f_{r,n} f_n + q^l_{r,n} l_n) + \sum_{n=1}^{N} \sum_{n'=1}^{N} \sum_{m=1}^{M} q_{r,m,n,n'} x_{m,n,n'} \leqslant Q_r$$

Plus court chemin élémentaire, formulation PLNE

Soit $G = (V, E)$ un graphe orienté, on note $d_e \in \mathbb{R}$ la "distance" pour parcourir e (pas forcément positive).

Problème : Quel est le plus court chemin entre deux sommets s et d, en en passant au plus par un seul sommet ? (cf algos prog.dynamique de Dijkstra et Floyd-Warshall, pb avec des cycle absorbants)

Soit $z_{v,v'} \in \{0, 1\}$ pour tout $(v, v') \in E$, où $z_{v,v'} = 1$ si l'arête (v, v') est utilisée dans le chemin de s à d dans l'ordre v et puis v'.

On cherche à minimiser $\sum_{e \in E} d_e z_e$ (expression linéaire)

s et d sont associés à exactement un sommet, tout autre sommet vérifie une loi de Kirchoff (= conservation du flux) :

$$\sum_{v \in \delta_+(s)} z_{s,v} = \sum_{v \in \delta_-(d)} z_{v,d} = 1$$

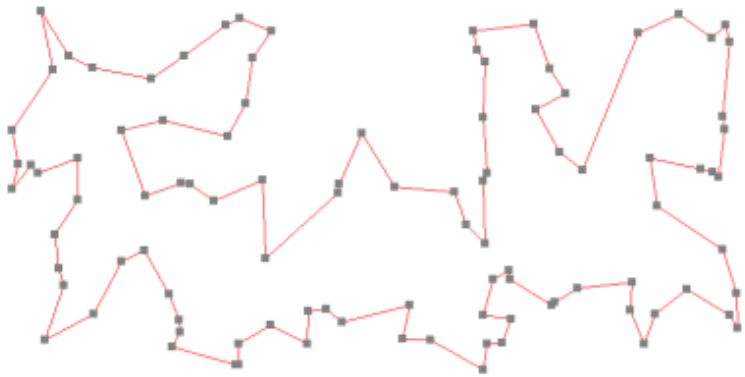
$$\forall v \in V - \{s, d\}, \sum_{v' \in \delta_-(v)} z_{v',v} = \sum_{v'' \in \delta_+(v)} z_{v,v''} \leqslant 1$$

N.B : PLNE non réalisable en cas d'absence de chemin de s à d.

Plan

Problèmes de voyageur de commerce, notations

- Un voyageur de commerce doit transiter par N villes, $\mathcal{N} = [\![1; N]\!]$.
- Tous les trajets sont possibles, la distance entre les villes $n \in \mathcal{N}$ et $n' \in \mathcal{N}$ est $d_{n,n'} = d_{n',n}$ (sinon, $d_{n,n'} = +\infty$ ou un majorant).
- Voyageur de commerce symétrique (TSP) : $d_{n,n'} = d_{n',n}$, graphe non orienté.
- Voyageur de commerce asymétrique (ATSP) : on n'a pas forcément $d_{n,n'} = d_{n',n}$, structure de graphe orienté.
- Minimiser la longueur du trajet du représentant de commerce.



$\Longrightarrow$ Un problème d'optimisation fortement NP-complet

Des cycles hamiltoniens au pb de voyageur de commerce

On considère un graphe orienté $G = (V, E)$

Un cycle Hamiltonien de G est un cycle qui passe par tous les sommets une fois et une seule.

L'existence d'un cycle Hamiltonien définit un pb NP-complet.

La recherche d'un cycle Hamiltonien orienté se ramène au problème de voyageur de commerce asymétrique dans un graphe complet : on considère un majorant strict de la distance des arêtes, $D > d_e$ pour tout $e \in E$. $|V|D$ est un majorant strict de la distance de tout cycle Hamiltonien. En affectant $d_e = |V|D$ pour tout $e \notin E$, l'existence d'un cycle Hamiltonien dans le graphe originel est équivalent à ce que l'optimum du TSP soit inférieur à $|V|D$.

Dasn un graphe non orienté, le problème de problème de voyageur de commerce est alors symétrique avec la même transformation.

Le TSP, un problème fondamental de l'optimisation combinatoire

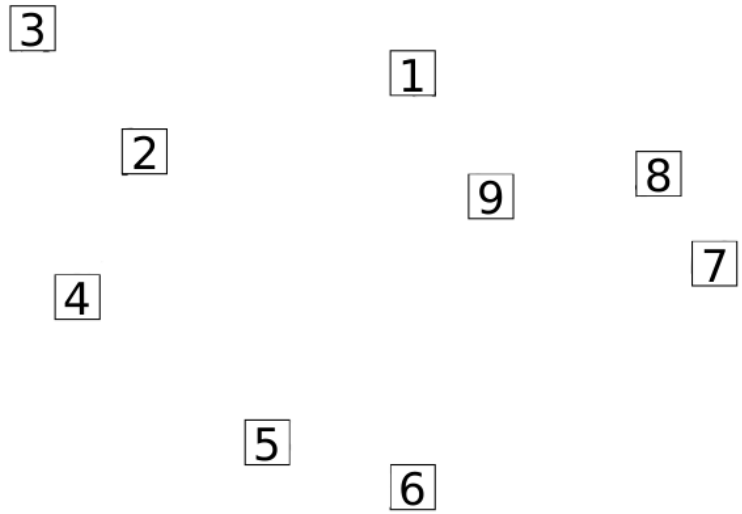
Bien sûr le TSP est un problème fondamental en RO pour ses extensions s'appliquant à des problèmes de logistique et de livraisons. (cf section suivante) extensions transport.

Plus fondamental, tant pour les heuristiques que la modélisation PLNE, c'est une base pour décrire la structure combinatoire de permutation, en modélisation PLNE et en opérateurs de méta-heuristiques :

- optimiser l'ordre des voitures à monter dans une chaîne d'assemblage (options et peinture). pb académiques de job-shop et de flow-shop
- un classement agrégé (de données biologiques par ex) se modélise comme une permutation, indiquant l'ordre des préférences de 1 à N.

Brancotte, B., Yang, B., Blin, G., Cohen-Boulakia, S., Denise, A., & Hamel, S. (2015). *Rank aggregation with ties : Experiments and analysis*. Proceedings of the VLDB Endowment (PVLDB), 8(11), 1202-1213.

Modélisation PLNE du pb de voyageur de commerce



On note :

- $\mathcal{N}$ l'ensemble des villes et $d_{n,n'}$ la distance entre les villes n et n'.
- Variables binaires $x_{n,n'} \in \{0,1\}$ pour $(n,n') \in \mathcal{N}^2$ avec $n' \neq n$.
- $x_{n,n'} = 1$ ssi le voyageur de commerce va dans la ville n' immédiatement après avoir visité la ville n.
- Objectif à minimiser $\sum_{n,n'} d_{n,n'} x_{n,n'}$.

Écriture des contraintes de flots (conditions nécessaires)

- A chaque ville n, il existe un unique successeur :

$$\forall n \in \mathcal{N}, \quad \sum_{n' \neq n} x_{n,n'} = 1$$

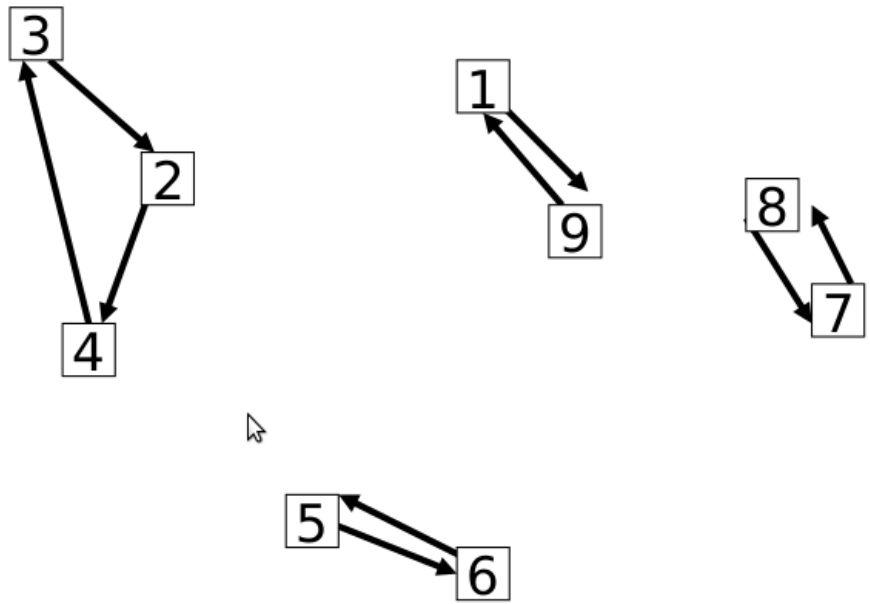
- A chaque ville b, il existe un unique prédécesseur :

$$\forall n \in \mathcal{N}, \quad \sum_{n' \neq n} x_{n',n} = 1$$

Est-ce suffisant ?

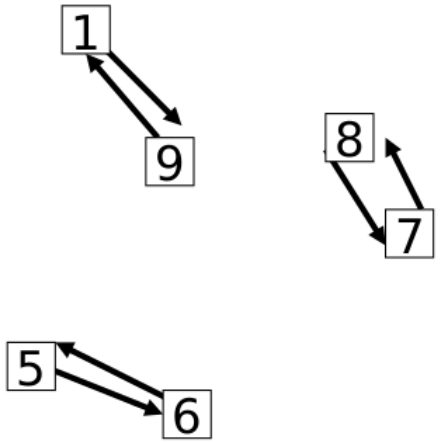
Insuffisance de la formulation précédente

Configuration réalisable (et intéressante en terme de coût) :



Comment rajouter les contraintes manquantes ?

Coupes de sous tour entre deux villes



$$x_{1,9} + x_{9,1} \leqslant 1$$

$$x_{7,8} + x_{8,7} \leqslant 1$$

$$x_{5,6} + x_{6,5} \leqslant 1$$

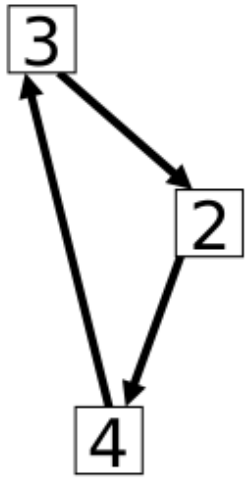
De manière générale :

$$\forall n \in \mathcal{N}, \forall n' > n, \quad x_{n,n'} + x_{n',n} \leqslant 1 \tag{29}$$

Variante équivalente :

$$\forall n \in \mathcal{N}, \forall n' > n, \quad x_{n,n'} \leqslant \sum_{n'' \neq n'} x_{n'',n} \leqslant 1 \tag{30}$$

Coupes de sous tour entre trois villes



$$x_{2,3} + x_{3,2} + x_{2,4} + x_{4,2} + x_{4,3} + x_{3,4} \leqslant 2$$

De manière générale :

$\forall n \in \mathcal{N}, \forall n'' > n' > n,$

$$x_{n,n'} + x_{n',n} + x_{n,n''} + x_{n'',n} + x_{n',n''} + x_{n'',n'} \leqslant 2 \qquad (31)$$

Coupes de sous-tour, de manière générale

Soit $\mathcal{S} \subsetneq \mathcal{N}$, la contrainte suivante coupe les sous tours entre les villes de $\mathcal{S}$ (sous ensemble strict des villes) :

$$\sum_{n \neq n' \in \mathcal{S}} x_{n,n'} + x_{n',n} \leqslant |\mathcal{S}| - 1 \tag{32}$$

Variante équivalente : il doit y avoir une arête partant d'un sommet de $\mathcal{N} - \mathcal{S}$ et arrivant dans $\mathcal{S}$:

$$\sum_{n \in \mathcal{S}} \sum_{n' \in \mathcal{N} - \mathcal{S}} x_{n,n'} \geqslant 1 \tag{33}$$

on se retrouve avec un nombre exponentiel de contraintes (2^n).

Formulation TSP de Dantzig, Fulkerson et Johnson

$$
\begin{aligned}
\min_{x} \quad & \sum_{n' \neq n} d_{n,n'} x_{n,n'} \\
\forall n \in \mathcal{N}, \quad & \sum_{n' \neq n} x_{n,n'} = 1 \\
\forall n \in \mathcal{N}, \quad & \sum_{n' \neq n} x_{n',n} = 1 \\
\forall \mathcal{S} \subset \mathcal{N}, \quad & \sum_{n \neq n' \in \mathcal{S}} x_{n,n'} + x_{n',n} \leqslant |\mathcal{S}| - 1 \\
\forall n' \neq n \quad & x_{n,n'} \in \{0,1\},
\end{aligned}
\tag{34}
$$

Nombre exponentiel de contraintes, on verra comment on peut utiliser de telles formulations

Plusieurs variantes avec un nombre exponentiel de contraintes existent

Formulations “compactes” du (A)TSP

Formulation compacte : formulation avec un nombre raisonnable, polynomial de contraintes, pour être utilisé directement par les solveurs.

Pour le (A)TSP, les formulations compactes présentent en général des relaxations continues moins bonnes que les formulations avec un nombre exponentiel de variables ou de contraintes.

Intérêt des formulations compactes : pour les extensions du TSP, des briques de base en modélisation. Pas toujours facile d'étendre les méthodes de résolution avec un nombre exponentiel de contraintes.

On présente dans la suite quelques unes de ces modélisations. On pourra se référer à l'article suivant pour une présentation plus exhaustive :

T. Öncan, I. Altınel, G. Laporte, *A comparative analysis of several asymmetric traveling salesman problem formulations* Computers & OR, vol 36, n 3, p637–654, 2009.

Formulations “compactes” du (A)TSP

On garde le modèle de base :

$$\begin{aligned} \min_{x} \quad & \sum_{n' \neq n} d_{n,n'} x_{n,n'} \\ \forall n \in \mathcal{N}, \quad & \sum_{n' \neq n} x_{n,n'} = 1 \\ \forall n \in \mathcal{N}, \quad & \sum_{n' \neq n} x_{n',n} = 1 \\ \forall n' \neq n \quad & x_{n,n'} \in \{0,1\}, \end{aligned} \tag{35}$$

et on cherche à avoir une description compacte des contraintes de sous-tour

N.B : cela nécessite de rajouter des variables, pas de description compacte avec uniquement les variables x

Formulation de Miller-Tucker-Zemlin (MTZ)

- On ajoute les variables $v_n \in [\![0, N-1]\!]$, pour tout $n \in \mathcal{N}$, indiquant l'ordre de passage,
- on choisit arbitrairement d'une ville origine $v_1 = 0$,
- on linéarise les contraintes $x_{n,n'} = 1 \Longrightarrow v_{n'} = v_n + 1$:
- Avec l'aide d'une valeur M "grande", ($M = N$ convient), il suffit d'ajouter les contraintes :

$$\forall n \neq n' \in \mathcal{N}, \quad v_{n'} \geqslant v_n + 1 - M(1 - x_{n,n'})$$

On peut se restreindre à un ajout de variables v_n continues et positives avec les contraintes $v_n \leqslant N - 1$ pour tout n.

Les contraintes de sous tour ne sont alors plus nécessaires pour avoir une solution réalisable.

On a ajouté N variables entières et N^2 contraintes additionnelles. (mais la relation continue est de piètre qualité)

Formulation de Gouveia et Pires (GP)

On considère les variables binaires additionnelles $y_{n,n'} \in \{0,1\}$ pour $n \neq n'$avec $y_{n,n'} = 1$ ssi la ville n' est visitée après (mais pas forcément immédiatement après) la ville n, en numérotant à partir de la ville 1.

Les contraintes à ajouter sont alors pour tous $j \neq k \neq l$:

$$x_{l,j} + y_{k,l} - y_{k,j} \leqslant 1 \tag{36}$$

$$x_{l,j} - y_{l,j} \leqslant 0 \tag{37}$$

$$x_{l,j} + y_{j,l} \leqslant 1 \tag{38}$$

Les contraintes (37) sont juste les implications qu'être immédiatement avant implique d'être avant $y_{l,j} = 0 \Longrightarrow x_{l,j} = 0$.

Les contraintes (38) sont l'incompatibilité entre l juste avant j et j avant l.

Les contraintes (36) peuvent se vérifier avec une table de vérité, cela exprime que si l est juste avant j , ie $x_{l,j} = 1$, une autre ville k est avant j et l ou après j et l. Si $x_{l,j} = 1$, $y_{k,l} = 1 \Longrightarrow y_{k,j} = 1$

$\Longrightarrow$ Ajout de $O(N^2)$ variables entières et $O(N^3)$ contraintes additionnelles.

Renforcement de la Formulation de Gouveia et Pires (GP)

À partir de l'équation $x_{l,j} + y_{k,l} - y_{k,j} \leqslant 1$, plusieurs contraintes plus fortes ont été proposées (issues de techniques de renforcement) :

$$x_{l,j} + x_{j,l} + y_{k,l} - y_{k,j} \leqslant 1 \tag{39}$$

$$x_{l,j} + x_{k,j} + x_{l,k} + y_{k,l} - y_{k,j} \leqslant 1 \tag{40}$$

Preuves par distinction de cas sur les x.

Les contraintes (39) : que l'on ait l juste avant j ou j juste avant l, ie $x_{l,j} + x_{j,l} = 1$, une autre ville k est avant j et l ou après j et l. (plus fort que précédemment)

Les contraintes (40) améliorent mieux en pratique la relaxation continue que (39), mais ne sont pas forcément plus fortes. On peut considérer ces deux jeux de contraintes pour encore améliorer légèrement la relaxation continue par rapport à (40) utilisé tout seul.

Formulation de Sarin, Sherali and Bhootra (SSB)

Mêmes variables binaires additionnelles que GP : $y_{n,n'} \in \{0,1\}$ pour $n \neq n'$avec $y_{n,n'} = 1$ ssi la ville n' est visitée après (mais pas forcément immédiatement après) la ville n, en numérotant à partir de la ville 1.

Les contraintes à ajouter sont alors pour tous $j \neq k \neq l$:

$$y_{l,j} + y_{j,l} = 1 \tag{41}$$

$$x_{l,j} \leqslant y_{l,j} \tag{42}$$

$$y_{l,j} + y_{j,k} + y_{k,l} \leqslant 2 \tag{43}$$

(43) est symétrique et peut se voir : si l avant j et j avant k, ie si $y_{l,j} = 1$ et $y_{j,k} = 1$ alors l est avant k, ie $y_{l,k} = 1$ et $y_{k,l} = 0$

$\Longrightarrow$ Ajout de $O(N^2)$ variables entières et $O(N^3)$ contraintes additionnelles.

Renforcement de la formulation SSB

À partir de l'équation $y_{l,j} + y_{j,k} + y_{k,l} \leqslant 2$, plusieurs contraintes plus fortes existent :

$$y_{l,j} + x_{j,l} + y_{j,k} + y_{k,l} \leqslant 2 \tag{44}$$

$$y_{l,j} + y_{j,k} + y_{k,l} + \frac{1}{3}(x_{j,l} + x_{k,j} + x_{l,k}) \leqslant 2 \tag{45}$$

Formulation de Fox, Gavish et Graves (FFG)

Variables $z_{n,n',i} \in \{0,1\}$ pour $n \neq n'$ et $i \in [\![1; N]\!]$ avec $z_{n,n',i} = 1$ ssi la ville n' est visitée juste après la ville n, aux positions i et $i+1$ après être parti de la ville 1 avec $i = 1$. On a $z_{n,n',1} = 0$ et $z_{n',n,N} = 0$ pour $n \neq 1$ et tout $n' \neq n$

$$\min_{z_{n,n',i} \in \{0,1\}} \quad \sum_{i=1}^{N} \sum_{n' \neq n} d_{n,n'} z_{n,n',i} \tag{46}$$

$$\forall n \in \mathcal{N}, \quad \sum_{i=1}^{N} \sum_{n' \neq n} x_{n,n',i} = 1 \tag{47}$$

$$\forall n' \in \mathcal{N}, \quad \sum_{i=1}^{N} \sum_{n \neq n'} x_{n,n',i} = 1 \tag{48}$$

$$\forall i \in [\![1; N]\!], \quad \sum_{n' \neq n} x_{n,n',i} = 1 \tag{49}$$

$$\forall n \in [\![2; N]\!], \quad \sum_{i=2}^{N} \sum_{n' \neq n} i \times x_{n,n',i} - \sum_{i=1}^{N} \sum_{n' \neq n} i \times x_{n',n,i} = 1 \tag{50}$$

$$\forall n' \neq n \quad z_{n,n',i} \in \{0,1\}, \tag{51}$$

Formulation FFG, remarques

FFG : Formulation basée sur des variables binaires différentes, qui vont permettre une modélisation de type flot.

Les contraintes :

$$\forall n \in [\![2; N]\!], \sum_{i=2}^{N} \sum_{n' \neq n} i \times x_{n,n',i} - \sum_{i=1}^{N} \sum_{n' \neq n} i \times x_{n',n,i} = 1 \tag{52}$$

peuvent être vues comme le regroupement (par agrégation) de contraintes de flots, les contraintes suivantes sont plus fortes :

$$\forall n \in [\![2; N]\!], \forall i \in [\![2; N]\!], \sum_{i=2}^{N} \sum_{n' \neq n} x_{n,n',i} = \sum_{i=1}^{N} \sum_{n' \neq n} x_{n',n,i-1} \tag{53}$$

La formulation FFG avait $O(N^3)$ variables et $O(N)$ contraintes, avec (53), cela ferait $O(N^3)$ variables et $O(N^2)$ contraintes, pour un faible gain en pratique.

TSP et heuristiques

Avec un espace des solutions réalisables de taille $N!$, les heuristiques permettent d'aller vers des tailles inacessibles aux méthodes exactes.

Le TSP, un cas d'étude intéressant pour les heuristiques où l'ensemble des solutions est défini selon une structure de permutation, c'est dans cet esprit que nous allons introduire la recherche locale.

Recherche locale : le maillon manquant aux heuristiques GRASP.

Recherche locale, voisinages et extremums locaux : une adaptation discrète des algorithmes de gradients de l'optimisation continue.

Comme précédemment, des heuristiques gloutonnes sont naturelles.

Ici, un résultat d'approximation pour des TSP avec inégalité triangulaire, hypothèse réaliste pour la pratique.

Algorithme glouton adaptatif pour le TSP

- La ville 1 est le point de départ.
- On choisit pour ville suivante, la ville la plus proche de 1, disons v_2.
- Pour choisir la troisième ville, on choisit la ville la plus proche de v_2 autre que $1 = v_1$
- On itère ...
- Une fois les villes $v_1, \ldots, v_k$ choisies, la ville suivante v_{k+1} est la ville la plus proche de v_k autre que $v_1, \ldots, v_{k-1}$

$\Longrightarrow$ Heuristique en temps $O(N^2)$

$\Longrightarrow$ Adaptation gloutonne randomisée, on peut choisir aléatoirement parmi els villes les plus proches (par exemple parmi les 3 villes les plus proches)

TSP : Algorithme d'approximation de Christofides

- Cas du TSP avec distances respectant l'inégalité triangulaire (hypothèse réaliste)
- A partir du graphe complet formé par les villes, avec comme poids les sommets, on peut calculer l'arbre couvrant de poids minimal (algo Prim ou Kruskal, en temps $O(N^2)$)
- On répare l'arbre couvrant en cycle Hamiltonien, opération en $O(N)$, ça donne même une 2-approximation.
- Amélioration dans l'algorithme de Christofides : réparation avec un algorithme de couplage parfait (perfect matching), donne même une 3/2-approximation, ie au plus 50% de sur-coût par rapport à l'optimum dans le pire cas (bien mieux en pratique).

`http://www.cs.cornell.edu/courses/cs681/2007fa/Handouts/christofides.pdf`

`https://pypi.org/project/Christofides/`

`https://github.com/sth144/christofides-algorithm-cpp`

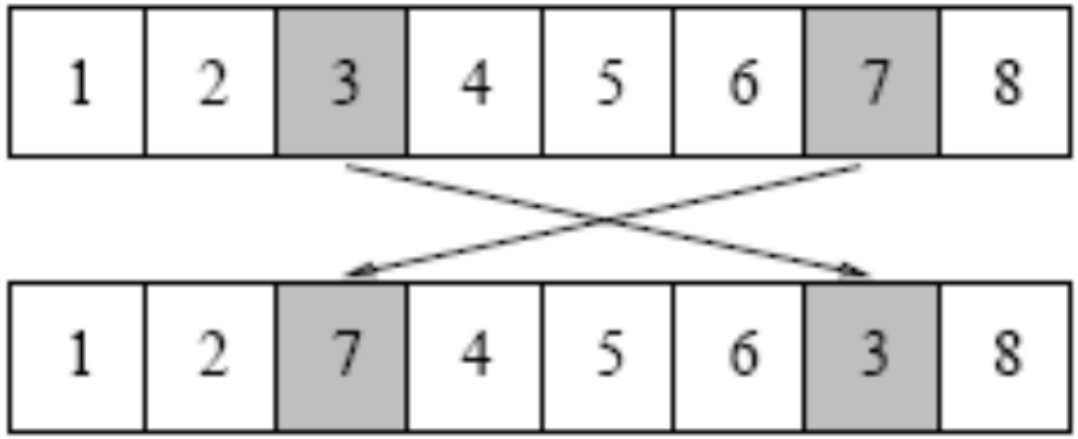
(Méta-)Heuristiques perturbatives de trajectoire

- Heuristique perturbative/recherche locale : on passe d'une solution à une autre par une petite modification locale, pour espérer au cours des itérations (avec un critère d'arrêt défini), avoir parcouru (ou convergé) vers une bonne solution
- Premier point : comment définir une modification locale ?
- On part d'une solution réalisable : utilise heuristique constructive en amont.
- critère d'arrêt : temps total défini, nombre d'itérations, ...
- trajectoire : par opposition aux populations (ex : algos génétiques,n évolutionnaires, colonies de fournis/d'abeilles, essaims particulaires), on n'a qu'une solution courante en mémoire

Voisinages, minimums locaux/globaux

- L'encodage définit un espace X de toutes les solutions réalisables.
- Un voisinage est une fonction $\mathcal{N}$ qui associe un sous-ensemble de X à toute solution $x \in X$. Une solution $x_0 \in \mathcal{N}(x)$ est dite voisine de x.
- Les voisinages dépendent du problème : point laissé générique lors de la définition d'une méta-heuristique, à spécifier.
- Pour un problème donné, de multiples voisinages peuvent être définis en général.

Exemple TSP : voisinage d'échange



voisinage d'échange : $\mathcal{N}(x)$ est l'ensemble des permutations atteignables à partir d'une permutation x de $[\![1; N]\!]$ en conservant au $N-2$ positions de villes définies par x

$|\mathcal{N}(x)| = N(N-1)/2$

Généralisation k-échanges (aussi noté k-opt) : on conserve au moins $N-k$ positions pour définir $\mathcal{N}_k^{swap}(x)$

$$|\mathcal{N}_k^{swap}(x)| = \frac{N!}{k!(N-k)!} = \Theta(N^k)$$

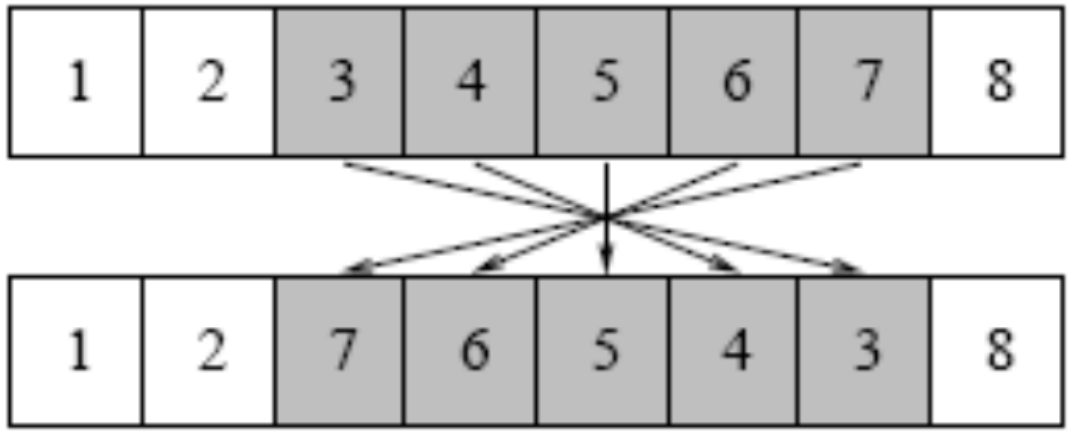
Exemple TSP : voisinage d'insertion

voisinage d'insertion : $\mathcal{N}(x)$ est l'ensemble des permutations atteignables à partir d'une permutation x de $[\![1; N]\!]$ en conservant $N-1$ ordres entre villes définies par x

$|\mathcal{N}(x)| = O(N^2)$

On peut étendre aussi à un ordre supérieur

Exemple TSP : voisinage d'inversion



voisinage d'inversion : $\mathcal{N}(x)$ est l'ensemble des permutations atteignables à partir d'une permutation x de $[\![1; N]\!]$ en appliquant un miroir à un sous ensemble $[\![a; b]\!]$ avec $a < b$ à choisir arbitrairement.

$|\mathcal{N}(x)| = N(N-1)/2$

Voisinages et minimums locaux

- L'encodage définit un espace X de toutes les solutions réalisables, pour le TSP, ça se définit par une permutation, un tableau de N éléments distincts de $[\![1; N]\!]$.
- Soit f la fonction objectif de l'optimisation est définie pour $x \in X$.
- Une solution $x \in X$ est un minimum local relativement à la structure de voisinage $\mathcal{N}$ si $f(x) \leqslant f(x_0)$ pour tout $x_0 \in \mathcal{N}(x)$.
- $x \in X$ est un minimum global si $f(x) \leqslant f(x_0), \forall x_0 \in X$
- Un minimum global est minimum local quelque soit le voisinage $\mathcal{N}$ considéré.
- Réciproquement, il existe en général des minimums locaux non globaux.

$\Longrightarrow$ Extension des notions d'optimisation continue, d'optimums locaux ou globaux

Rappel : Minimum local/global en optimisation continue

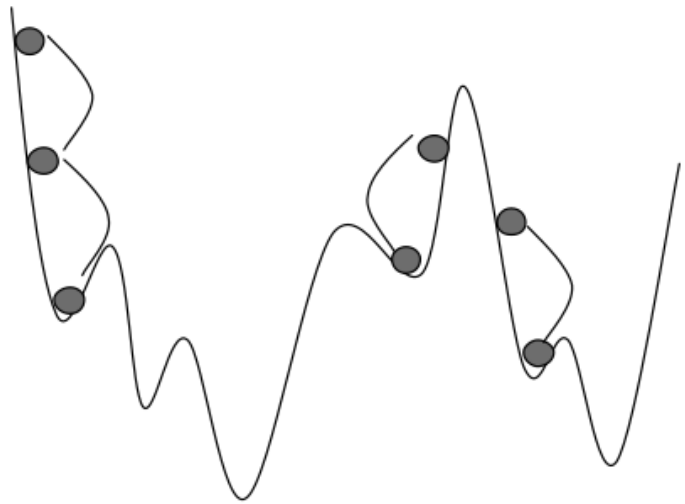
Soit $f : A \to \mathbb{R}$.

x_0 est un minimum local si

$\exists \varepsilon > 0 \; \forall x \in]x_0 - \varepsilon, x_0 + \varepsilon[\cap A, \; f(x_0) \leqslant f(x$

x_0 est un minimum global si

$$\forall x \in A, \; f(x_0) \leqslant f(x)$$



Si $f : A \to \mathbb{R}$ est dérivable, et si un point intérieur $x \in \mathring{A}$ est un extremum local, cela implique $f'(x_0) = 0$. (la réciproque est fausse)

Et généralisation $f : A \to \mathbb{R}^n$, avec distance/norme de $\mathbb{R}^n$, f' désignant gradient.

$\Longrightarrow$ Quelle différence majeure sur les propriétés de voisinages continu/discret ?

Différence entre voisinages continu/discret

Les voisinages discrets sont uniquement définis, la localité est une notion sur un $x \in]x_0 - \varepsilon, x_0 + \varepsilon[$ avec $\varepsilon > 0$ petit, aussi petit soit il.

Cette notion (et donc la dérivée, les gradients) ne peut être retranscrite en discret.

Dérivée discrète ? OK, l'algo de gradient sera étendu ...

Différence majeure : les extremas locaux sont définis de manière unique en continu, alors qu'en discret, un extremun local est défini relativement à un voisinage.

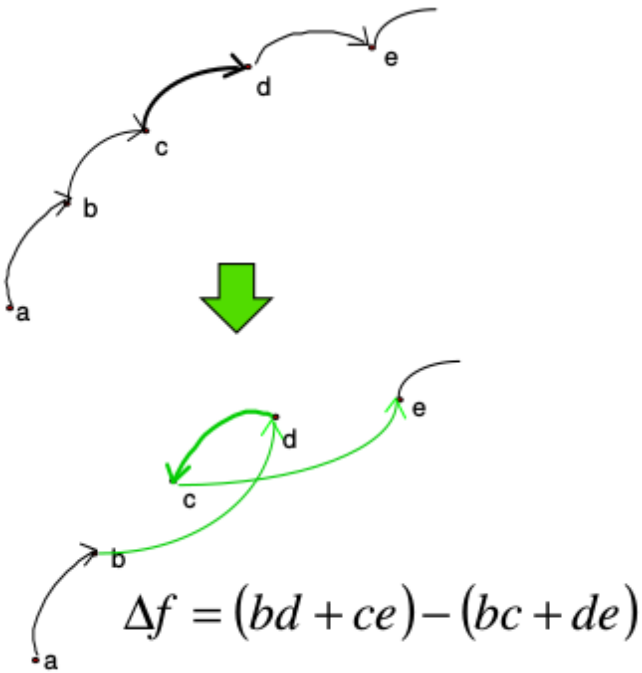
Un minimum local pour un voisinage discret peut ne plus être minimum local en considérant un autre voisinage discret.

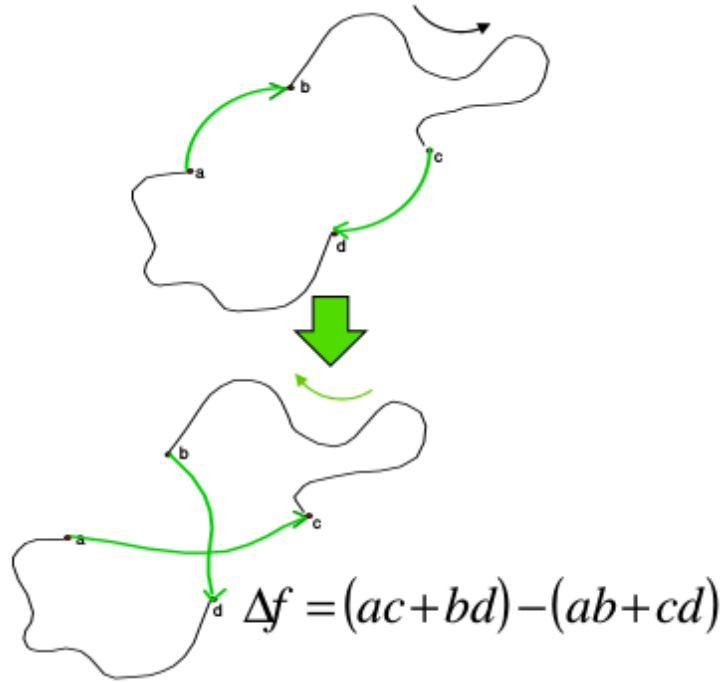
Si $\mathcal{N}_1(x) \subset \mathcal{N}_2(x)$ pour tout $x \in X$, un minimum local pour $\mathcal{N}_2$ l'est aussi pour $\mathcal{N}_1$ (preuve triviale)

Prouver qu'une solution est minimum local

- Soit X un espace d'encodage, f la fonction objectif définie sur X et $\mathcal{N}$ un voisinage.
- Question : comment prouver que $x \in X$ est minimum local
- Vérifier : $f(x) \leqslant f(x_0)$ pour tout $x_0 \in \mathcal{N}(x)$.
- sur les exemples, $\mathcal{N}(x)$ de taille polynomiale, calcul $f(x_0)$ polynomial ...
- On peut mieux faire, pour aller plus vite !

Point clé : calcul du coût de solution dans un voisinage





Calcul rapide du coût de solution par modification :

- A partir d'une solution x, on connaît $f(x)$.
- On calcule $f(x_0)$ à partir de $f(x)$ en considérant uniquement les modifications
- Sur les exemples précédents, $f(x_0)$ se calcule en $O(1)$ à partir de $f(x)$

Exemple TSP : voisinage d'insertion

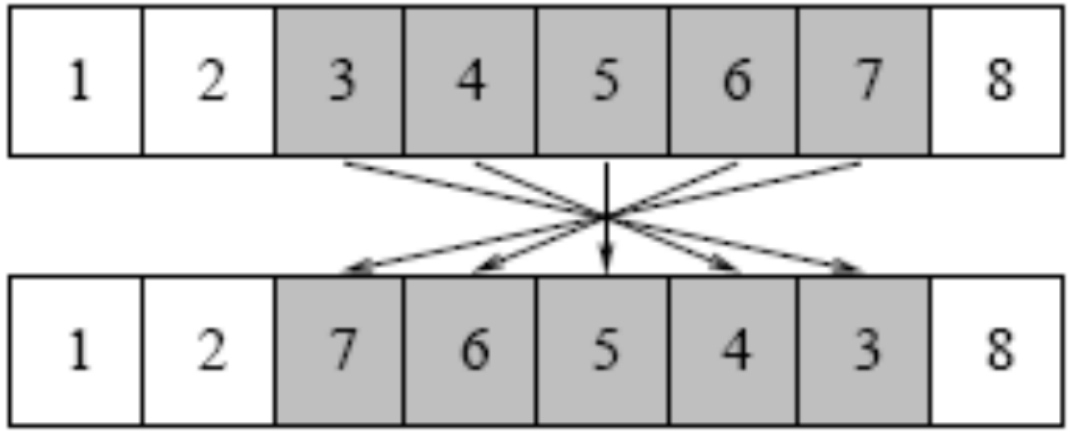
$f(x_0) = f(x) - d_{5,6} - d_{6,7} + d_{1,6} + d_{6,2}$

$|\mathcal{N}(x)| = N^2$

Prouver que x est un minimum local est en $O(N^2)$

Calcul naïf était en en $O(N^3)$, cela a son importance en pratique !

Exemple TSP : voisinage d'inversion



$f(x_0) = f(x) - d_{2,3} - d_{7,8} + d_{2,7} + d_{3,8}$

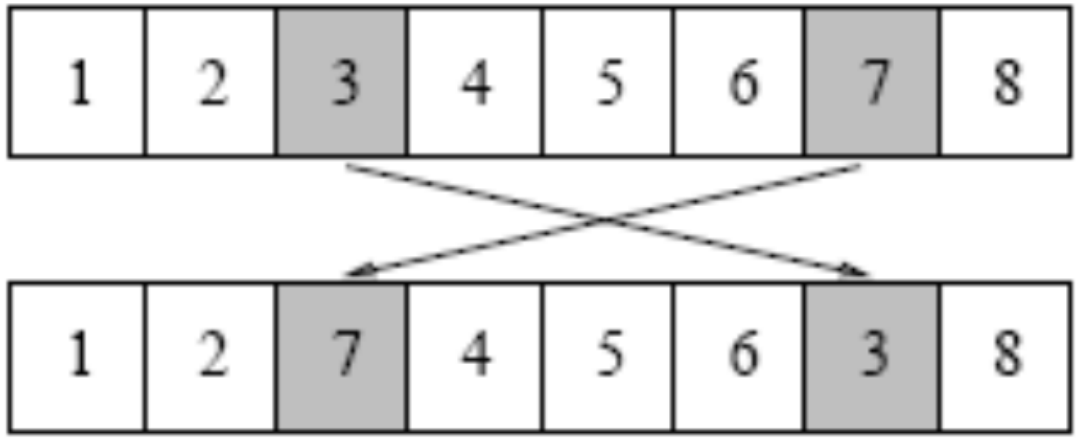
$|\mathcal{N}(x)| = O(N^2)$

Prouver que x est un minimum local est en $O(N^2)$

Calcul naïf était en en $O(N^3)$, cela a son importance en pratique !

N.B : ici la formule est vraie uniquement pour le TSP symétrique, la complexité du calcul local est moins bonne pour le TSP asymétrique.

Exemple TSP : voisinage d'échange



$f(x_0) = f(x) - d_{2,3} - d_{3,4} - d_{6,7} - d_{7,8} + d_{2,7} + d_{7,4} + d_{6,3} + d_{3,8}$

$|\mathcal{N}(x)| = O(N^2)$

Prouver que x est un minimum local est en $O(N^2)$

Calcul naïf était en en $O(N^3)$, cela a son importance en pratique !

Hill climbing : une première recherche locale

- Paramètres : solution initiale $x_i \in X$, on se donne un voisinage $\mathcal{N}$
- Initialisation : $x = x_i$, $f = f(x)$
- TANT QUE (critère d'arrêt temps ou nombre d'itération atteint)
- On parcourt tout le voisinage $\mathcal{N}(x)$ en calculant les $f(x_0)$
- Existe t'il une solution $x_0 \in X$ telle que $f(x_0) < f(x)$?
- Si NON, on sort de la boucle TANT QUE
- Si OUI, on actualise $x = \text{argmin}_{x_0 \in \mathcal{N}(x)} f(x_0)$ et $f = f(x)$.
- FIN TANT QUE
- On renvoie $x, f(x)$

Hill climbing : une recherche locale de descente

- Si le critère d'arrêt de fin d'itération n'est pas atteint, on renvoie un minimum local.
- Chaque itération est améliorante strictement, sinon, on s'arrête.
- Hill Climbing suit la direction de plus grande descente, c'est l'analogue de l'algorithme de gradient.
- Algorithme de recherche locale le plus simple.
- Variantes pour économiser du temps à chaque itération, on peut arrêter l'exploration du voisinage :
  - à la première solution améliorante trouvée ;
  - quand un nombre d'améliorations a été trouvé, et prendre la meilleure solution améliorante ;
  - critère d'arrêt avec amélioration trouvée lié à la complexité. ex : voisinages TSP en $O(N^2)$, on peut arrêter l'exploration au bout de $O(N)$ évaluations si on a trouvé une solution améliorante.

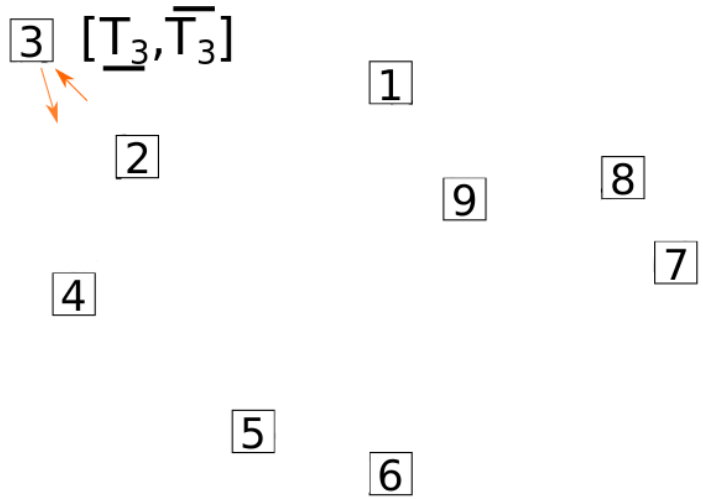
Sortir d'un minimum local

- En pratique, il existe souvent de multiples minimums locaux, de qualité très variable.
- Hill Climbing peut converger rapidement dans un minimum local de mauvaise qualité, le minimum local trouvé dépend de la solution initiale.
- Problématique : explorer différents minimums locaux diversifiés en temps imparti.
- intensification/exploration : deux concepts pour bien paramétrer/équilibrer une recherche locale
- De nombreuses méta-heuristiques pour offrir des solutions à ces probléamtiques

$\Longrightarrow$ sensibilisation à la recherche locale, pour poursuivre, des cours de méta-heuristiques

Plan

Problème de voyageur de commerce avec fenêtres de temps



TSP with Time Windows

Le voyageur de commerce peut passer par une ville $n > 1$ dans une certaine fenêtre de temps imposée $[\underline{T}_n, \bar{T}_n]$, en comptant que le départ à la ville $n = 1$ était à $T_1 = 0$. Quelle est la distance minimale à parcourir pour traverser les villes exactement une fois et revenir à la ville de départ $n = 1$ en respectant les contraintes de fenêtres de temps ? N.B : en cas d'arrivée avant la fenêtre de temps, il y a attente jusqu'au temps $\underline{T}_n$.

Notation additionnelle : $T_{n,n'} \geqslant 0$: temps de parcours de la ville n à la ville n'

Exemple : Optimisation du parcours du postier selon des heures de présence.

Modélisation PLNE du problème TSP-TW (1)

On garde les modélisations compactes précédentes, avec les variables $x_{n,n'} = 1$ ssi le voyageur de commerce va dans la ville n' immédiatement après avoir visité la ville n.

Variables (continues) additionnelles : $t_n \geqslant 0$ l'heure de passage à la ville n avec $T_1 = 0$. On a bien sûr les contraintes de fenêtre de temps :

$$\forall n > 1, \quad \underline{T}_n \leqslant t_n \leqslant \bar{T}_n$$

.

Contraintes de "transitions" : $x_{n,n'} = 1 \Longrightarrow t_{n'} \geqslant t_n + T_{n,n'}$. (il peut y avoir un temps d'attente). Linéarisation standard en posant M un majorant du temps de parcours, par exemple avec $M = \max_n(\bar{T}_n + T_{n,1})$:

$$\forall n' > 1, \forall n, \quad t_{n'} \geqslant t_n + T_{n,n'} - M(1 - x_{n,n'})$$

N.B : on pourrait aussi avoir une contrainte sur le temps de retour à $n = 1$, disons T^{retour} :

$$\forall n' > 1, \quad t_n + T_{n,1} \leqslant T^{retour}$$

Modélisation PLNE du problème TSP-TW (2)

En pratique, les formluations avec "big M" sont souvent très faibles, la relaxation continue peut arriver à ignorer ces contraintes.

Variables (continues) additionnelles : $t_{n,n'} \geqslant 0$ l'heure de passage à la ville $n > 1$ avec $T_1 = 0$, sachant que la ville suivante est n', sinon $t_{n,n'} = 0$.

Par rapport à la modélisation précédente $t_n = \sum_{n'} t_{n,n'}$, un seul $t_{n,n'}$ est non nul, sa valeur est t_n. Les contraintes de fenêtre de temps peuvent s'écrire :

$$\forall n > 1, \quad \underline{T}_n x_{n,n'} \leqslant t_{n,n'} \leqslant \bar{T}_n x_{n,n'}$$

.

Les contraintes de transitions peuvent alors s'écrire sans "bigM" :

$$\forall n' > 1, \forall n, \quad \sum_{n''} t_{n',n''} \geqslant t_{n,n'} + T_{n,n'} x_{n,n'}$$

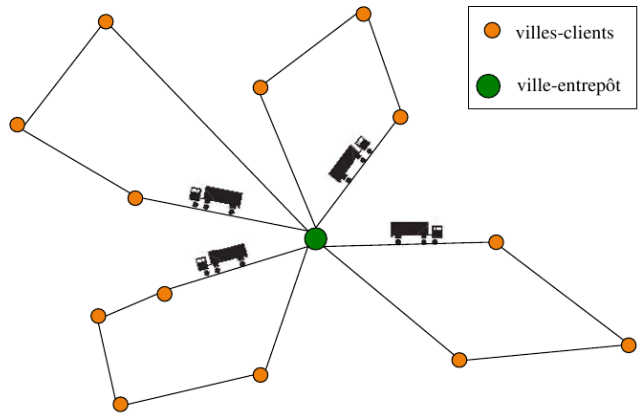
On a la contrainte voulue si $x_{n,n'} = 1$, le cas $x_{n,n'} = 0$ implique des inégalités $0 \leqslant 0$.

Problèmes de tournées de véhicules

Vehicle Routing Problem (VRP) : problème de tournées de véhicules.

On a N livraisons indicées $n \in [\![1, N]\!]$.

On a V véhicules identiques pour réaliser les livraisons, en partant et en revenant à un dépôt $n = 0$



on note $\mathcal{N} = [\![0, N]\!]$ et $\mathcal{V} = [\![1, V]\!]$

On veut minimiser la longueur totale des trajets en utilisant plusieurs véhicules.

CVRP : avec des contraintes de capacité, il n'est pas réalisable qu'un seul véhicule desserve toutes les livraisons : chaque livraison n nécessite une capacité (par exemple volume) limitée au total à C, chaque livraison consommant une capacité c_n dans le budget C.

CVRP, modélisation PLNE

Variables $x_{v,n,n'} = 1$ ssi le véhicule v va au noeud $n' \in \mathcal{N}$ immédiatement après avoir été au noeud $n \in \mathcal{N}$.

On peut avoir $x_{v,0,0} = 1$ si le véhicule ne sort pas. Pour tout $n > 0$, on a $x_{v,n,n} = 0$

$$\min_x \quad \sum_{v\in\mathcal{V}} \sum_{n\in\mathcal{N}} \sum_{n'\in\mathcal{N}} d_{n,n'} x_{v,n,n'}$$

$$\forall n > 0, \quad \sum_{v\in\mathcal{V}} \sum_{n'\in\mathcal{N}} x_{v,n,n'} = 1$$

$$\forall v \in \mathcal{V}, \forall n > 0, \quad \sum_{n'\in\mathcal{N}} x_{v,n,n'} = \sum_{n'\in\mathcal{N}} x_{v,n',n}$$

$$\forall v \in \mathcal{V}, \quad \text{contraintes pour interdire les sous-tours du véhicule } v$$

$$\forall v \in \mathcal{V}, \quad \sum_{n'\neq n} c_n x_{v,n',n} \leqslant C$$

$$\forall v \in \mathcal{V}, \forall n, n' \in \mathcal{N} \quad x_{v,n,n'} \in \{0, 1\},$$

Variantes de problèmes de tournées de véhicules

Il existe de multiples extensions des CVRP :

- VRPTW : VRP avec fenêtres de temps
- MDVRP : VRP des dépôts différents
- HFVRP : VRP avec des véhicules hétérogènes (influe sur coût, vitesse de déplacement, capacité)
- VRP avec des véhicules électriques (contraintes de rechargement)
- Green/Polluting VRP : considérations développement durable
- "pick-up and delivery" VRP
- "last mile delivery" VRP : par exemple avec des drones pour finaliser les livraisons, plusieurs drones par camion.

Et de nombreuses autres contraintes inspirés de problèmes réels, la flexibilité de la modélisation PLNE le permet !

CONCLUSIONS et PERSPECTIVES

Bilan

On a vu comment modéliser de nombreux problèmes classiques d'optimisation dans les graphes et des variantes en PLNE :

- Cliques et stables maximums
- Colorations
- Couvertures
- Problèmes de flots, coupes, multi-flots
- Plus court chemins
- Voyageur de commerce et extensions

Cela a illustré les techniques de modélisation/linéarisation PLNE

On a vu des heuristiques sur quelques uns de ces problèmes, pour introduire des grandes idées de constructions gloutonnes et de recherche locale

Perspectives naturelles

Vous devriez être convaincus à présents de la diversité des problèmes d'optimisation qui se modélisent en PLNE, un dernier TD sur des applications d'ordonnancement ou de planification.

Une question incontournable à ce stade :

Comment ça se résout de manière générale un PLNE ? ?

Quels algorithmes de résolution ?

Comment s'utilisent les solveurs PLNE libres et commerciaux ?

Questions entrouvertes, poursuivies par ce cours

Renforcement de formulation, est ce que des techniques générales existent ? Comment les coupes de cliques sont elles utilisées ?

L'analyse polyédrale, les formulations plus fortes, est ce que cela a un fort impact en pratique ?

Comment résoudre un PLNE avec un nb exponentiel de contraintes ou de variables ?

Les symétries, qu'est ce que ça change en PLNE ?

On n'avait pas promis un lien avec des méthodes de recherche arborescente en IA ?

Perspectives entrouvertes, à poursuivre dans d'autres cours

La programmation linéaire et la dualité sont aussi des outils pour l'informatique théorique. La suite au MPRI !

Algorithmes d'approximation, l'ouvrage de référence (en accès libre), peut vous convaincre de l'intérêt des bases solides en PL fournies par ce cours :

Williamson, D. P., & Shmoys, D. B. (2011). The design of approximation algorithms. Cambridge university press.

`https://www.designofapproxalgs.com/book.pdf`

Les méta-heuristiques, on a eu une vision partielle. Rdv dans plusieurs cours du M1 informatique. Un ouvrage de référence pour approfondir :

Talbi, El-Ghazali. Metaheuristics : from design to implementation. Vol. 74. John Wiley & Sons, 2009.

Des slides sont en ligne `http://www.lifl.fr/~talbi/metaheuristic/`