CHPS0742
Optimisation

Cours 4

Problèmes NP-Difficiles
Résolution par méthodes exactes

Pierre Delisle
Université de Reims Champagne-Ardenne
Département de Mathématiques et Informatique
Septembre 2018

Plan de la séance

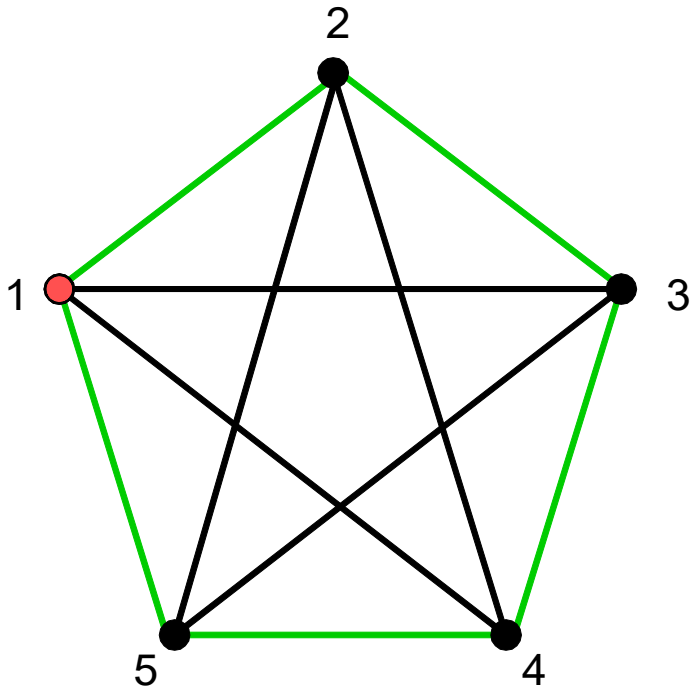
- P, NP, problèmes NP-complets, NP-difficiles, …
  - Problème du sac à dos
  - Problème du Voyageur de Commerce
- Algorithmes de résolution exacts
  - Méthodes par séparation et évaluation
  - ...

Le problème du Voyageur de Commerce

Problèmes NP-Complets, NP-Difficiles

Préalable : cycle hamiltonien

- Cycle qui passe une fois et une seule par chaque sommet
- La longueur d'un cycle est la somme des longueurs des arêtes le constituant



Le cycle (1, 2, 3, 4, 5, 1) est un cycle hamiltonien

Le problème de Voyageur de Commerce (VC)

- En anglais : Travelling Salesman Problem (TSP)
- Étant donné un graphe complet valué …
- … à *n* sommets …
- … on cherche un cycle hamiltonien de longueur minimum de ce graphe

Un algorithme trivial

- Énumérer tous les cycles hamiltoniens du graphe, calculer leur longueur et retenir celui qui minimise la fonction longueur
- Un graphe complet à *n* sommets
  - (n – 1) ! / 2 cycles hamiltoniens
- Impossible à mettre en oeuvre, même lorsque l'ordre du graphe est petit
  - *n* = 20 → un ordinateur traitant 1 million de cycles hamiltoniens par seconde → 19 siècles de calcul !!!
- On ne connaît pas d'algorithme fondamentalement meilleur pour résoudre ce problème de façon exacte

Problème de reconnaissance

- Problème d'optimisation du Voyageur de Commerce (OVC)
  - Étant donné un graphe *G* d'ordre *n*, complet et valué par une fonction à valeurs entières strictement positives
  - Déterminer la longueur minimum d'un cycle hamiltonien dans *G*
- Problème de reconnaissance (ou de décision) du VC (RVC)
  - Étant donné … à valeurs entières strictement positives
  - Existe-t-il dans le graphe *G* un cycle hamiltonien de longueur inférieure ou égale à *k* ?
- Ces 2 problèmes ne diffèrent que par un facteur constant
  - Algorithme polynomial qui résoud l'un résoud l'autre
  - On s'intéresse aux problèmes de reconnaissance pour évaluer la difficulté des problèmes d'optimisation associés

Classes P et NP

- Classe P
  - Un problème est dit polynomial s'il existe un algorithme de complexité polynomiale permettant de répondre à la question posée dans ce problème, quelle que soit la donnée de celui-ci
  - La classe P est l'ensemble de tous les problèmes de reconnaissance polynomiaux
  - Tous les problèmes vus jusqu'à maintenant
- Classe NP
  - Un problème de reconnaissance est dans la classe NP si on peut vérifier en temps polynomial qu'une solution proposée permet d'affirmer que la réponse est «oui» pour cette instance
  - RVC est dans la classe NP → on peut vérifier qu'une solution est un cycle hamiltonien et que sa longueur est <= *k*
- P = NP ?

Problèmes NP-Complets

- Un problème est *NP-Complet* s'il est dans la classe NP et si on peut le ramener, par une transformation polynomiale, à un autre problème de la classe NP
- Le problème du Voyageur de Commerce est un problème NP-Complet
  - L'existence d'un algorithme polynomial pour résoudre un tel problème entraînerait l'existence d'algorithmes polynomiaux pour résoudre n'importe quel problème de NP (P = NP)
- Démontrer qu'un problème est NP-Complet permet d'éviter de perdre du temps à recherche un algorithme polynomial pour le résoudre
  - On se tourne alors vers d'autres méthodes

Problèmes NP-Difficiles

- On ne s'intéresse plus seulement aux problèmes de reconnaissance, mais aussi aux problèmes d'optimisation
- Un problème NP-difficile est un problème au moins aussi difficile qu'un problème NP-complet
  - Si le problème de décision associé à un problème d'optimisation est NP-complet, alors le problème d'optimisation est NP-difficile
- Par conséquent, le problème d'optimisation du Voyageur de Commerce est NP-Difficile

Algorithmes de résolution

- Si on veut absolument utiliser une méthode exacte pour résoudre ces problèmes de façon optimale, il existe des algorithmes qui accélèrent la résolution …
  - Méthodes par séparation et évaluation
  - Programmation dynamique
- On se tournera toutefois plus souvent vers des méthodes approchées
  - Algorithmes qui, pour les problèmes d'optimisation, donneront une solution que l'on espérera aussi proche que possible de l'optimum en un temps « raisonnable »
  - Heuristiques, métaheuristiques, ...

Méthodes par séparation et évaluation

Complexité de la programmation linéaire ? Du simplexe ?

- Problème de programmation linéaire (nb réels)
  - Polynomial
- Algorithme du simplexe (Dantzig)
  - Exponentiel ! → O(exp *n*)
  - Problème à 3*n* variables et 2*n* contraintes → $O(2^n)$
- Algorithme de Kachian (ellipsoïde)
  - $O(n^6)$
- Algorithme de Karmarkar (point intérieur)
  - $O(n^{3,5})$

Problème de programmation linéaire en nombres entiers

- Problème de programmation linéaire standard avec des *contraintes d'intégrité*
  - On impose aux variables d'être entières
- Si n est le nombre de variables :

Maximiser	$c\,x$
Sous les	$Ax \leq b$
Contraintes	$x \geq 0$
	$x \in \mathbb{N}^n$

Complexité

- Programmation linéaire en nombre réels
  - Polynomial
- Programmation linéaire en nombres entiers
  - NP-complet
  - On cherchera donc généralement à le résoudre de façon approchée

Problème du sac à dos

- En anglais : *knapsack problem*
- On veut constituer le contenu d'un sac à dos
  - Avec n objets de volumes v_1, … v_n
  - La somme des volumes des n objets est supérieure au volume V du sac à dos
  - Il faut choisir quels objets seront conservés, donc on affecte une utilité u_1, … u_n à chaque objet
- Applications pratiques
  - Gestion de portefeuille, chargement de bateaux/avions, allocation de ressources, ...

Modélisation du problème

- Introduction de *n* variables de décision
  - $x_1, \ldots x_n$ (x_i représente l'objet *i*)
  - À valeurs dans {0, 1} (1 si on prend l'objet, 0 sinon)

Maximiser $\sum_{j=1}^{n} u_j x_j$

Sous les Contraintes $\sum_{j=1}^{n} v_j x_j \leq V$

$x_j \in \{0,1\}$ pour $1 \leq j \leq n$

- Problème de programmation linéaire en variables bivalentes (aussi appelé « en 0-1 ») à une seule contrainte

- Sac à dos généralisé : variables entières positives ou nulles, 1 seule contrainte

Résolution heuristique 1

- Relâchement des contraintes d'intégrité
  - On résout en variables réelles
  - Puis on arrondit les solutions obtenues en prenant leurs parties entières par défaut
- Heuristique
  - Donne une solution approchée (donc pas nécessairement optimale)
  - En temps polynomial
- Exemple

Résolution heuristique 2

- Approche qualité/prix
- On classe les variables en ordre décroissant des rapports utilité/volume
- On donne ensuite à chaque variable la plus grande valeur possible compte tenu des choix précédents
  - Valeur entière positive ou nulle pour le sac à dos généralisé
- Algorithme « glouton » (on traite les variables une à une sans remettre en cause les décisions)
- Exemple

Méthode par séparation et évaluation (branch and bound)

- Résolution de problèmes d'optimisation combinatoire avec un grand nombre de solutions envisageables
- On utilise une arborescence dont
  - Sommets : ensembles de solutions réalisables
  - Racine : ensemble de toutes les solutions réalisables
- L'expansion de l'arborescence est régie selon
  - Un principe de séparation
  - Un principe d'évaluation
  - L'utilisation d'une borne
  - Une stratégie de développement

Principe de séparation

- On partage, en fonction d'un certain critère...
- … l'ensemble des solutions réalisables contenues dans un sommet de l'arborescence...
  - ensemble qui n'est pas explicitement connu
- … en sous-ensembles qui deviennent alors dans l'arborescence les fils du sommet
- Il ne faut pas perdre ni ajouter de solution
  - L'union des sous-ensembles associés au fils du sommet doit être égale à l'ensemble associé au sommet
- Exemple

Principe de séparation

- Pour connaître la solution optimale
  - Il faut calculer la valeur de la fonction objectif pour toutes les feuilles non vides de l'arborescence
  - Ça fait beaucoup de feuilles !
- On peut améliorer la méthode pour éviter l'examen de certaines branches
  - On ne développera pas un sommet
    - Lorsqu'on pourra montrer qu'il ne contient pas la solution optimale
    - Lorsqu'on sait résoudre directement le problème correspondant à ce sommet

Principe d'évaluation et utilisation de la borne

- Pour un problème d'optimisation où on maximise un objectif
  - Borne : valeur qu'on sait atteindre pour l'objectif par une solution réalisable (minorant du maximum)
  - Évaluation d'un sommet
    - Détermination d'un majorant de l'ensemble des valeurs de l'objectif des solutions de ce sommet
    - On saura alors qu'on ne pourra faire mieux que la valeur de l'évaluation
    - Elle est dite exacte lorsque la valeur donnée par l'évaluation est atteinte par un élément de l'ensemble associé au sommet

Principe d'évaluation et utilisation de la borne

- Pour une maximisation, la borne et l'évaluation permettent de ne pas développer un sommet *S*
  - Si l'évaluation de *S* est inférieure ou égale à la borne (on ne pourra pas trouver une solution meilleure que la borne dans cette branche)
  - Si l'évaluation de S est exacte et strictement supérieure à la borne
    - On a alors trouvé une meilleure solution que la borne
    - On modifie donc la borne et on se retrouve au cas précédent (évaluation de S inférieure ou égale à la borne)

Principe d'évaluation et utilisation de la borne

- Il est donc important
  - De calculer une bonne borne
  - De concevoir une fonction d'évaluation assez fine
  - De sorte à éliminer le plus de branches possibles le plus tôt possible
- Par contre
  - Les calculs associés doivent pouvoir être effectués en temps raisonnable...

Stratégie de développement

- Détermination de la façon dont on va construire l'arborescence (application du critère de séparation)
  - Stratégie en profondeur
  - Stratégie de l'évaluation la plus grande
  - Stratégie en largeur

Stratégie de développement en profondeur

- On descend dans les branches jusqu'à trouver un sommet qu'on peut éliminer
  - On remonte pour redescendre dans une autre direction
  - Si on connaissait l'arborescence, ce serait équivalent à un parcours en profondeur
- Avantages
  - Minimise la place mémoire → on ne conserve que la branche explorée et non toute l'arborescence
  - Minimise les accès disque si le problème est trop grand pour tenir en mémoire principale

Stratégie de l'évaluation la plus grande

- Pour une maximisation
  - On pourrait suspecter que les sommets qui ont l'évaluation la plus grande contiennent la solution optimale
  - Pas nécessairement justifié
- Stratégie de type « meilleur d'abord »
- Gestion de l'arborescence plus difficile et qui consomme plus d'espace mémoire

Stratégie en largeur

- On évalue tous les sommets à un niveau donné avant de descendre
- Peu utilisé
  - Rarement efficace
  - Très grande occupation mémoire
- Exemple : application au sac à dos

Méthodes par séparation et évaluation

Application au problème du Voyageur de Commerce

Forme linéaire du problème du Voyageur de Commerce

- On considère le graphe complet $K_n = (X, E)$ à n sommets
- Valuation sur l'ensemble E des arêtes
  - Poids d'une arête $\{u,v\} \rightarrow p_{uv}$
- Variable associée à chaque arête $\{u,v\}$
  - Variable bivalente (0-1) $\rightarrow x_{uv}$
  - 1 si on garde l'arête, 0 sinon

Forme linéaire du problème du Voyageur de Commerce

- Représentation de la contrainte de constituer un cycle hamiltonien
  - Pour tout sommet v, la somme des valeurs des variables associées aux arêtes ayant v comme extrémité est égale à 2
    - Dans un cycle hamiltonien, tout sommet est de degré 2
  - Pour tous sous-ensemble Y de X autre que X, le nombre d'arêtes ayant ses 2 extrémités dans Y est strictement plus petit que |Y|
    - L'ensemble des arêtes conservées ne se décompose pas en plusieurs cycles

Forme linéaire du problème du Voyageur de Commerce

Minimiser $\sum_{\{u,v\} \in E} p_{uv} x_{uv}$

Sous les Contraintes

$\forall u \in X, \quad \sum_{v \in X} x_{uv} = 2$

$\forall Y \subset X$ avec $Y \neq X, \quad \sum_{\substack{\{u,v\} \in E \\ u \in Y,\, v \in Y}} x_{uv} < |Y|$

$\forall \{u, v\} \in E, \quad x_{uv} \in \{0,1\}$

On minimise le poids total des arêtes retenues du cycle hamiltonien

Dans un cycle hamiltonien, tout sommet est de degré 2

Pas de cycle dans l'ensemble des arêtes conservées

Les variables peuvent prendre la valeur 0 (on ne conserve pas l'arête) ou 1 (on conserve l'arête)

Définition d'une fonction d'évaluation

- Une chaîne constitue un arbre particulier
  - La chaîne obtenue en supprimant d'un cycle hamiltonien de G un sommet quelconque x_0 et les 2 arêtes qui lui sont adjacentes
    - est un arbre couvrant de $G - x_0$
    - est de poids supérieur ou égal au poids d'un arbre couvrant de poids minimum de $G - x_0$
- Cycle hamiltonien $\rightarrow$ tout sommet est de degré 2
  - La somme des poids des 2 arêtes d'un cycle hamiltonien adjacentes à un x_0 fixé
    - Est supérieure ou égale à la somme des poids des 2 arêtes les plus légères incidentes à x_0 dans G

Définition d'une fonction d'évaluation

- Si on choisit de façon arbitraire un sommet x_0
  - Le poids d'un cycle hamiltonien quelconque est supérieur ou égal à la somme des poids des 2 arêtes les plus légères adjacentes à x_0 …
  - … + le poids d'un arbre couvrant de poids minimum de $G - x_0$
  - Cette quantité est un minorant du poids d'un cycle hamiltonien optimal
  - On pourra donc l'utiliser comme fonction d'évaluation → on ne pourra trouver de cycle hamiltonien de poids inférieur à cette quantité

Définition d'une borne

- Il faut définir un cycle hamiltonien correspondant au «mieux que l'on puisse faire »
  - Majorant du minimum
  - Permettra de couper les branches correspondant à des solutions de poids supérieur
- Choisir un cycle hamiltonien au hasard ?
  - Mauvaise borne → on ne coupera pas grand chose !

Définition d'une borne

- Heuristique du plus proche voisin
  - On choisit aléatoirement un sommet de départ …
  - … puis on prend le sommet le plus proche …
  - … et on répète jusqu'à avoir inclus tous les sommets, puis on reconnecte le sommet initial
  - $O(n^2)$
- Il existe de meilleures méthodes, plus complexes, qu'on verra plus tard
  - Heuristiques spécialisées, métaheuristiques

Description d'une méthode par séparation et évaluation

- On choisit (une fois pour toutes) un sommet x_0
- À chaque sommet *S* de l'arborescence (incluant x_0)
  - On l'évalue par la méthode vue précédemment (acpm de G – x_0 + les 2 arêtes les plus légères partant de x_0)
  - Si l'évaluation est supérieure ou égale à la borne, on abandonne *S* (on sait déjà faire aussi bien)
  - Sinon, on développe en séparant les sommets qui correspondent à un degré supérieur à 2 en interdisant successivement les arêtes
- Exemple

La semaine prochaine

Méthodes de résolution approchées pour le Voyageur de Commerce