Résolution approchée de PLNE par des heuristiques

RCP 104

CNAM, 2012-2013

Résolution « approchée » ?

- Résoudre un PLNE peut prendre du temps !
- Mais si on a besoin d'une solution rapidement ?
  - Idée : rechercher une « bonne » solution
    - Solution admissible
    - Pas nécessairement optimale, mais meilleure que la plupart des autres solutions admissibles !
    - Intérêt : recherche rapide (si possible, cf PVC) !
- Ces sol. (et les algorithmes pour les rechercher) sont dit(e)s approché(e)s, ou **heuristiques**
- Possibilité de validation par des **bornes inf./sup.**

Rappel : définition et propriétés des relaxations

- Définition limitée à dessein au cadre de ce cours :

Un problème (R) est une **relaxation** d'un PLNE (P) **si toute solution de (P) est solution de (R)**

1. **[Min] Si (R) est une relaxation de (P), alors sa valeur optimale est une borne inférieure pour (P)**
2. Si (R) n'a pas de solutions admissibles, (P) non plus
3. Si une solution optimale de (R) est admissible pour (P), alors elle est également optimale pour (P)

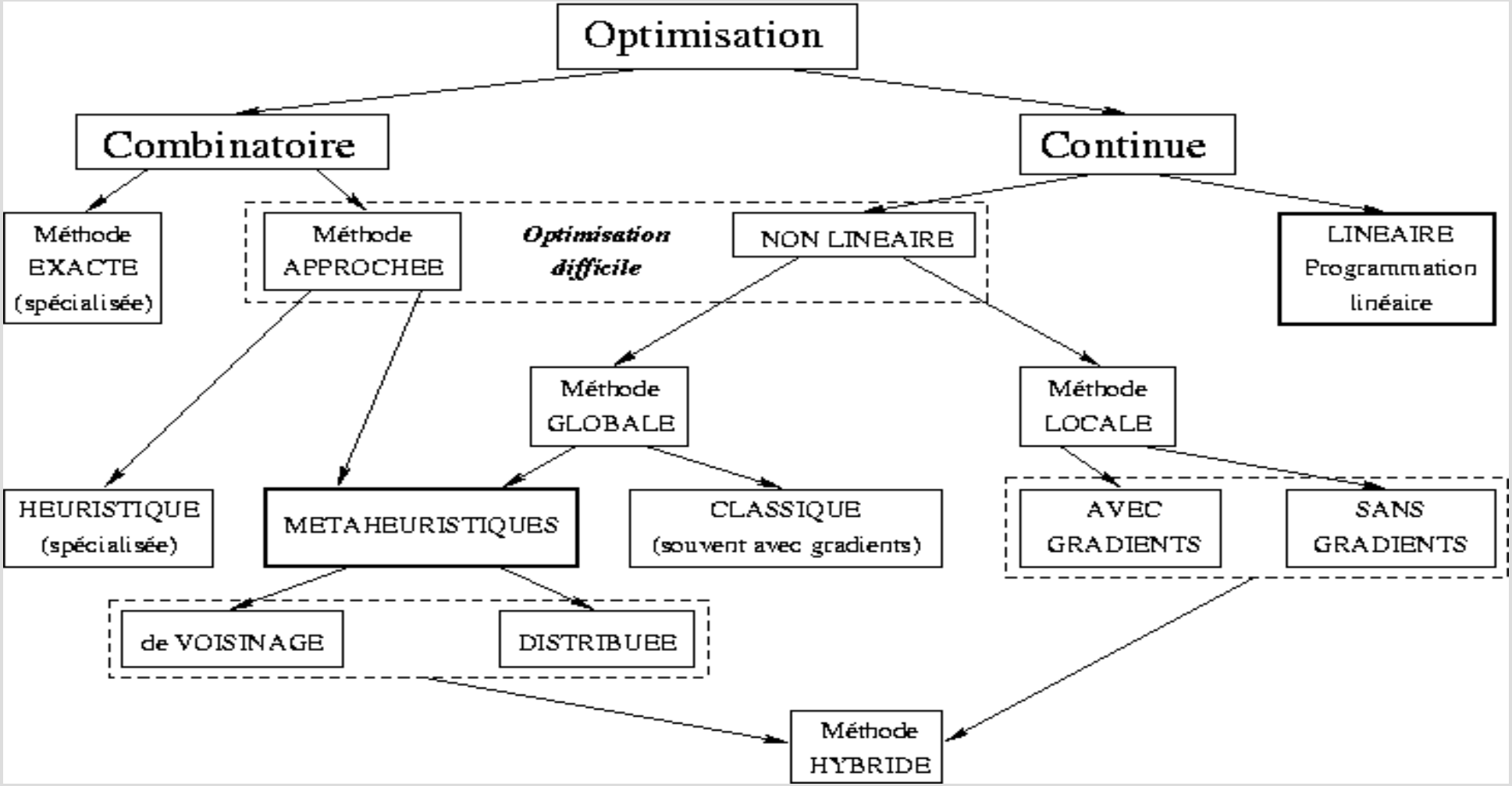
Rappel : relaxation continue (RC)

- Soit le PLNE (P)={min f(x), x admissible et entier}
  - Question : relaxation (R) plus simple à résoudre ?
- Le PL (R)={min f(x), x admissible} est appelé **relaxation continue** (ou linéaire) de (P)
  - Dans (R), on ignore les contraintes d'intégrité de (P)
  - Propriété : (R) est bien une relaxation de (P), car toute solution de (P) est solution de (R) !
  - Corollaire (cf PL1) : si (R) admet une solution optimale entière, cette solution est optimale pour (P) !
  - Intérêt de (R) : (R) est un PL, et peut donc être résolu à l'aide d'algorithmes efficaces (tels le simplexe) !

Rappel : encadrement de la valeur optimale d'un PLNE

- Une relaxation d'un PLNE fournit une borne inf. (si on min.) ou sup. (si on max.)
- La valeur d'une solution admissible fournit une borne sup. (si on min.) ou inf. (si on max.)
- La valeur optimale se trouve quelque part entre ces deux bornes !
- Une heuristique peut donc également être utilisée (par exemple) pour initialiser un B&B (méthode « exacte », c'est-à-dire avec garantie d'optimalité)

Panorama des méthodes de résolution



Algorithmes approchés

- Différentes méthodes à utiliser :
  - Méthodes par S&E avortées
  - Algorithmes approchés particuliers (adaptés au pb)
    - Méthode/heuristique gloutonne (PVC, Couverture)
    - Heuristique de réparation (SAD, SAD bidimensionnel)
    - Heuristique par recherche locale
    - Autre algorithme/heuristique spécifique
  - Méthodes génériques (métaheuristiques)
    - Recherche tabou (ou “avec tabous”)
    - Recuit simulé
    - Algorithmes génétiques
    - Autres : VNS, colonies de fourmis, etc.

Méthodes par S&E avortées

- Consistent simplement à exécuter partiellement une méthode par Séparation & Evaluation, et à l'arrêter quand un critère d'arrêt est vérifié :
  - Temps limite (risque = pas de borne primale !)
  - Ecart prédéfini entre borne primale / borne duale
  - Critère mixte : temps limite si borne primale existe OU écart prédéfini entre borne primale/duale
- Avantage : réutilise les méthodes par S&E !
- Inconvénient : réutilise les méthodes par S&E !

Heuristiques gloutonnes

- Description générale :
  - Principe : faire le choix (afin d'obtenir une solution partielle) qui semble être le meilleur à l'étape courante, et construire le reste de la solution sans jamais remettre en cause ce choix
  - Avantage : souvent très rapides, et la solution obtenue peut ensuite éventuellement être améliorée
  - Inconvénient : la solution obtenue n'est en général pas de très bonne qualité

Problème du voyageur de commerce (PVC)

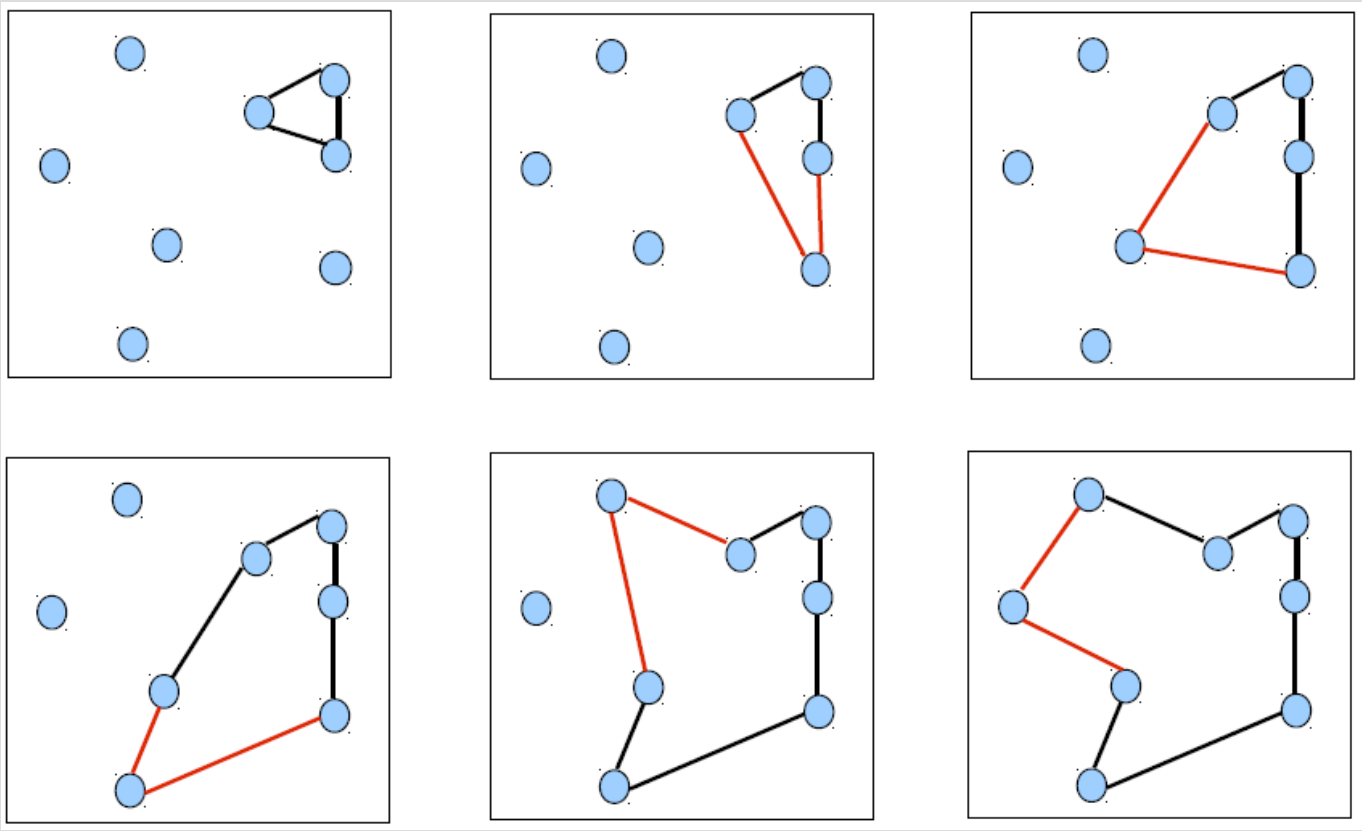
- Un graphe orienté G=(S,A), dont les n sommets (villes) sont reliés par des arêtes (routes) valuées par des distances
- PVC = trouver, dans G, un circuit hamiltonien (= passant une et une seule fois par chaque ville) de distance totale minimum
- Remarque : dans le cas général, trouver un circuit hamiltonien (= une solution admissible) est déjà un problème « difficile », mais si G est « complet » c'est un problème trivial.

Heuristiques gloutonnes pour le PVC (G complet)

- Heuristique du plus proche voisin :
  - v := ville initiale v1 (v = ville courante)
  - R := S-{v1} (R = villes pas encore « visitées »)
  - Tant que R non vide faire
    - Visiter la ville v' de R la plus proche de v
    - R := R-{v'}
    - v := v'
  - Retourner en v1
- Temps d'exécution quadratique (en $O(n^2)$) :
  - Avantage : simple et relativement rapide
  - Inconvénient : ne tient pas compte des conséquences indirectes du choix consistant à aller à la ville la plus proche (par exemple, revenir à la ville initiale peut avoir un coût prohibitif)

Heuristiques gloutonnes pour le PVC (G complet)

- Heuristique par insertion : idée = construire un tour en introduisant à chaque étape une nouvelle ville dans un sous-tour (ou tour partiel)



Heuristiques gloutonnes pour le PVC (G complet)

- Heuristique par insertion :
  - Choisir arbitrairement deux villes v1 et v2
  - S' := {v1, v2} (S' = villes du sous-tour en construction)
  - A' := {(v1,v2), (v2,v1)} (A' = arêtes du sous-tour en construction)
  - Tant que |S'| < |S| faire
    - S' := S' + {une ville v choisie arbitrairement dans S-S'}
    - Déterminer les 2 villes v' et v'' consécutives dans S' qui minimisent la quantité dist(v',v)+dist(v,v'')-dist(v',v'')
    - A' := A' + {(v',v), (v,v'')} – (v',v'')

Heuristiques gloutonnes pour le PVC (G complet)

- Deux variantes :
  - Plus proche insertion : on choisit la ville v la plus proche du sous-tour courant, çàd qui minimise min{dist(v',v) : v' dans S'}
  - Plus lointaine insertion : on choisit la ville v la plus éloignée du sous-tour courant, çàd qui maximise min{dist(v',v) : v' dans S'}
- Temps d'exécution quadratique pour chaque variante
  - La seconde insère au mieux les villes les plus lointaines, qui devront faire partie du tour de toute façon
  - En pratique (c'est-à-dire empiriquement), on constate que c'est en général cette variante qui est la plus efficace

Heuristique gloutonne pour le problème de couverture

- Problème de couverture :
  - Donnée : un graphe non orienté G=(S,A)
  - Problème : trouver un sous-ensemble de sommets C de S de cardinalité minimale tel que, pour toute arête [u,v] de A, u est dans C ou v est dans C
- Heuristique gloutonne :
  - C := ensemble vide (couverture en construction)
  - A' := A (arêtes pas encore « couverte »)
  - Tant que A' non vide faire
    - Choisir arbitrairement une arête [u,v] dans A'
    - C := C + {u,v}
    - A' := A' – {[u,v]} – {arêtes ayant une extrémité en commun avec [u,v]}
- Au plus |A| itérations, et la solution C obtenue a au plus deux fois le nombre de sommets d'une solution optimale (car les arêtes choisies n'ont pas d'extrémités en commun) ➔ « garantie de performance » de 2 a priori (peut être meilleure en pratique)

Heuristiques de réparation

- Principe :
  - Déterminer une « bonne » solution non admissible (obtenue, par exemple, à l'aide d'une relaxation)
  - Reconstruire, à partir de cette solution non admissible, une solution admissible sans trop dégrader la valeur de la fonction objectif
  - Plusieurs types de relaxations :
    - Relaxation continue ➔ arrondi de la RC
    - Relaxation par agrégation de contraintes
    - Etc.

Algorithme approché spécifique : le sac-à-dos (1/2)

- Idée : mettre à 1 toutes les variables qui valent 1 dans la solution optimale de la RC (et à 0 les autres) fournit-il une bonne solution entière ?
- Vérifions sur un exemple :

$\max x+(b-1)y$

$x+by \leq b$

$x,y \in \{0,1\}$

- On prend x=1 et y=0, car 1/1>(b-1)/b :
  - Valeur de la solution (entière) obtenue = 1
  - Valeur de la solution (entière) optimale = b-1 !

Algorithme approché spécifique : le sac-à-dos (2/2)

- 2e idée : prendre la meilleure de 2 sol. simples
  - 1ère sol. : la solution précédente (**arrondi** de la RC)
  - 2e sol. : choisir le 1er objet non inclus dans le SAD
  - Solution admissible, facile à calculer
- Analyse du pb $\{\max \Sigma_i c_i x_i : \Sigma_i a_i x_i \leq b,\ x_i \in \{0,1\}\ \forall i\}$:
  - Soit j la dernière variable à 1 dans la 1ère solution : 1ère sol. de valeur $\Sigma_{i=1 \text{ à } j}\, c_i$ et 2ème sol. de valeur c_{j+1}
  - Solution approchée de valeur $\max\{\Sigma_{i=1 \text{ à } j}\, c_i,\ c_{j+1}\}$
  - « Garantie de performance » de 2 *a priori*

Heuristique de réparation pour le sac-à-dos bidimensionnel

- Sac-à-dos bidimensionnel :
  - Mêmes données et fonction objectif que le sac-à-dos
  - Deux contraintes au lieu d'une
- Principe de l'heuristique de réparation :
  - Sommer les 2 contraintes en une seule
  - Résoudre l'instance de sac-à-dos obtenue
  - Retirer les objets qui « débordent » du sac, en commençant par les moins « intéressants »

A retenir...

- Avantages des algo. approchés spécifiques :
  - Exploitent les particularités du problème
  - Parfois, analyse théorique de l'algo. possible (garantie *a priori*)
  - Peuvent être utilisés **avant** une métaheuristique ou même un Branch & Bound (solution initiale)
- Inconvénients des algo. approchés spécifiques :
  - Exploitent les particularités du pb = non portables !
  - Contrairement aux méthodes par S&E avortées, la conception d'un tel algorithme reste délicate quand trouver une borne primale est difficile (cf PVC dans le cas général)