Projet

Zacharie ALES - Daniel PORUMBEL

2024 - 2025

Déroulement général du module

Programmation

- 12/11 : TP Julia
- 19/11 : TP C++
- 26/11 : TP OPL

Projet

- Aujourd'hui : Cours robustesse + présentation du projet
- 03/12, 10/12, 17/12 : Séance de projet (sur rendez-vous)
- 15/12 : Rendu partie théorique
- 07/02 : Rendu rapport final
- 14/02 : Soutenances

Modalités de notations

1. Rendu de la partie théorique
2. Rapport final
   Partie théorique et pratique
3. Contenu scientifique
4. Soutenance
5. Utilisation de Github

A faire avant les TP

Installation de logiciels

- CPLEX (qui inclut OPL)
- Julia
- C++

Sous Windows, le lien entre CPLEX et VisualStudio n'est pas toujours aisé suivant les versions

Machine virtuelle

Possibilité d'utiliser une machine virtuelle

1. Récupérer la machine virtuelle
2. Installer le logiciel VirtualBox
   Ou installer les packages virtualbox et virtualbox-qt sous linux
3. Lancer le logiciel et créer la machine virtuelle
   - Nouvelle
   - Champ “Type” choisir “Linux
   - Champ “Version” choisir “Ubuntu (64-bit)”
   - Suivant > Fixer la mémoire vive > Suivant
   - Cocher "Utiliser un fichier de disque dur virtuel existant” > Sélectionner le fichier
   - Créer

MPRO - Projet d'optimisation robuste

Cours Projet

ALES Zacharie (zacharie.ales@ensta.fr)
PORUMBEL Daniel (daniel.porumbel@cnam.fr)

Adapté de la présentation ROADEF 2017 de Michael Poss

Outline

Sommaire

Optimisation robuste

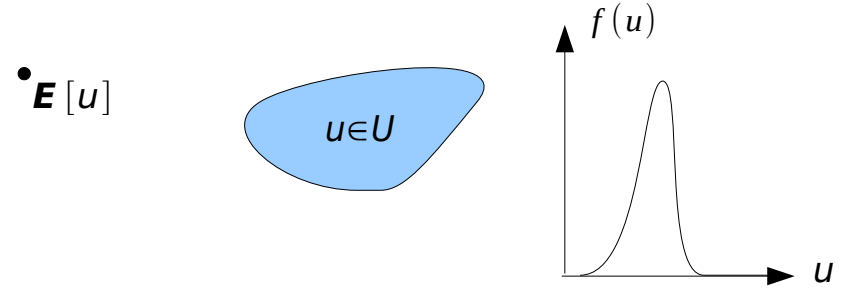
Notation

- u : donnée(s) d'un problème d'optimisation
  Ex : distance entre deux villes, poids d'un objet, ...

Problème

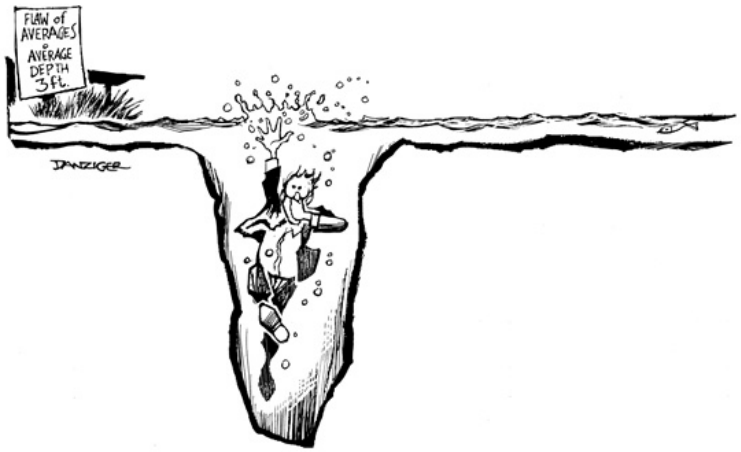
Comment prendre en compte l'aléa de la valeur de u ?

Mean value (Deterministic) *Robust* *Stochastic*



L'optimisation robuste considère le pire des cas

Contrairement à l'optimisation stochastique qui optimise en moyenne



Types de problèmes robustes

Statique

Prise de décisions ⇢ L'incertitude est révélée

- Facile pour la PL☺
- $\mathcal{NP}$-difficile pour l'optimisation combinatoire ☹
- Il existe des reformulation MILP ☺

Deux niveaux

Prise de décision ⇢ L'incertitude est révélée ⇢ Autres décisions

- $\mathcal{NP}$-difficile pour la PL ☹
- Algorithmes de décomposition ☺

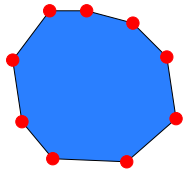
Plusieurs niveaux

Décisions ⇢ Incertitude ⇢ Décisions ⇢ Incertitude ⇢ $\cdots$

- $\mathcal{NP}$-difficile pour la PL☹
- Non résolvable optimalement ☹

Incertitude discrète ou continue

Soit $\mathcal{U}$ = sommets($\mathcal{P}$)



Remarque

Dans la plupart des cas, $\mathcal{U} \sim \mathcal{P}$

Exceptions

- contraintes robustes $f(x,u) \leq b$ avec f non-concave en u
- problèmes multi-niveaux avec des variables d'ajustement entières

  Décisions prises après que l'incertitude soit révélée

Optimisation combinatoire robuste

CO - Problème combinatoire

- $\mathcal{X} \subseteq \{0,1\}^n, u \in \mathbb{R}^n$ (Données ; Contraintes)

$$\min_{x \in \mathcal{X}} ux$$

(Variables)

$\mathcal{U}$-*CO* - Problème combinatoire robuste avec incertitude sur u

- $\mathcal{X} \subseteq \{0,1\}^n, \mathcal{U}^0 \subset \mathbb{R}^n$ (Valeurs possibles de u)

Solution x qui minimise... $\rightarrow$ $\min_{x \in \mathcal{X}} \max_{u^0 \in \mathcal{U}^0} u^0 x$ (... le coût de son pire scénario u^0)

Problème combinatoire robuste avec regret

$$\min_{x \in \mathcal{X}} \max_{u^0 \in \mathcal{U}^0} \min_{y \in \mathcal{X}} \left(u^0 x - u^0 y\right)$$

(Coût optimal pour u^0)

Solution optimale si on sait que l'incertitude prendra la valeur u^0

Contraintes robustes

Contraintes

$$\mathcal{X} = \mathcal{X}^{comb} \cap \mathcal{X}^{num}$$

$\mathcal{X}^{comb}$: Contraintes **statiques** — $\mathcal{X}^{num}$: Contraintes **robustes**

$\mathcal{U}$-CO

$$\min_x \max_{u^0 \in \mathcal{U}^0} u^0 x$$
$$\text{s.c. } x \in \mathcal{X}^{comb}$$
$$x \in \mathcal{X}^{num}$$

Exemple de contraintes robustes

- $\mathcal{X}^{num} = \left\{ x \in \{0,1\}^n \text{ tel que } {u^j}^T x \leq b_j, \quad \mathbf{u^j} \in \mathcal{U}^j \; \forall j \in [m] \right\}$

$[m] \triangleq \{1, 2..., m\}$

Valeurs possibles des coefficients de la contrainte j

Exemple de problème robuste

Sac à dos statique

v_i : valeur en € de l'objet i — $x_i = \begin{cases} 1 & \text{si } i \text{ est dans le sac} \\ 0 & \text{sinon} \end{cases}$

$$\max_x v^T x$$

$$p^T x \leq K \leftarrow \text{Capacité du sac}$$

p_i : poids de l'objet i

$$x \in \{0,1\}^n$$

Variante robuste

$$\max_x \min_{v \in \mathcal{V}} v^T x$$

Incertitude sur les prix

$$p^T x \leq K \qquad p \in \mathcal{P}$$

Incertitude sur les poids

$$x \in \{0,1\}^n$$

Exemple de problème robuste

$\mathcal{V}$: incertitude sur les valeurs (objectif)

Au plus 10€ est retiré à la valeur des objets

$$\mathcal{V} = \left\{ v = \bar{v} - \delta^v \mid \dots \right\}$$

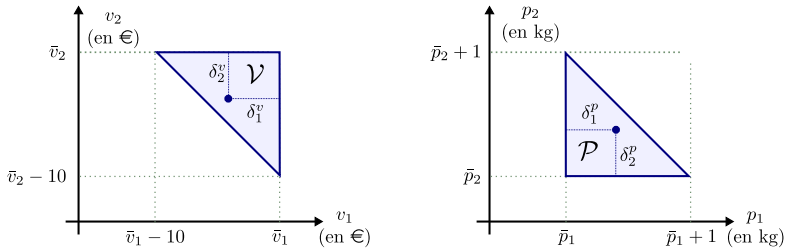
$\bar{v}$: Valeur nominale **(constante)** — δ^v : Valeur retirée **(variable)**

$\mathcal{P}$: incertitude sur les poids (contraintes)

1kg peut être ajouté au poids de l'ensemble des objets

$$\mathcal{P} = \left\{ p = \bar{p} + \delta^p \mid \dots \right\}$$

$\bar{p}$: Poids nominal **(constant)** — δ^p : Poids ajouté **(variable)**



Incertitude discrète : $\mathcal{U}$-CO est difficile - Kouvelis and Yu [2013]

Théorème

Le plus court chemin robuste, l'affectation, l'arbre couvrant, ..., sont $\mathcal{NP}$-difficile même quand $|\mathcal{U}| = 2$

i.e. 2 ***scenarios***

Preuve - Problème de sélection

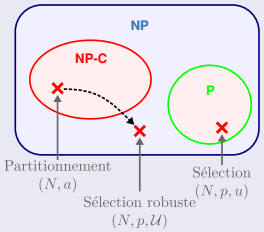
1. Problème de sélection : $\min_{S \subseteq N, |S| = p} \sum_{i \in S} u_i$

   Choisir p objets de poids minimal

2. Problème de sélection ROBUSTE : $\min_{S \subseteq N, |S| = p} \max_{u \in \mathcal{U}} \sum_{i \in S} u_i$

3. Problème de partitionnement : $\min_{S \subseteq N, |S| = |N|/2} \max\left(\sum_{i \in S} a_i, \sum_{i \in N \setminus S} a_i \right)$

   Poids minimal de la plus grande des deux parties



   Construction d'une instance de séléction robuste à partir d'une instance de partitionnement

4. Réduction : $p = \frac{|N|}{2}$, et $\mathcal{U} = \{u^1, u^2\}$ tel que

   On séléctionne la moitié des objets

$$u_i^1 = a_i \quad \text{et} \quad u_i^2 = \frac{2}{|N|} \sum_k a_k - a_i$$

$\sum_{i \in N} u_i^1$: poids des objets séléctionnés — $\sum_{i \in N} u_i^2$: poids des autres objets

$$\Rightarrow \quad \forall S \; \max_{u \in \mathcal{U}} \sum_{i \in S} u_i = \max\left(\sum_{i \in S} a_i, \sum_{i \in N \setminus S} a_i \right)$$

Incertitude polyhédrale: $\mathcal{U}$-CO est toujours difficile (mais résolvable)

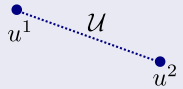
Est-ce mieux quand l'incertitude est décrite par un polytope plutôt que des scénarios ?

Théorème

Le plus court chemin robuste, l'affectation, l'arbre couvrant, ... sont $\mathcal{NP}$-difficile même quand $\mathcal{U}$ a une description compacte

Preuve

1. $\mathcal{U} = \text{conv}(u^1, u^2)$
2. $ux \leq b, \quad u \in \mathcal{U} \quad \Leftrightarrow \quad ux \leq b, \quad u \in \text{ext}(\mathcal{U})$



Théorème

Ben-Tal and Nemirovski [1998]

Le problème $\mathcal{U}$-CO est équivalent à un MILP

Sommaire

Dualisation - Incertitude sur les coûts

Théorème

Ben-Tal and Nemirovski [1998]

Soient $\alpha \in \mathbb{R}^{\ell \times n}$ et $\beta \in \mathbb{R}^{\ell}$ définissant le polytope

$$\mathcal{U} := \{u \in \mathbb{R}^n_+ \ : \ \alpha u \leq \beta\}$$

Le problème $\min_{x \in \mathcal{X}} \max_{u \in \mathcal{U}} ux$ est équivalent à un MILP compact

Ne veut pas dire que le problème garde sa complexité (ex : perte d'unimodularité)

Preuve

1. Dualiser la maximisation interne

$$\min_{x \in \mathcal{X}} \max_{u \in \mathcal{U}} ux = \min_{x \in \mathcal{X}} \min \left\{ \beta^T z \ : \ \sum_{k \in [\ell]} \alpha_{ki} z_k \geq x_i, i \in [n], z \geq 0 \right\}$$

2. Regrouper les deux min

Exemple de dualisation - Problème de sac à dos

Rappel

- $\mathcal{V} = \left\{ v = \overline{v} - \delta^v \mid \sum_{i \in [n]} \delta^v_i \leq 10, \ \delta^v \geq 0 \right\}$

1.1 - Isoler δ^v

$\max_x \min_{v \in \mathcal{V}} v^T x =$ .... $=$ ....

Développer $\mathcal{V}$

$\sum_{i \in [n]} \delta^v_i \leq 10$, $\delta^v \geq 0$; $\sum_{i \in [n]} \delta^v_i \leq 10$, $\delta^v \geq 0$

2.1 - Dualiser la minimisation interne

$\min_{\delta^v} -{\delta^v}^T x = \max_{\alpha} 10\alpha$

$\sum_{i \in [n]} \delta^v_i \leq 10 \ \ (\alpha)$, $\delta^v \geq 0$; $\alpha \leq -x_i \ \ i \in [n]$, $\alpha \leq 0$

x est considéré constant pour la dualisation !

3.1 - Remplacer la minimisation par son dual dans le problème initial

$\max_x \left\{ \overline{v}^T x + \min_{\delta^v} -{\delta^v}^T x \right\} =$ .... $=$ ....

$\sum_{i \in [n]} \delta^v_i \leq 10$, $\delta^v \geq 0$; $\alpha \leq -x_i \ \ i \in [n]$, $\alpha \leq 0$; $\alpha \leq -x_i \ \ i \in [n]$, $\alpha \leq 0$

Example de dualisation - Problème de sac à dos

Rappel

- $\mathcal{P} = \left\{ p = \bar{p} + \delta^p \mid \sum_{i\in[n]} \delta_i^p \leq 1,\ \delta^p \geq 0 \right\}$

1.2 - Isoler δ^p

$p^T x \leq K \ \ \forall p \in \mathcal{P} \Leftrightarrow$ ..... such that $\sum_{i\in[n]} \delta_i^p \leq 1$ and $\delta^p \geq 0$

Développer $\mathcal{P}$

2.2 - Dualiser la maximisation interne

$$\max_{\delta^p} x^T \delta^p \quad \sum_{i\in[n]} \delta_i^p \leq 1 \ \ (\beta) \quad \delta^p \geq 0 \qquad = \min_{\beta} \beta \quad \beta \geq x_i \ \ i \in [n]$$

3.2 - Remplacer la minimisation par son dual dans le problème initial

$$\bar{p}^T x + \max_{\delta^p} \delta^{pT} x \leq K \quad \Leftrightarrow \quad \sum_{i\in[n]} \delta_i^p \leq 1,\ \delta^p \geq 0$$

..... $\beta \geq x_i \ \ i \in [n]$ $\Leftarrow$ ..... $\beta \geq x_i \ \ i \in [n]$

Sommaire

Algorithme de plans coupants (Bertsimas et al. [2016])

$\mathcal{U}$-CO

$$\min_x \left\{ \max_{v\in\mathcal{V}} v^T x : \ x \in \mathcal{X}^{num} \ \ x \in \mathcal{X}^{comb} \right\}$$

Reformulation de l'objectif

$$\min_{z,x} \left\{ \ . : \ \text{.....}', \ \text{.....}', \ x \in \mathcal{X}^{num} \ \ x \in \mathcal{X}^{comb} \right\}$$

Algorithme de plans coupants (Bertsimas et al. [2016])

Definition - **Problème maître (MP)**

$$\min_{z,x} \left\{ z : \ v^T x \leq z, \quad v \in \mathcal{V}^* \subset \mathcal{V}, \ u^{jT} x \leq b_j, \quad j = 1,\dots,m, \ u^j \in \mathcal{U}^{j*} \subset \mathcal{U}^j, \ x \in \mathcal{X}^{comb} \right\}$$

Sous-problème 1 (SPo)

$$\max_{v\in\mathcal{V}} v^T x^*$$

Fixé !

Sous-problème 2 (SPj)

$$\max_{u^j\in\mathcal{U}^j} u^{jT} x^*$$

Fixé !

Algorithme des plans coupants

1. **Résoudre** $MP \rightarrow x^*, z^*$
2. **Résoudre** SPo $\rightarrow v^*$ **et** SP1 à SPm $\rightarrow u^{1*}, \dots, u^{m*}$
3. **Si** $v^{*T} x^* > z^*$ **et/ou** $u^{j*T} x^* > b_j$ **alors**
   - $\mathcal{V}^* \leftarrow \mathcal{V}^* \cup \{v^*\}$ **et/ou** $\mathcal{U}^{j*} \leftarrow \mathcal{U}^{j*} \cup \{u^{j*}\}$
   - aller à 1

Exemple d'algorithme de plans coupants - Problème de sac à dos

Notations

- $\mathcal{V} = \left\{ v = \overline{v} - \delta^v \mid \sum_{i \in S} \delta_i^v \leq 10,\ \delta^v \geq 0 \right\}$
  - $\overline{v}$: valeurs des objets (€)
- $\mathcal{P} = \left\{ p = \overline{p} + \delta^p \mid \sum_{i \in [n]} \delta_i^p \leq 1,\ \delta^p \geq 0 \right\}$
  - $\overline{p}$: poids des objets (kg)

1 - Reformuler l'objectif

Original	Reformulé
$\max_x \min_{v \in \mathcal{V}} v^T x$	$\max_{z,x}$.
$p^T x \leq K \quad p \in \mathcal{P}$	....
$x \in \{0,1\}^n$	$p^T x \leq K \quad p \in \mathcal{P}$
	$x \in \{0,1\}^n$

2 - Définir le problème maître (MP)

Remplacer $\mathcal{V}$ et $\mathcal{P}$ par de petits sous-ensembles $\mathcal{V}^*$ et $\mathcal{P}^*$

ex : $\{v = \overline{v}\}$ ex : $\{p = \overline{p}\}$

Exemple d'algorithme de plans coupants - Problème de sac à dos

3 - Résoudre MP

$\Rightarrow x^*$ et z^*

4 - Vérifier si x^* et z^* satisfont toutes les contraintes robustes de $\mathcal{V}$ et $\mathcal{P}$

$\Rightarrow$ nécessite de résoudre 1 sous-problème pour chaque ensemble d'incertitude

Exemple d'algorithme de plans coupants - Problème de sac à dos

Rappel

$$\mathcal{V} = \left\{ v = \overline{v} - \delta^v \mid \sum_{i \in S} \delta_i^v \leq 10,\ \delta^v \geq 0 \right\}$$

4.1 - Sous-problème lié à $\mathcal{V}$

$\min_{v \in \mathcal{V}} v^T x^* =$ ....

.....

.....

- Le résoudre $\Rightarrow \delta^{v*}$
- Si z^* et x^* ne satisfont pas le scénario δ^{v*} de $\mathcal{V}$
  Si $z^* > (\overline{v} - \delta^{v*})^T x^*$
  Ajouter le scénario au MP
  Ajouter la contrainte $z \leq (\overline{v} - \delta^{v*})^T x$

x^* est constant !

Exemple d'algorithme de plans coupants

Rappel

$$\mathcal{P} = \left\{ p = \overline{p} + \delta^p \mid \sum_{i \in [n]} \delta_i^p \leq 1,\ \delta^p \geq 0 \right\}$$

4.2 - Sous-problème lié à $\mathcal{P}$

$p^T x^* \leq K \ \forall p \in \mathcal{P} \Leftrightarrow$ ....

$\sum_{i \in [n]} \delta_i^p \leq 1$

$\delta^p \geq 0$

- Le résoudre $\Rightarrow \delta^{p*}$
- Si z^* et x^* ne satisfont pas le scénario δ^{p*} de $\mathcal{P}$
  Si $(\overline{p} + \delta^{p*})^T x^* > K$
  Ajouter le scénario au MP
  Ajouter la contrainte $(\overline{p} + \delta^{p*})^T x \leq K$

5

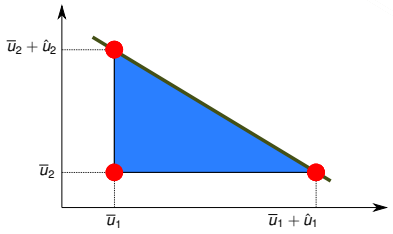
Répéter les étapes 3 et 4 jusqu'à ce que tous les scénarios soient satisfait

En pratique on ne sait pas si c'est la dualisation ou les plans coupants qui seront les plus efficaces

Sommaire

Optimisation combinatoire robuste $\mathcal{U}^\Gamma$

- En pratique, tous les paramètres ne prennent pas leur valeur maximale
  $\Rightarrow$ ensemble d'incertitude $\mathcal{U}^\Gamma$



$$\mathcal{U}^\Gamma = \left\{ \overline{u} \leq u \leq \overline{u} + \hat{u}, \sum_{i \in [n]} \frac{u_i - \overline{u}_i}{\hat{u}_i} \leq 1 \right\}$$

Théorème - - Bertsimas and Sim [2003], Goetzmann et al. [2011], Álvarez-Miranda et al. [2013], Lee and Kwon [2014]

Incertitude sur l'objectif $\mathcal{U}^\Gamma$-*CO* $\Rightarrow$ résoudre $\sim n/2$ problèmes *CO*

Incertitude sur m contraintes $\mathcal{U}^\Gamma$-*CO* $\Rightarrow$ résoudre $\sim (n/2)^m$ problèmes *CO*

Autre ensembles convexes $\mathcal{U}$ (pour rappel $\mathcal{U} \Leftrightarrow \text{conv}(\mathcal{U})$)

Déviation totale

$$\left\{ \overline{u} \leq u \leq \overline{u} + \hat{u}, \sum_{i \in [n]} (u_i - \overline{u}_i) \leq \Omega \right\} \Rightarrow \text{résoudre 2 problèmes } CO$$

Incertitude de type sac à dos - Poss [2017]

$$\left\{ \overline{u} \leq u \leq \overline{u} + \hat{u}, \sum_{i \in [n]} a_i u_i \leq b \right\} \Rightarrow \text{résoudre } n \text{ problèmes } CO$$

Decision-dependent - Poss [2013, 2014], Nohadani and Sharma [2016]

$$\left\{ \overline{u} \leq u \leq \overline{u} + \hat{u}, \sum_{i \in [n]} a_i u_i \leq b(x) \right\} \Rightarrow \text{résoudre } n \text{ problèmes } CO$$

Ellipsoides aux axes parallèles - Mokarami and Hashemi [2015]

$$\left\{ \sum_{i \in [n]} \left(\frac{u_i - \overline{u}_i}{\hat{u}_i} \right)^2 \leq \Omega \right\} \Rightarrow \text{résoudre } n \max_i \hat{u}_i \text{ problèmes } CO$$

Est-ce que tous les problèmes sont simples ?

Non, les problèmes difficiles doivent :

1. avoir un nombre non-constant de contraintes "linéaires" robustes ; ou
2. avoir des contraintes ou coûts "non-linéaires"

Théorème - Pessoa et al. [2015]

Le plus court chemin $\mathcal{U}^{\Gamma}$-robuste avec fenêtre de temps est $\mathcal{NP}$-difficile au sens fort

Théorème - Bougeret et al. [2016]

Minimiser la somme pondérée des dates de fin est $\mathcal{NP}$-difficile au sens fort

Conclusion

Problèmes statiques

- Solutions numériques par **dualisation** ou **décomposition**
- $\mathcal{U}$ "bonne" structure et objectif non linéaire
  $\Rightarrow$ problème ouverts intéressants

Problèmes ajustables

- Sujet d'actualité
- Très dur à résoudre !

Sommaire

Présentation du projet

Objectif

Implémenter et comparer des méthodes de résolution pour un problème de tournées de véhicules robuste

Notations

- $V = \{1, ..., n\}$ sommets :
  - 1 : entrepôt contenant des vaccins
  - $\{2, ..., n\}$: clients à livrer
- $t_{ij} \in \mathbb{N}$: durée du trajet entre les sommets i et j
- $d_i \in \mathbb{N}$: nombre de vaccins à livrer au client i
- $C \in \mathbb{N}$: capacité en vaccin de chaque véhicule

Présentation du projet

Définition - **Tournée**

Suite de sommets $T = \{t_1, t_2, ..., t_{|T|}\}$ telle que :

- la tournées débute et termine à l'entrepôt
  $t_1 = t_{|T|} = 1$
- la somme des demandes des clients visités n'excède pas la capacité d'un véhicule
  $\sum_{i=2}^{|T|-1} d_{t_i} \leq C$
- tous les clients visités sont différents
  $t_i \neq t_j \ \forall i, j \in \{2, ..., |T|-1\} \ i \neq j$

Problème de tournées de véhicules statique

$\min_{x \in \mathcal{X}^{comb}} tx$ ← Durée des tournées

$\mathcal{X}^{comb} \subseteq \left\{ x \in \{0,1\}^{|V^2|} \mid x \text{ est un ensemble de tournées livrant tous les clients} \right\}$

Présentation du projet

Problème de tournées de véhicules robuste

$$\min_{x \in \mathcal{X}^{comb}} \max_{t' \in \mathcal{U}} t'x$$

Donnée Donnée

$$\mathcal{U} = \left\{ \{ t'_{ij} = \hat{t}_{ij} + \delta^1_{ij}(\hat{t}_i + \hat{t}_j) + \delta^2_{ij}\hat{t}_i\hat{t}_j \}_{ij \in A} \right.$$

$$\left. \text{tel que } \sum_{ij \in A} \delta^1_{ij} \leq T,\ \sum_{ij \in A} \delta^2_{ij} \leq T \times T,\ \delta^1_{ij} \in [0,1],\ \delta^2_{ij} \in [0,2]\ \forall ij \in A \right\}$$

Donnée Variable Variable

Instances

Exemple de fichier de données

```
n = 5
t = [  0 260 864 263 374;
     260   0 796  59 114;
     864 796   0 855 797;
     263  59 855   0 130;
     374 114 797 130  0]
th = [6, 1, 2, 8, 1]
T = 6
d = [0, 6, 2, 5, 3]
C = 12
```


Travail demandé

1 - Partie théorique

1. Modélisation **compacte** du problème statique
2. Modélisation **compacte** du problème robuste
3. Définition du sous-problème associé à $\mathcal{U}$
4. Dualisation du problème associé à l'incertitude de l'objectif

2 - Implémentation (langage de programmation libre)

Pour ce problème, implémenter et comparer

1. un algorithme de plans coupants
2. un branch-and-cut
3. la dualisation
4. une heuristique ou une formulation non compacte

Algorithme de plans coupants

Principe

MP$\leftarrow$ problème de partitionnement statique

1. $x^* \leftarrow$ résoudre MP
2. $u^* \leftarrow$ résoudre le sous-problème associé à l'objectif
   Ajouter un scénario s'il n'est pas satisfait
3. Aller à l'étape 1 si une contrainte est ajoutée

Choix d'implémentation

- Définition de MP
  Choix des scénarios initialement dans $\mathcal{U}$
- Résolution du sous-problème
  De manière exacte ou approchée

Branch-and-cut

Principe

Similaire aux plans coupants mais les sous-problèmes sont résolus à chaque noeuds de l'arbre de branchement de MP qui retournent une solution entière
Avantage : ne pas repartir de 0 après chaque ajout de contrainte

Comment faire ?

Définition - **Callback**

Fonctions données en entrée d'un solveur MILP

Type de callbacks

Nom	Quand est-il appelé ?	Que fait-il ?
User cut	nouvelle relaxation	trouve des coupes
Lazy	nouvelle solution entière	trouve des coupes
Heuristic	nouvelle relaxation	trouve une solution entière
Branch	nouvelle relaxation	choisi sur quelle variable brancher
Node	nouvelle relaxation	choisi le prochain sommet à explorer

Choix d'implémentation

- Similaire aux plans coupants
- Utilisation possible d'autres callback

Dualisation

Principe

Résoudre le problème dualisé

Heuristique

Principe

Trouver une "bonne" solution heuristiquement

- Donner un gap en calculant la relaxation du problème dualisé

Comparaison des résultats

Comparaison des méthodes

- Temps de résolution à l'optimum
- Gap si le temps maximal est atteint

Comparaison des solutions statiques et robustes

- Objectif
- Nombre de tournées
- ...

Présentation des résultats

- Tableaux
  ex : une instance par ligne
- Diagramme de performances
  Trace le nombre d'instances résolues en fonction du temps

Links with other courses

- Optimisation dans l'incertain
- RORT project
  Shortest-path with attacker

Références I

E. Álvarez-Miranda, I. Ljubić, and P. Toth. A note on the bertsimas & sim algorithm for robust combinatorial optimization problems. 4OR, 11(4): 349–360, 2013.

A. Ben-Tal and A. Nemirovski. Robust convex optimization. Mathematics of Operations Research, 23(4):769–805, 1998.

D. Bertsimas and M. Sim. Robust discrete optimization and network flows. Math. Program., 98(1-3):49–71, 2003.

Dimitris Bertsimas, Iain Dunning, and Miles Lubin. Reformulation versus cutting-planes for robust optimization. Computational Management Science, 13(2):195–217, 2016.

M. Bougeret, Artur A. Pessoa, and M. Poss. Robust scheduling with budgeted uncertainty, 2016. Submitted.

K.-S. Goetzmann, S. Stiller, and C. Telha. Optimization over integers with robustness in cost and few constraints. In WAOA, pages 89–101, 2011.

P. Kouvelis and G. Yu. Robust discrete optimization and its applications, volume 14. Springer Science & Business Media, 2013.

Références II

Taehan Lee and Changhyun Kwon. A short note on the robust combinatorial optimization problems with cardinality constrained uncertainty. 4OR, pages 373–378, 2014.

Shaghayegh Mokarami and S Mehdi Hashemi. Constrained shortest path with uncertain transit times. Journal of Global Optimization, 63(1): 149–163, 2015.

Omid Nohadani and Kartikey Sharma. Optimization under decision-dependent uncertainty. arXiv preprint arXiv:1611.07992, 2016.

A. A. Pessoa, L. Di Puglia Pugliese, F. Guerriero, and M. Poss. Robust constrained shortest path problems under budgeted uncertainty. Networks, 66(2):98–111, 2015.

M. Poss. Robust combinatorial optimization with variable budgeted uncertainty. 4OR, 11(1):75–92, 2013.

M. Poss. Robust combinatorial optimization with variable cost uncertainty. European Journal of Operational Research, 237(3):836–845, 2014.

Michael Poss. Robust combinatorial optimization with knapsack uncertainty. 2017. Available at `hal.archives-ouvertes.fr/tel-01421260`.

MPRO – Projet ECMA 2024 - 2025

1 Objectifs du projet

Le but de ce projet est l'étude d'un problème de tournées de véhicules robuste. Vous devrez résoudre ce problème :

1. par un algorithme de plans coupants ;
2. par un algorithme de *branch-and-cut* (via un LazyCallback) ;
3. par dualisation ;
4. par une heuristique ou la résolution d'une formulation non compacte.

2 Présentation du problème

2.1 Problème statique

On considère un problème de tournées de véhicules dans lequel un entrepôt doit livré des vaccins à un ensemble de clients. Le problème est caractérisé par :

- un ensemble de sommets $V = \{1, ..., n\}$ tel que le sommet 1 correspond à l'entrepôt des véhicules où se trouve le stock de vaccins et les autres sommets aux clients qu'il faut livrer. Soit $A = \{ij \in V^2 \mid i \neq j\}$;
- une durée $t_{ij} \in \mathbb{N}$ associée à chaque couple de sommets $(i, j) \in A$ $i \neq j$ correspondant au temps minimal pour qu'un véhicule se rende du sommet i au sommet j ;
- une demande $d_i \in \mathbb{N}$ associée à chaque sommet $i \in V \backslash \{1\}$ correspondant au nombre de vaccins à livrer au client i ;
- une capacité maximale $C \in \mathbb{N}$ correspondant au nombre maximal de vaccins qu'il est possible de stocker dans un véhicule. Cette valeur est identique pour tous les véhicules.

On définit une *tournée* T comme une suite de sommets $T = \{t_1, t_2, ..., t_{|T|}\}$ telle que :

- la tournée débute et termine à l'entrepôt (*i.e.*, $t_1 = t_{|T|} = 1$) ;
- la somme des demandes des clients visités n'excède pas la capacité d'un véhicule (*i.e.*, $\sum_{i=2}^{|T|-1} d_{t_i} \leq C$) ; et
- tous les clients visités sont différents (*i.e.*, $t_i \neq t_j \ \forall i, j \in \{2, ..., |T| - 1\}$ $i \neq j$).

L'objectif du problème statique consiste à déterminer un ensemble de tournées permettant de visiter l'ensemble des clients tout en minimisant la durée totale des trajets.

Précisions :

- Le nombre de véhicules n'est pas limité et n'intervient pas dans la fonction objectif.
- Chaque client doit être visité dans **exactement** 1 tournée. Ainsi, même s'il peut parfois être intéressant de visiter plusieurs fois un même client, ceci n'est pas autorisé.

Par exemple, une solution comportant les deux tournées $\{1, 2, 3, 1\}$ et $\{1, 4, 5, 6, 1\}$ ne sera donc admissible que si $d_2 + d_3 \leq C$ et $d_4 + d_5 + d_6 \leq C$. La valeur de l'objectif associée à cette solution est $t_{1,2} + t_{2,3} + t_{3,1} + t_{1,4} + t_{4,5} + t_{5,6} + t_{6,1}$.

2.2 Problème robuste

Nous souhaitons résoudre une version robuste du problème statique car il est possible que les valeurs choisies pour les temps de trajets soient sous-évaluées.

2.2.1 Incertitude sur les distances

A chaque sommet $i \in V$ est associé un entier $\hat{t}_i$. ← Donnée du problème Nous supposons que l'augmentation de la durée de trajet entre deux sommets i et j peut au maximum être $\hat{t}_i + \hat{t}_j + 2\hat{t}_i\hat{t}_j$. L'augmentation total des durées est limitée par un paramètre $T \in \mathbb{N}$.

L'ensemble des valeurs que peuvent prendre les durées est :

$$\mathcal{U} = \left\{ \{t'_{ij} = t_{ij} + \delta^1_{ij}(\hat{t}_i + \hat{t}_j) + \delta^2_{ij}\hat{t}_i\hat{t}_j\}_{ij\in A} \text{ tel que } \sum_{ij\in A} \delta^1_{ij} \leq T,\ \sum_{ij\in A} \delta^2_{ij} \leq T^2,\ \delta^1_{ij} \in [0,1],\ \delta^2_{ij} \in [0,2]\ \forall ij \in A \right\} \tag{1}$$

Remarque : Seules δ^1_{ij} et δ^2_{ij} sont des variables. En effet, t_{ij}, $\hat{t}_i$ et T sont des données du problème.

3 Travail demandé

Exercise 1 Modélisation papier

1. Proposer une modélisation du problème statique sous la forme d'un programme linéaire en nombre entiers **compact**. Il est généralement difficile d'obtenir une modélisation compacte pour les problèmes de tournées de véhicules mais le fait que le nombre de véhicules n'intervient pas dans la fonction objectif le permet ici. Pour ce faire, une possibilité est d'utiliser les inégalités dites MTZ qui sont illustrées ici sur le problème de voyageur de commerce.
2. Proposer une modélisation du problème robuste sous la forme d'un programme mixte en nombres entiers.
3. **Résolution par plans coupants et LazyCallback**
   a) Modifier le problème afin que la robustesse n'apparaisse plus dans l'expression de l'objectif mais dans les contraintes.
   b) Définir l'ensemble $\mathcal{U}^*$ que vous utiliserez initialement dans le problème maître.
   c) Exhiber le sous-problème nécessaire à la résolution du problème robuste par plans coupants (les contraintes devront être exprimées de manière explicite).
   d) Quelles sont les conditions à satisfaire pour qu'une solution du problème maître soit optimale ?
   e) Quelle est l'expression des coupes ajoutées par ce sous-problème ?
4. **Résolution par dualisation**
   a) Reformuler l'objectif du problème robuste afin d'isoler le terme faisant intervenir les variables δ^1_{ij} et δ^2_{ij}.
   b) Exhiber le problème interne lié à ces variables.
   c) Dualiser ce problème.
   d) Utiliser cette dualisation afin de présenter le problème robuste sous la forme d'un simple programme linéaire en nombres entiers.

Exercise 2 Résolution numérique

1. Implémenter la résolution du problème robuste par dualisation, par plans coupants et par *branch-and-cut*. Vous pourrez tenter d'améliorer les performances de ces méthodes en les adaptant. Vous pouvez par exemple, essayer de résoudre heuristiquement le sous-problème, d'ajouter des callbacks, de fournir des solutions initiales à CPLEX, de retirer de la symétrie, de simplifier les données en entrée, de renforcer les coupes robustes ajoutées, ...
2. Choisir une des deux options :
   - *(Choix 1)* - Implémenter une heuristique permettant de résoudre ce problème. Cette heuristique devra fournir une garantie de performances a posteriori (en utilisant par exemple la relaxation du problème dualisé).
   - *(Choix 2)* - Résoudre le problème robuste en utilisant une formulation non compacte du problème.

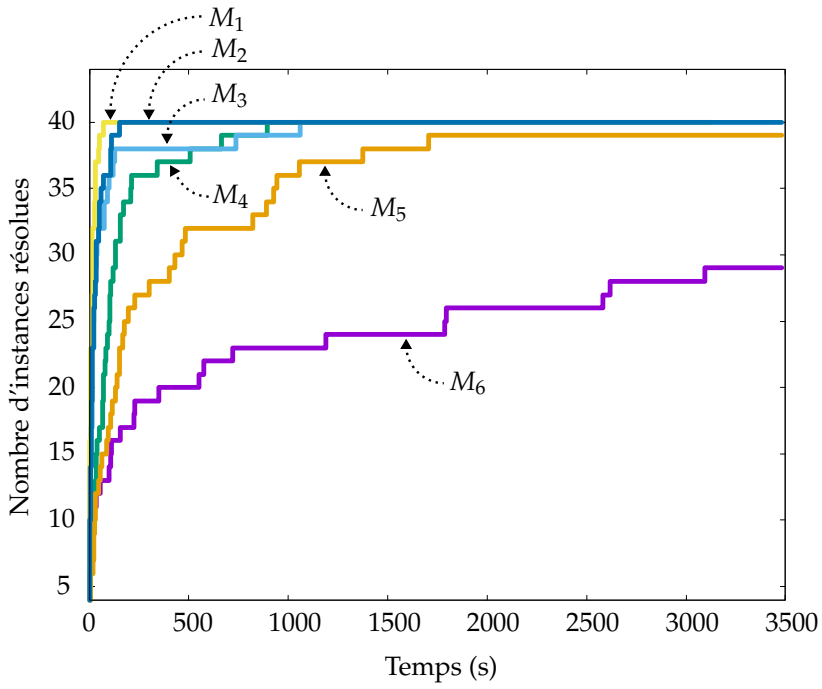


Figure 1: Exemple de diagramme de performances.

3. a) Comparer les performances de ces quatre méthodes sur les données fournies. Vous ne pourrez résoudre optimalement toutes les instances. Vous pourrez donc fixer un temps maximal et calculer le gap entre la meilleure solution réalisable obtenue (si vous en avez obtenu une) et la meilleure relaxation du problème. Etudier également les différences de performances entre les instances euclidiennes et non euclidiennes.

   b) Présenter vos résultats sous la forme d'un tableau tel que le suivant (la colonne *PR* correspond au prix de la robustesse qui est le gap entre la valeur de l'objectif d'une solution optimale robuste et d'une solution optimale statique) :

Instance	**PR**	**Plans coupants**		**Branch-and-cut**		**Dualisation**		**Heuristique**	
		Time	**Gap**	**Time**	**Gap**	**Time**	**Gap**	**Time**	**Gap**
instance_1	8%	10s	0%	10s	0%	10s	0%	10s	0%
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

   c) La lecture de tableaux de ce type pouvant parfois être fastidieuses, nous allons également représenter les résultats sous la forme de diagrammes de performances. Un diagramme de performances est une représentation graphique des performances de différentes méthodes sur les mêmes instances d'un même problème. Il représente pour chaque méthode le nombre d'instance résolues en fonction du temps. Par exemple, la Figure 1 permet de comparer aisément les performances de 6 méthodes.

4 Echéancier et modalités

Ce projet s'effectue en binôme.

- **Création du dépôt git (avant le 19 novembre)**

  Ce dépôt devra contenir un fichier README.md et les deux membres du binôme devront chacun au moins avoir réalisé un commit.

  Si vous utilisez un dépôt github privé, il faut également nous y inviter en utilisant ces adresses email : zacharie.ales@gmail.com et daniel.porumbel@cnam.fr.

- **Modélisation papier (à déposer le 15 décembre 2024 sur git)**

  Une modélisation vous sera fournie par mail quelques jours après la date de rendu (afin de vous permettre de corriger d'éventuelles erreurs de modélisation).

- **Rapport (à déposer le 15 février 2025 sur git)**

  Ce rapport devra présenter les algorithmes proposés ainsi que les résultats expérimentaux obtenus. Vous mentionnerez en annexe une solution optimale obtenue pour chaque instance résolue ainsi que la valeur de son objectif.

- **Soutenances (14 février 2025)**

  L'évaluation se fera d'une part sur la justification et l'analyse des méthodes de résolution proposées pour le problème étudié, et d'autre part sur la qualité effective de ces algorithmes de résolution.