RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

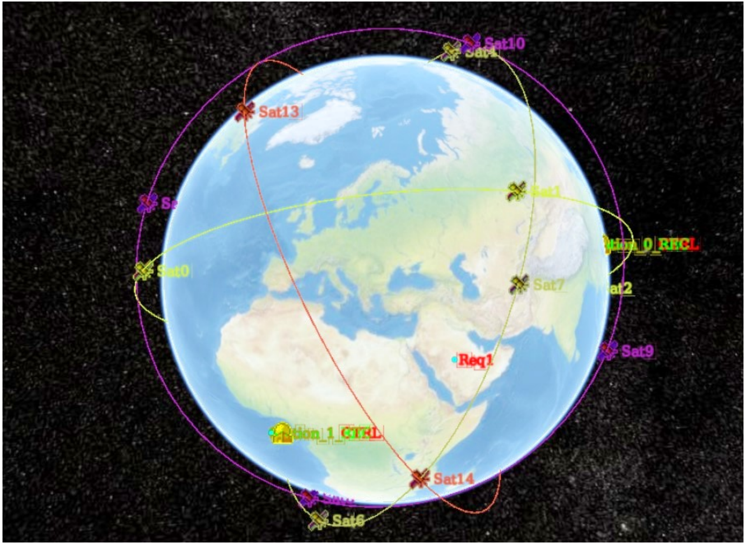
ONERA
THE FRENCH AEROSPACE LAB

Techniques d'allocation de lots avec des préférences conflictuelles représentées par des graphes acycliques dirigés pondérés

Sara Maqrot Stéphanie Roussel ***Gauthier Picard*** Cédric Pralet
ONERA/DTIS, Université de Toulouse, France

Conférence Nationale en Intelligence Artificielle (CNIA), 27/07/2022

Introduction
Motivation : constellations de satellites d'observation de la Terre



Introduction

Motivation : constellations de satellites d'observation de la Terre

- **Problème :** exploitation d'une même constellation par plusieurs utilisateurs

Réservation hors-ligne
Portions d'orbite systématiques — périodes d'exclusivité → **Planification en ligne**
Acquisition d'images

- **Concept actuel d'attribution :** premier arrivé, premier servi
- **Objectif**

Préférences (POI, horaires tolérance, ...) → **Portions candidates** (objets à partager) → **Partage utilitariste** ou **équitable**

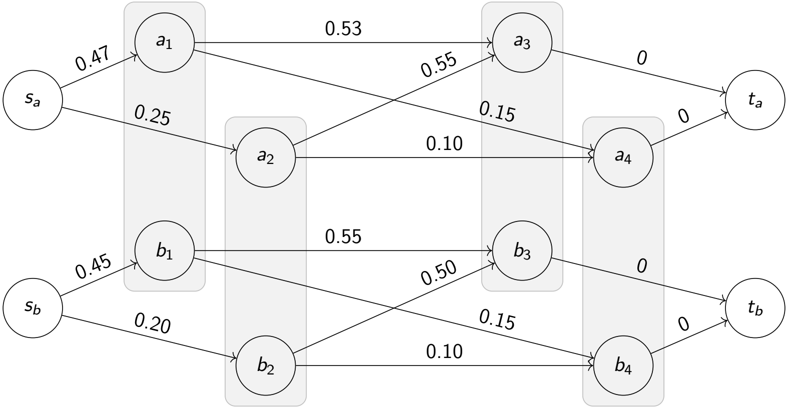
Introduction
Exemple de problème d'allocation de portions d'orbite

- 2 agents (a en rouge, b en bleu) demandant des acquisitions :
  - de points d'intérêt (POI) autour de la même zone
  - autour de 2 plots temporels ($8h$ et $12h$) tous les jours
    avec une tolérance avant et après chaque plot (en gris)
- 1 satellite donnant accès à 2 portions candidates pour chaque plot ($a_1, \ldots, a_4, b_1, \ldots, b_4$)

Menu du jour

Menu du jour

Modèle du problème
Exemple de problème d'allocation de portions d'orbite



Modèle du problème
Définitions et notations

Définition (PADAG)

Un problème d'*allocation de multiple chemins conflictuels dans des graphes acycliques dirigés avec arêtes pondérées* (PADAG) est un tuple $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$, où :

- $\mathcal{A} = \{1, \ldots, n\}$ un ensemble d'*agents*
- $\mathcal{G} = \{g_1, \ldots, g_m\}$ un ensemble de DAGs (préférences) où chaque $g = \langle V_g, E_g, u_g \rangle$
- $\mu : \mathcal{G} \rightarrow \mathcal{A}$ fait correspondre chaque graphe g dans $\mathcal{G}$ à son propriétaire a dans $\mathcal{A}$
- $\mathcal{C} \subset \{(v, v') | (v, v') \in V_g \times V_{g'}, g, g' \in \mathcal{G}^2, \mu(g) \neq \mu(g')\}$ un ensemble de conflits entre des paires d'éléments de deux graphes distincts de $\mathcal{G}$ provenant de deux agents distincts

Modèle du problème (cont.)

Définitions et notations

Définition (Allocation)

Une *allocation* est une fonction π qui associe, à chaque graphe $g \in \mathcal{G}$, un chemin $\pi(g)$ de s_g à t_g dans g
Par extension, l'allocation pour l'agent a est donnée par $\pi(a) = \cup_{g \in \mu^{-1}(a)} \pi(g)$

Définition (Allocation valide)

Une allocation π est *valide* si pour chaque paire de graphes distincts g et g' il n'y a pas de conflit entre les nœuds dans les chemins résultants, c'est-à-dire

$$(\pi(g) \times \pi(g')) \cap \mathcal{C} = \emptyset$$

Modèle du problème (cont.)

Définitions et notations

Quels objectifs pour un PADAG ?

- Utilitariste : maximiser la somme des utilités des chemins affectés
- Equitable (leximin) : maximiser lexicographiquement le vecteur d'utilité ordonné

Modèle du problème (cont.)
Définitions et notations

Théorème (NP-complétude du problème utilitariste)

Déterminer s'il existe une allocation valide π telle que l'**évaluation utilitaire** $u(\pi)$ est supérieure ou égale à une valeur donnée est NP-complet

(preuve : réduction polynomiale de 3-SAT (qui est NP-complet) à PADAG utilitariste)

Théorème (NP-complétude du problème leximin)

Il est NP-complet de décider s'il existe une allocation valide dont l'**évaluation leximin** est supérieure ou égale à un vecteur d'utilité donné

(preuve : réduction polynomiale de 3-SAT (qui est NP-complet) à PADAG leximin avec un agent possédant tous les graphes)

Menu du jour

Comment résoudre un PADAG ?

1. Allocation utilitariste optimale (util)
2. Allocation leximin optimale (lex)
3. Allocation leximin approchée (a-lex)
4. Allocation gloutonne (greedy)
5. Allocation *round-robin* sur les chemins (p-rr)
6. Allocation *round-robin* sur les nœuds (n-rr)

Concepts communs aux approches MILP

- Variables, pour tout DAG $g = \langle V_g, E_g, u_g \rangle$
  - x_e : vraie si $e \in E_g$ est présente dans le chemin solution $\pi(g)$
  - β_v : vraie si $v \in V_g$ est présent dans le chemin solution $\pi(g)$
- Contraintes

pour définir tous les chemins possibles

$$\sum_{e \in \mathrm{In}(v)} x_e = \sum_{e \in \mathrm{Out}(v)} x_e, \quad \forall g \in \mathcal{G}, \forall v \in V_g \setminus \{s_g, t_g\} \tag{1}$$

$$\sum_{e \in \mathrm{Out}(s_g)} x_e = 1, \quad \forall g \in \mathcal{G} \tag{2}$$

$$\sum_{e \in \mathrm{In}(t_g)} x_e = 1, \quad \forall g \in \mathcal{G} \tag{3}$$

pour respecter les conflits entre objets

$$\sum_{e \in \mathrm{In}(v)} x_e = \beta_v, \quad \forall g \in \mathcal{G}, \forall v \in V_g \setminus \{s_g, t_g\} \tag{4}$$

$$\sum_{v \in c} \beta_v \leq 1, \quad \forall c \in \mathcal{C} \tag{5}$$

pour assurer des chemins de sources à destination

$$\beta_{s_g} = \beta_{t_g} = 1, \quad \forall g \in \mathcal{G} \tag{6}$$

Allocation utilitariste (util)

$$P_{\text{util}}(\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle) \qquad \text{maximiser} \quad \sum_{a \in \mathcal{A}} \sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) \cdot x_e \tag{7}$$

$$\text{t.q.} \quad (1), (2), (3), (4), (5), (6)$$

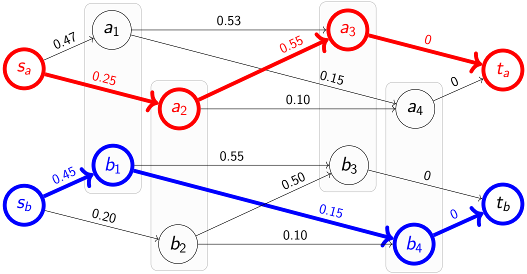
$$x_e \in \{0, 1\}, \quad \forall a \in \mathcal{A}, \forall g \in \mathcal{G}_a, \forall e \in E_g \tag{8}$$

Allocation utilitariste (util)

$$P_{\text{util}}(\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle) \qquad \text{maximiser} \quad \sum_{a \in \mathcal{A}} \sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) \cdot x_e \tag{7}$$

$$\text{t.q.} \quad (1), (2), (3), (4), (5), (6)$$

$$x_e \in \{0, 1\}, \quad \forall a \in \mathcal{A}, \forall g \in \mathcal{G}_a, \forall e \in E_g \tag{8}$$



$$u(\pi_{\text{util}}) = u(\{a \mapsto \{s_a, a_2, a_3, t_a\}) + u(b \mapsto \{s_b, b_1, b_4, t_b\}) = 0.80 + 0.60 = 1.40$$

Allocation leximin optimale (lex)
Schéma général

1. Maximiser la pire utilité
$$\Lambda_1 = 0.62$$
2. Maximiser la seconde pire utilité sachant que la pire utilité est 0.62
$$\Lambda_2 = 0.70$$
3. Maximiser la pire utilité sachant que les pires utilités sont 0.62 et 0.70

$\ldots$

Allocation leximin optimale (lex)
Schéma général

1. Maximiser la pire utilité

$$\Lambda_1 = 0.62$$

2. Maximiser la seconde pire utilité sachant que la pire utilité est 0.62

$$\Lambda_2 = 0.70$$

3. Maximiser la pire utilité sachant que les pires utilités sont 0.62 et 0.70

$\ldots$

Remarques

- Chaque étape nécessite la résolution d'un MILP
- Autant d'étapes que d'agents
- Ce sont les niveaux d'utilité qui sont affectés, pas les agents

Allocation leximin optimale (lex)

Algorithme

Algorithme 1 : Algorithme leximin (lex)

Données : Un PADAG $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$
Résultat : Une allocation leximin-optimale π
pour $K = 1$ à $|\mathcal{A}|$ **faire**
  $(\lambda^*, sol) \leftarrow P_{\text{lex}}(\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle, K, [\Lambda_1, \ldots, \Lambda_{K-1}])$
  $\Lambda_K \leftarrow \lambda^*$
pour $g \in \mathcal{G}$ **faire**
  $\pi(g) \leftarrow \{v \in V_g \mid sol(\beta_v) = 1\}$
retourner π

$$\text{maximiser} \quad \lambda \tag{9}$$

$$\text{t.q.} \quad (1), (2), (3), (4), (5), (6)$$

$$z_a = \sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) \cdot x_e, \quad \forall a \in \mathcal{A} \tag{10}$$

$$\sum_{a \in \mathcal{A}} y_{ak} = 1, \quad \forall k \in [1..K-1] \tag{11}$$

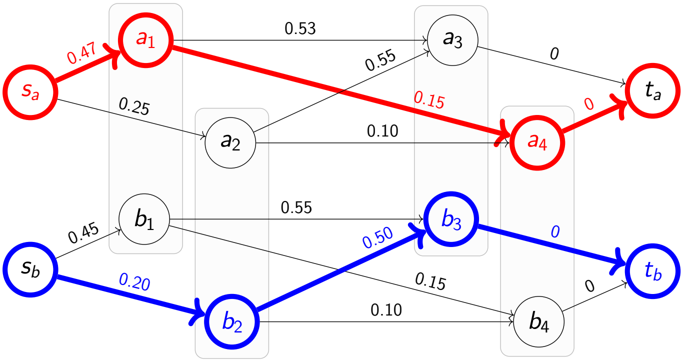
$$\sum_{k \in [1..K-1]} y_{ak} \leq 1, \quad \forall a \in \mathcal{A} \tag{12}$$

$$\lambda \leq z_a + M \sum_{k \in [1..K-1]} y_{ak}, \quad \forall a \in \mathcal{A} \tag{13}$$

$$z_a \geq \sum_{k \in [1..K-1]} \Lambda_k \cdot y_{ak}, \quad \forall a \in \mathcal{A} \tag{14}$$

avec $\Lambda = [\Lambda_1, \ldots, \Lambda_n]$ le vecteur des utilités par ordre croissant

Allocation leximin optimale (lex)
Exemple



$$u(\pi_{\text{lexi}}) = u(\{a \mapsto \{s_a, a_1, a_4, t_a\}\}) + u(b \mapsto \{s_b, b_2, b_3, t_b\}) = 0.62 + 0.70 = 1.32.$$

Allocation leximin approchée (a-lex)
Schéma général

1. Maximiser la pire utilité et choisir l'agent associé

$$\cancel{A_1 = 0.62} \qquad u_a = 0.62$$

2. Maximiser la seconde pire utilité sachant que l'utilité de l'agent a est 0.62

$$\cancel{A_2 = 0.70} \qquad u_b = 0.70$$

3. Maximiser la pire utilité sachant que l'utilité de l'agent a est 0.62 et l'utilité de b est 0.70

...

Allocation leximin approchée (a-lex)

Schéma général

1. Maximiser la pire utilité et choisir l'agent associé

$$\cancel{A_1 = 0.62} \qquad u_a = 0.62$$

2. Maximiser la seconde pire utilité sachant que l'utilité de l'agent a est 0.62

$$\cancel{A_2 = 0.70} \qquad u_b = 0.70$$

3. Maximiser la pire utilité sachant que l'utilité de l'agent a est 0.62 et l'utilité de b est 0.70

...

Remarques

- Chaque étape nécessite la résolution d'un MILP de plus en plus en plus réduit
- Autant d'étapes que d'agents
- Ce sont les agents qui sont affectés, pas les utilités
- Si pas d'égalités de pires scores, alors a-lex $\equiv$ lex

Allocation leximin approchée (a-lex)
Algorithme

Algorithme 2 : Algorithme leximin approché (a-lex)

Données : Un PADAG $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$
Résultat : Une allocation maximin-optimale π
$\Delta \leftarrow [-1, \ldots, -1]$
pour $K = 1$ à $|\mathcal{A}|$ **faire**
  $(\delta^*, sol) \leftarrow P_{\text{a-lex}}(\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle, \Delta)$
  $S \leftarrow \operatorname*{argmin}_{a \in \mathcal{A} \mid \Delta_a = -1} \sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) sol(x_e)$
  $\hat{a} \leftarrow$ choisir un agent $a \in S$
  $\Delta_{\hat{a}} \leftarrow \delta^*$
pour $g \in \mathcal{G}$ **faire**
  $\pi(g) \leftarrow \{v \in V_g \mid sol(\beta_v) = 1\}$
retourner π

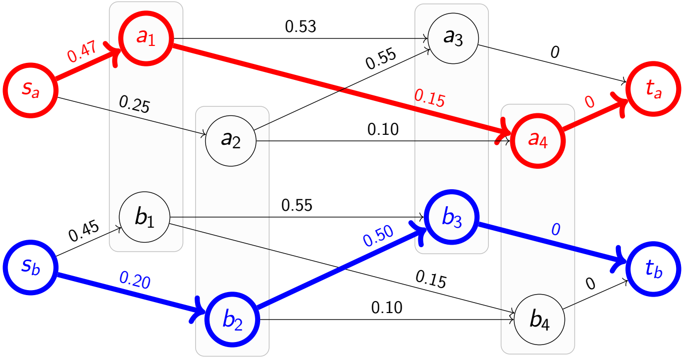
$$\text{maximiser} \quad \delta \tag{15}$$

$$\text{t.q} \quad (1), (2), (3), (4), (5), (6)$$

$$\delta \leq \sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) x_e, \quad \forall a \in \mathcal{A} \mid \Delta_a = -1 \tag{16}$$

$$\sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) x_e \geq \Delta_a, \quad \forall a \in \mathcal{A} \mid \Delta_a \neq -1 \tag{17}$$

Allocation leximin approchée (a-lex)
Exemple



$$u(\pi_{\text{approx}}) = u(a \mapsto \{s_a, a_1, a_4, t_a\}) + u(b \mapsto \{s_b, b_2, b_3\}) = 0.62 + 0.70 = 1.32.$$

Allocation gloutonne (greedy)
Approche utilitariste rapide (polynomiale)

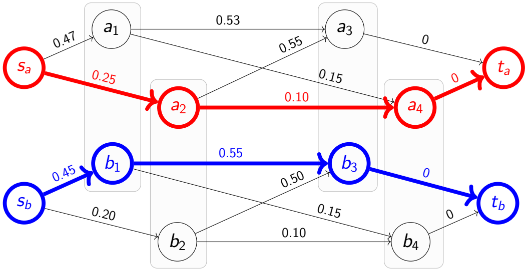
Algorithme 3 : Algorithme glouton (greedy)

Données : Un PADAG $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$
Résultat : Une allocation π
tant que $\mathcal{G} \neq \emptyset$ **faire**
    déterminer g^* le graphe d'utilité maximale avec le chemin p
    $\pi(g^*) \leftarrow p$
    pour $g \in \mathcal{G}$ **faire**
        $V_g \leftarrow \{v \in V_g \mid \forall w \in \pi(g^*), \{v, w\} \notin \mathcal{C}\}$
        $E_g \leftarrow \{(v, w) \in E_g \mid v \in V_g, w \in V_g\}$
    $\mathcal{G} \leftarrow \mathcal{G} \setminus \{g^*\}$
retourner π

Allocation gloutonne (greedy)
Approche utilitariste rapide (polynomiale)

Algorithme 4 : Algorithme glouton (greedy)

Données : Un PADAG $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$
Résultat : Une allocation π
tant que $\mathcal{G} \neq \emptyset$ **faire**
  déterminer g^* le graphe d'utilité maximale avec le chemin p
  $\pi(g^*) \leftarrow p$
  pour $g \in \mathcal{G}$ **faire**
    $V_g \leftarrow \{v \in V_g \mid \forall w \in \pi(g^*), \{v, w\} \notin \mathcal{C}\}$
    $E_g \leftarrow \{(v, w) \in E_g \mid v \in V_g, w \in V_g\}$
  $\mathcal{G} \leftarrow \mathcal{G} \setminus \{g^*\}$
retourner π



$$u(\pi_{\text{greedy}}) = u(a \mapsto \{s_a, a_3, a_4, t_a\}) + u(b \mapsto \{s_b, b_1, b_3, t_b\}) = 0.35 + 1.0 = 1.35$$

Allocations *round-robin* (p-rr et n-rr)
Approche égalitaristes rapides (polynomiales)

Principe

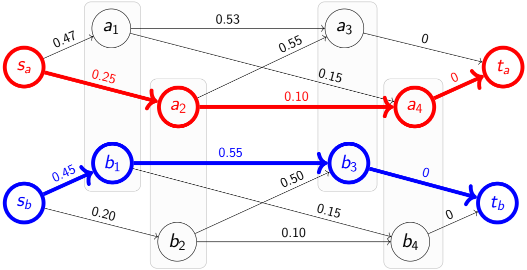
- Les agents choisissent un objet à tour de rôle
- p-rr : les objets sont des chemins
- n-rr : les objets sont des nœuds
- p-rr résulte sur un équilibre de Nash

Allocations *round-robin* (p-rr et n-rr)

Approche égalitaristes rapides (polynomiales)

Principe

- Les agents choisissent un objet à tour de rôle
- p-rr : les objets sont des chemins
- n-rr : les objets sont des nœuds
- p-rr résulte sur un équilibre de Nash



$$u(\pi_{\text{p-rr}}) = u(\pi_{\text{n-rr}}) = u(a \mapsto \{s_a, a_3, a_4, t_a\}) + u(b \mapsto \{s_b, b_1, b_3, t_b\}) = 0.35 + 1.0 = 1.35$$

Menu du jour

Cadre expérimental

Constellation orbite basse (500km altitude)

- $n_p \in \{2, 4, 8, 16\}$ plans orbitaux répartis avec inclinaison de 60 degrés
- 2 satellites opposés par plan orbital

Agents (4 utilisateurs)

- 2 requêtes par agent
  - position : POIs dans la même région (France)
  - plots : tous les jours à $8:00+\delta_r$, $12:00+\delta_r$, et $16:00+\delta_r$, $\delta_r \in [-2, 2]$
  - avec une tolérance d'une heure
- horizon de 365 jours (1095 couches/plots temporels)

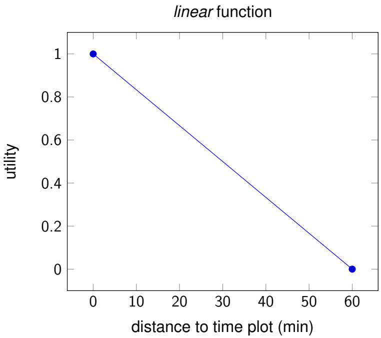
n_p	2	4	8	16
largeur	3.08	5.41	10.05	19.38
conflits	26798.80	45636.06	82971.20	158180.20
durée moyenne (s)	603.28	600.10	599.87	598.75

Cadre expérimental (cont.)

Utilités

- Utilités attachées aux portions et non aux transitions en portions

- Même fonction pour tous les utilisateurs : linéaire en la distance entre le milieu du créneau et le plot demandé

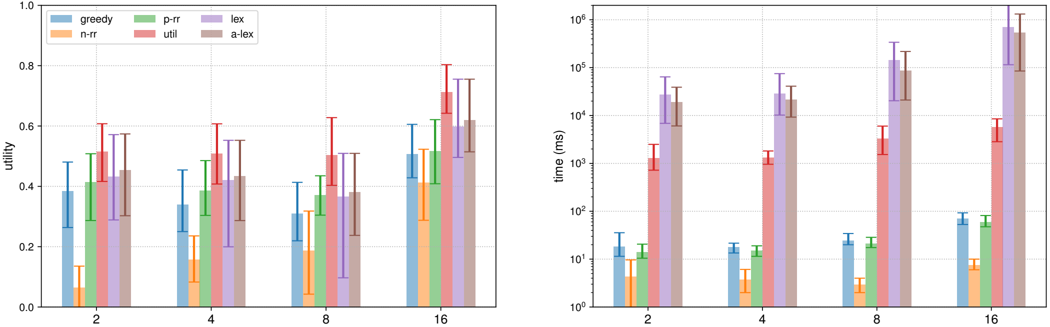


Cadre expérimental (cont.)

Logiciels et matériel

- solveurs codés en Java, et API Java de IBM CPLEX 20.1 pour util, lex et a-lex
- CPU Intel(R) Xeon(R) E5-2660 v3 @ 2.60GHz à 20 cœurs, 62GB RAM, Ubuntu 18.04.5 LTS
- 30 instances de PADAG générés aléatoirement par configuration

Résultats
Utilité et temps de calcul

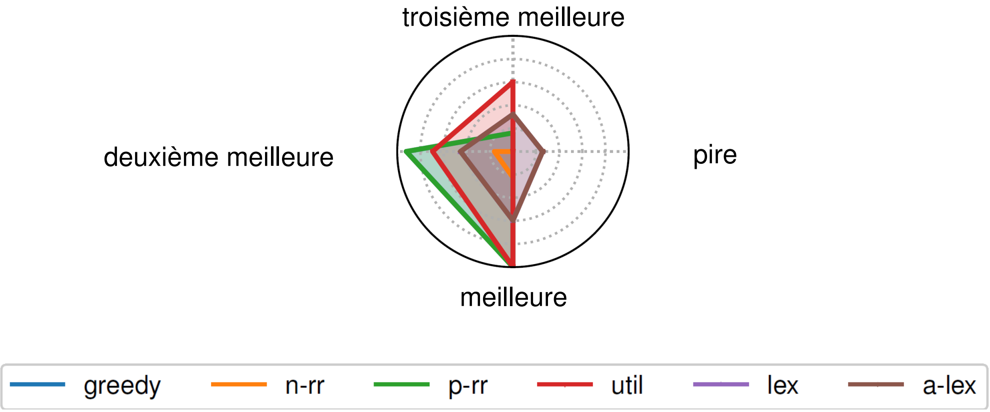


- a-lex fournit des allocations à presque 85% de la valeur optimale en moyenne
- lex a des performances équivalentes à a-lex (écart $<$ 5% en moyenne)
- p-rr fournit des allocations à presque 71% de l'optimal
- n-rr donne des allocations de très faible utilité
- greedy se comporte légèrement moins bien que p-rr

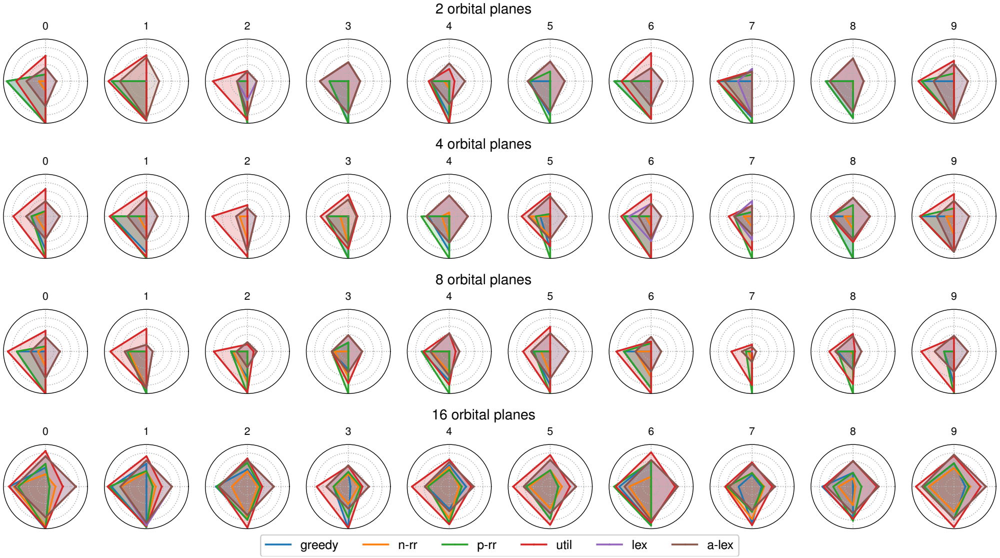
Résultats (cont.)
Utilité et temps de calcul

⇒ Les problèmes de constellation de grande taille sont plus faciles à résoudre du point de vue utilitaire par les algorithmes non optimaux, puisqu'il existe plus d'options pour éviter les conflits malgré leur nombre élevé

Résultats
Profils d'utilité (ordre leximin)



Résultats
Equité (10 premières instances sur 30)



Résultats (cont.)
Equité (10 premières instances sur 30)

- greedy alloue injustement aux 2 ou 3 premiers utilisateurs
- les *round-robin* sont plus équitables, mais peinent sur les grandes constellations
- util donne des profils avec la plus grande surface mais quatrième utilisateur négligé
- lex et a-lex se comportent presque identiquement

Menu du jour

Conclusions

- Première approche pour d'allocation de lots avec des préférences conflictuelles représentées par des graphes acycliques dirigés pondérés (PADAG)
- Plusieurs méthodes de résolution explorées
  - util, lex, a-lex, greedy, p-rr, n-rr
- Expérimentations sur cas d'allocation de portions d'orbite
- a-lex est le meilleur compromis entre utilitarisme, équité et temps de calcul

Perspectives

- Travaux en cours
  - Considérer d'autres types de requêtes et donc de graphes
    - requêtes répétitives
    - requêtes systématiques
    - requêtes hétérogènes
  - Considérer d'autres méthodes pour ces graphes spécifiques
    - Programmation par contraintes (PAIS'22)
    - Recherche locale et Min-conflict
- Travaux futurs
  - Considérer des aires (AOI à la place de POI)
  - Considérer des applications différentes (e.g. NFV)

Merci pour votre attention !

www.onera.fr