RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

ONERA
THE FRENCH AEROSPACE LAB

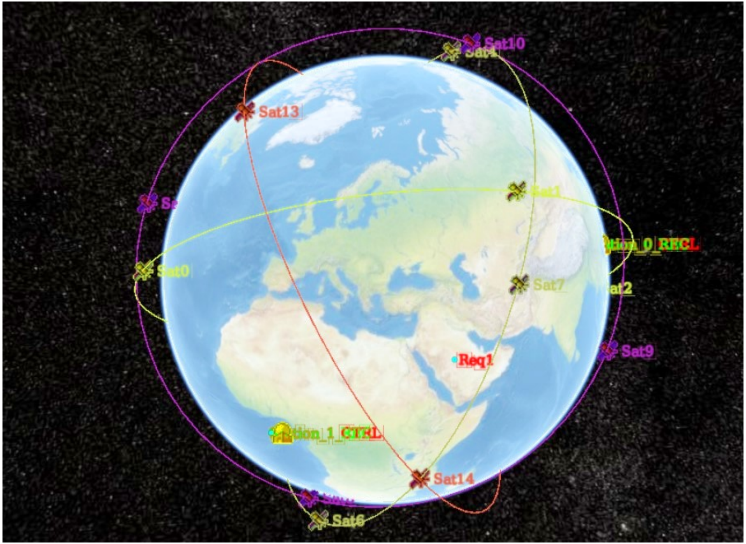
Bundle allocation with conflicting preferences represented as weighted directed acyclic graphs

Application to orbit slot ownership

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

20th International Conference on Practical Applications of Agents and Multi-Agent Systems, 15/07/2022

Introduction
Motivation: Earth observation satellite constellations



Introduction
Motivation: Earth observation satellite constellations

- **Problem :** exploitation of the same constellation by several stakeholders

Offline reservation systematic orbit slots —— Exclusivity periods ⟶ **Online planning** Image acquisition

- **Current allocation scheme:** first come, first served
- **Objective**

Preferences (POI, dates tolerance, ...) ⟶ **Candidate slots** (Objects to allocate) ⟶ **Utilitarian** or **fair allocation**

Introduction
Sample orbit slot allocation problem

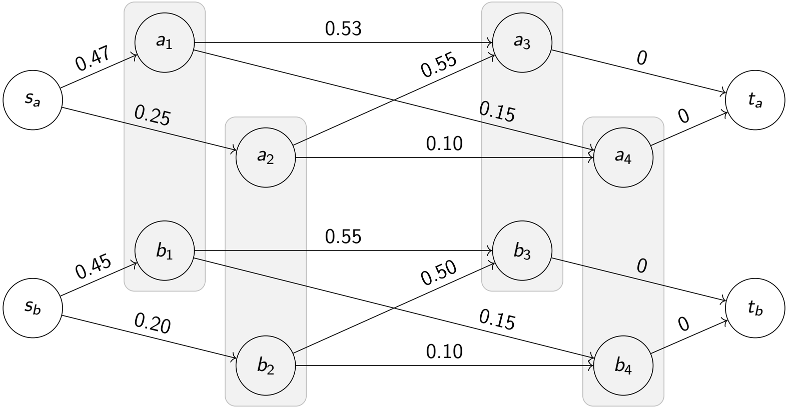
- 2 agents (a in red, b in blue) requesting acquisitions:
  - of points of interest (POI) around the same region
  - around 2 time points ($8h$ and $12h$) every day
    with a tolerance before and after each time point (in gray)
- 1 satellite giving access to 2 orbit slots for each time point
  $(a_1, \ldots, a_4, b_1, \ldots, b_4)$

Today's Menu

Today's Menu

Problem Model

Sample orbit slot allocation problem



Problem Model
Definitions and notations

Definition (PADAG)

A problem of *path allocation in multiple conflicting edge-weighted directed acyclic graphs* (PADAG) is a tuple $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$, where

- $\mathcal{A} = \{1, \ldots, n\}$ a set of *agents*
- $\mathcal{G} = \{g_1, \ldots, g_m\}$ a set of DAGs (preferences) where each $g \in \mathcal{G}$
- $\mu : \mathcal{G} \to \mathcal{A}$ maps each graph $g \in \mathcal{G}$ to its owner $a \in \mathcal{A}$
- $\mathcal{C} \subseteq \{(v, v') | (v, v') \in V_g \times V_{g'}, g, g' \in \mathcal{G}^2, \mu(g) \neq \mu(g')\}$ is a set of conflicts between pairs of items

Problem Model (cont.)
Definitions and notations

Definition (Allocation)

An *allocation* is a function π that associates, with each graph $g \in \mathcal{G}$, one path $\pi(g)$ from s_g to t_g in g

By extension, the allocation for agent a is given by $\pi(a) = \cup_{g \in \mu^{-1}(a)} \pi(g)$

Definition (Valid allocation)

An allocation π is *valid* if for each pair of distinct graphs g and g' there is no conflict between nodes in the resulting paths, i.e. $(\pi(g) \times \pi(g')) \cap \mathcal{C} = \emptyset$

Problem Model (cont.)
Definitions and notations

Which objectives for a PADAG?

- Utilitarian: maximize the sum of allocated path utilities
- Fair (leximin): maximize lexicographically sorted vector of agent utilities

Problem Model (cont.)

Definitions and notations

Theorem (NP-completeness of utilitarian PADAG)

Determining whether there exists a valid allocation π such that utilitarian evaluation $u(\pi)$ is greater than or equal to a given value is NP-complete

(proof sketch: polynomial reduction of 3-SAT (which is NP-complete) to PADAG)

Theorem (NP-completeness of leximin PADAG)

It is NP-complete to decide whether there exists a valid allocation whose leximin evaluation is greater than or equal to a given utility vector

(proof sketch: polynomial reduction of 3-SAT (which is NP-complete) to PADAG with a single agent owning all the graphs)

Today's Menu

How to solve PADAG?

1. Optimal utilitarian allocation (util)
2. Optimal leximin allocation (lex)
3. Approximate utilitarian allocation (a-lex)
4. Greddy allocation (greedy)
5. *round-robin* path allocation (p-rr)
6. *round-robin* node allocation (n-rr)

Common Concepts for MILP-based Methods

- Variables, for each DAG $g = \langle V_g, E_g, u_g \rangle$
  - x_e: 1 if $e \in E_g$ is in a solution path $\pi(g)$
  - β_v: 1 if $v \in V_g$ is in a solution path $\pi(g)$
- Constraints

to define all possible paths

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

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

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

to respect conflicts between items

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

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

to enforce source to destination paths

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

Utilitarian Optimal Allocation (util)

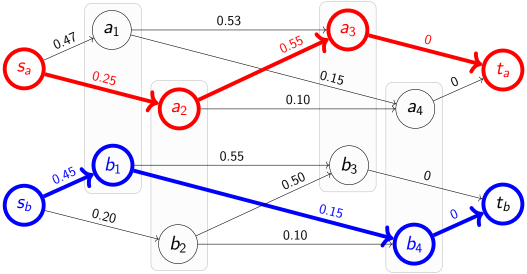
$$
\begin{aligned}
P_{\text{util}}(\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle) \qquad \text{maximize} \quad & \sum_{a \in \mathcal{A}} \sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) \cdot x_e & (7) \\
\text{s.t.} \quad & (1), (2), (3), (4), (5), (6) \\
& x_e \in \{0, 1\}, \quad \forall a \in \mathcal{A}, \forall g \in \mathcal{G}_a, \forall e \in E_g & (8)
\end{aligned}
$$

Utilitarian Optimal Allocation (util)

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

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

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



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

Leximin Optimal Allocation (lex)
General Scheme

1. Maximize worst utility

$$\Lambda_1 = 0.62$$

2. Maximize second worst utility given the worst utility is 0.62

$$\Lambda_2 = 0.70$$

3. Maximize the third worst utility given the first and second worst utilities are0.62 and 0.70

…

Leximin Optimal Allocation (lex)

General Scheme

1. Maximize worst utility

$$\Lambda_1 = 0.62$$

2. Maximize second worst utility given the worst utility is 0.62

$$\Lambda_2 = 0.70$$

3. Maximize the third worst utility given the first and second worst utilities are 0.62 and 0.70

$\ldots$

Remarks

- Each step requires solving a MILP
- As many steps as agents
- These are utility levels that are assigned, not the agents

Algorithm 1: Leximin Optimal Allocation (lex)

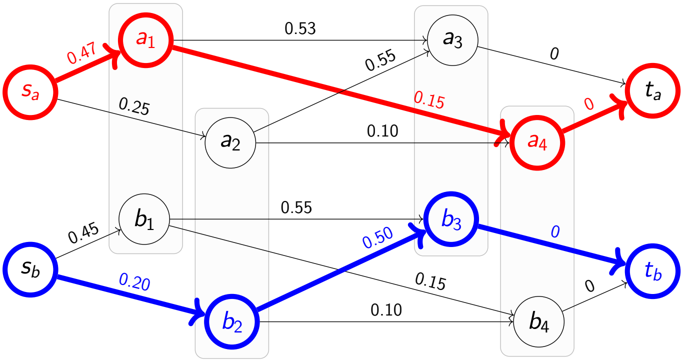
Data: A PADAG $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$
Result: A leximin optimal allocation π
for $K = 1$ *à* $|\mathcal{A}|$ **do**
  $(\lambda^*, sol) \leftarrow P_{\text{lex}}(\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle, K, [\Lambda_1, \ldots, \Lambda_{K-1}])$
  $\Lambda_K \leftarrow \lambda^*$
for $g \in \mathcal{G}$ **do**
  $\pi(g) \leftarrow \{v \in V_g \mid sol(\beta_v) = 1\}$
return π

$$
\begin{aligned}
\text{maximize} \quad & \lambda && (9)\\
\text{s.t.} \quad & (1), (2), (3), (4), (5), (6) && \\
& z_a = \sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) \cdot x_e, \quad \forall a \in \mathcal{A} && (10)\\
& \sum_{a \in \mathcal{A}} y_{ak} = 1, \quad \forall k \in [1..K-1] && (11)\\
& \sum_{k \in [1..K-1]} y_{ak} \leq 1, \quad \forall a \in \mathcal{A} && (12)\\
& \lambda \leq z_a + M \sum_{k \in [1..K-1]} y_{ak}, \quad \forall a \in \mathcal{A} && (13)\\
& z_a \geq \sum_{k \in [1..K-1]} \Lambda_k \cdot y_{ak}, \quad \forall a \in \mathcal{A} && (14)
\end{aligned}
$$

with $\Lambda = [\Lambda_1, \ldots, \Lambda_n]$ the utility profile ordered by increasing utility level

Leximin Optimal Allocation (lex)

Example



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

Approximate Leximin Allocation (a-lex)
General Scheme

1. Maximize worst utility and choose the associated agent

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

2. Maximize the second worst utility given agent a's utility is 0.62

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

3. Maximize the third worst utility given agent a's utility is 0.62 and b's utility is 0.70

   ...

Approximate Leximin Allocation (a-lex)

General Scheme

1. Maximize worst utility and choose the associated agent

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

2. Maximize the second worst utility given agent a's utility is 0.62

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

3. Maximize the third worst utility given agent a's utility is 0.62 and b's utility is 0.70

$\dots$

Remarks

- Each step requires solving a smaller MILP
- As many steps as agents
- These are agents which are assigned, not utility levels
- Without utility equivalences, a-lex $\equiv$ lex

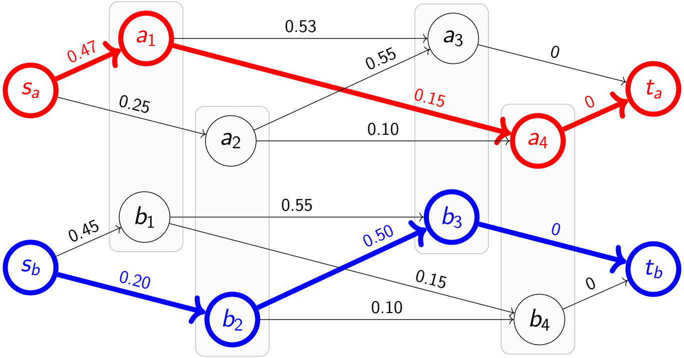
Approximate Leximin Allocation (a-lex)
Algorithm

Algorithm 2: Approximate Leximin Allocation (a-lex)

Data: A PADAG $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$
Result: An approximate leximin allocation π
$\Delta \leftarrow [-1, \ldots, -1]$
for $K = 1$ *à* $|\mathcal{A}|$ **do**
  $(\delta^*, sol) \leftarrow P_{\text{a-lex}}(\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle, \Delta)$
  $S \leftarrow \operatorname{argmin}_{a \in \mathcal{A} \mid \Delta_a = -1} \sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) sol(x_e)$
  $\hat{a} \leftarrow$ choose an agent $a \in S$
  $\Delta_{\hat{a}} \leftarrow \delta^*$
for $g \in \mathcal{G}$ **do**
  $\pi(g) \leftarrow \{v \in V_g \mid sol(\beta_v) = 1\}$
return π

$$\text{maximize} \quad \delta \tag{15}$$
$$\text{t.q} \quad (1), (2), (3), (4), (5), (6)$$
$$\delta \leq \sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) x_e, \quad \forall a \in \mathcal{A} \mid \Delta_a = -1 \tag{16}$$
$$\sum_{g \in \mathcal{G}_a} \sum_{e \in E_g} u_g(e) x_e \geq \Delta_a, \quad \forall a \in \mathcal{A} \mid \Delta_a \neq -1 \tag{17}$$

Approximate Leximin Allocation (a-lex)
Example



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

Greedy Allocation (greedy)
Fast (polynomial) utilitarian approach

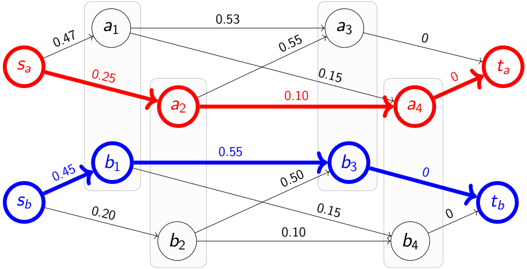
Algorithm 3: Greedy Allocation (greedy)

Data: A PADAG $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$
Result: An allocation π
while $\mathcal{G} \neq \emptyset$ **do**
    determine max utility graph g^* with best path p
    $\pi(g^*) \leftarrow p$
    for $g \in \mathcal{G}$ **do**
        $V_g \leftarrow \{v \in V_g \mid \forall w \in \pi(g^*), \{v, w\} \notin \mathcal{C}\}$
        $E_g \leftarrow \{(v, w) \in E_g \mid v \in V_g, w \in V_g\}$
    $\mathcal{G} \leftarrow \mathcal{G} \setminus \{g^*\}$
return π

Greedy Allocation (greedy)
Fast (polynomial) utilitarian approach

Algorithm 4: Greedy Allocation (greedy)

Data: A PADAG $\langle \mathcal{A}, \mathcal{G}, \mu, \mathcal{C} \rangle$
Result: An allocation π
while $\mathcal{G} \neq \emptyset$ **do**
  determine max utility graph g^* with best path p
  $\pi(g^*) \leftarrow p$
  for $g \in \mathcal{G}$ **do**
    $V_g \leftarrow \{v \in V_g \mid \forall w \in \pi(g^*), \{v, w\} \notin \mathcal{C}\}$
    $E_g \leftarrow \{(v, w) \in E_g \mid v \in V_g, w \in V_g\}$
  $\mathcal{G} \leftarrow \mathcal{G} \setminus \{g^*\}$
return π



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

Round-robin Allocations (p-rr and n-rr)

Fast (polynomial) fair approaches

Principle

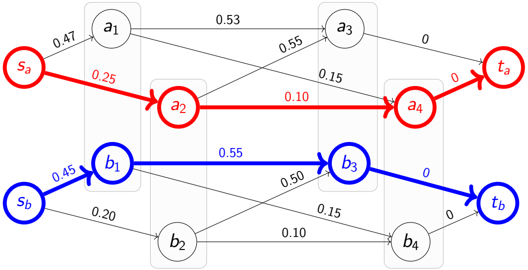
- Agents choose items in turn
- p-rr: items are paths
- n-rr: items are nodes
- p-rr reaches a Nash equilibrium

Round-robin Allocations (p-rr and n-rr)

Fast (polynomial) fair approaches

Principle

- Agents choose items in turn
- p-rr: items are paths
- n-rr: items are nodes
- p-rr reaches a Nash equilibrium



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

Today's Menu

Experimental Setup

Low Earth orbit constellation (500km altitude)

- $n_p \in \{2, 4, 8, 16\}$ regularly-spaced orbital planes having a 60-degree inclination
- 2 anti-phase satellites per plane

Agents (4 users)

- 2 requests per agent
  - position: POIs in the same region (France)
  - time points : every day at $8:00+\delta_r$, $12:00+\delta_r$, et $16:00+\delta_r$, $\delta_r \in [-2, 2]$
  - with a 1h-tolerance
- time horizon: 365 days (1095 layers in DAGs)

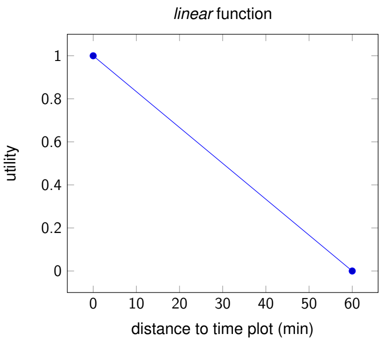
n_p	2	4	8	16
width	3.08	5.41	10.05	19.38
conflicts	26798.80	45636.06	82971.20	158180.20
mean duration (s)	603.28	600.10	599.87	598.75

Experimental Setup (cont.)

Utilities

- Utilities only attached to slots and not to transitions between slots

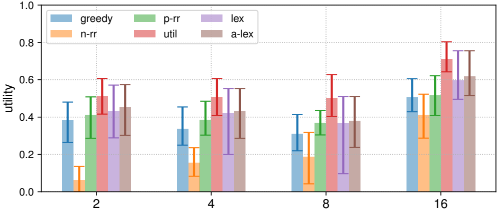
- Same function for all users: linear with the distance to the center of the requested time point

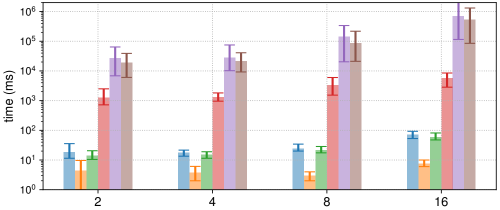


Experimental environment

- solvers implemented in Java, and IBM CPLEX 20.1 Java API for util, lex and a-lex
- CPU Intel(R) Xeon(R) E5-2660 v3 @ 2.60GHz with 20 cores, 62GB RAM, Ubuntu 18.04.5 LTS
- 30 randomly generated PADAG instances per configuration

Results
Utility and computation time



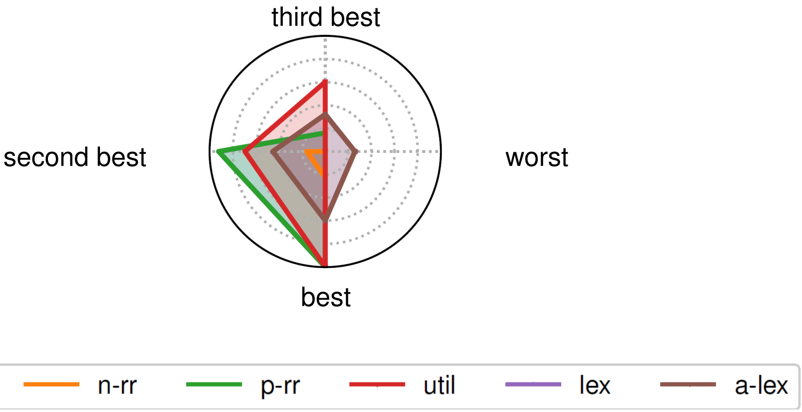


- a-lex performs at almost 85% of the optimal utility
- lex is equivalent to a-lex ($< 5\%$ on average)
- p-rr performs at almost 71% of the optimal utility
- n-rr results in very low utility allocations
- greedy is slightly worse than p-rr

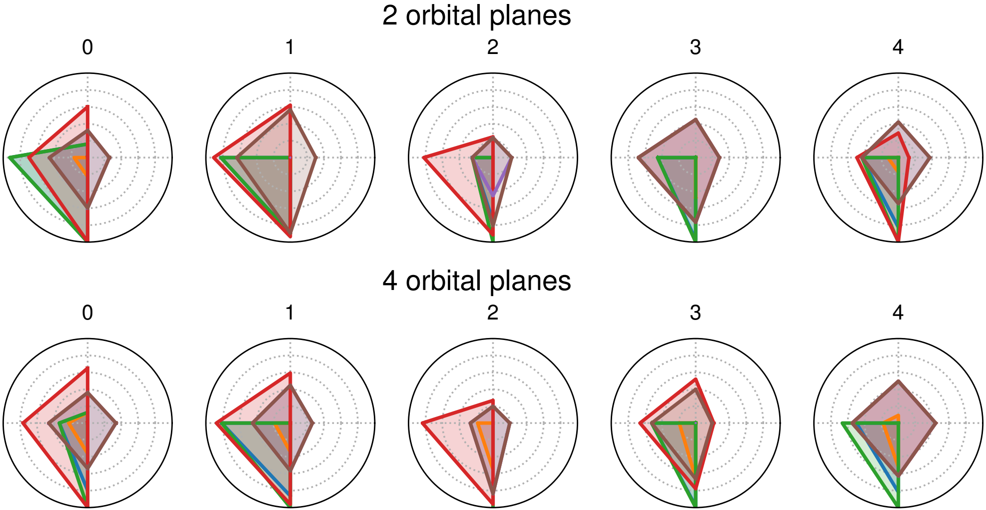
Results (cont.)
Utility and computation time

⇒ Larger constellation problems are easier to solve for the utilitarian perspective for non optimal algorithms, because there are more options to build conflict-free paths

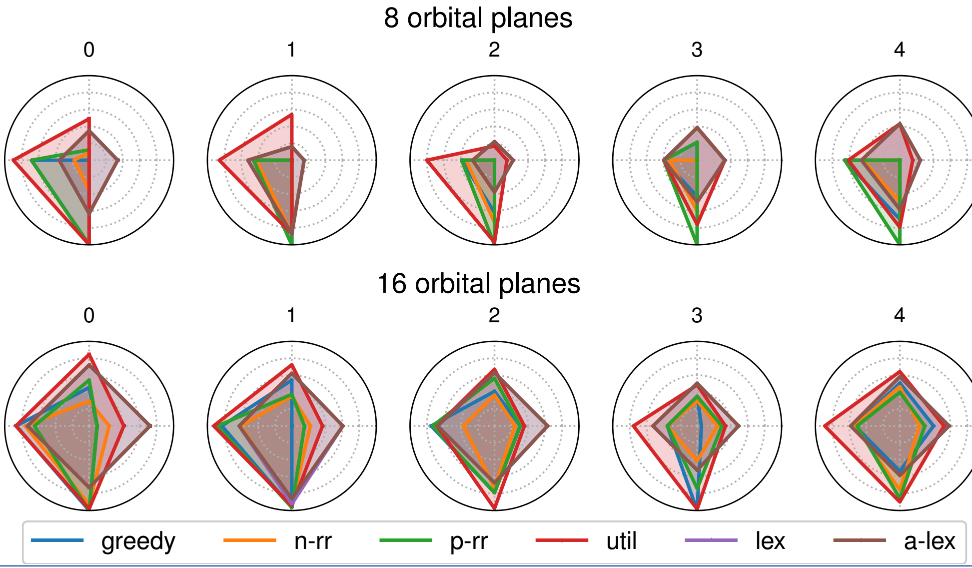
Results
Uitlity profiles (leximin ordering)



Results
Fairness (10 first instances over 30)



Results (cont.)
Fairness (10 first instances over 30)



Results (cont.)
Fairness (10 first instances over 30)

- greedy unfairly allocate to the first agents
- *round-robin* methods are more fair, but struggle on larger constellations
- util neglects the last user
- lex and a-lex are the most fair and are equivalent

Today's Menu

Conclusions

- First approach to allocate bundles with conflicting preferences represented as DAGs
- Several investigated methods
  - util, lex, a-lex, greedy, p-rr, n-rr
- Experiments on orbit slot allocation problems
- a-lex is the best compromise between utilitarianism, fairness and computation time

Perspectives

- Work in progress
  - Consider other types of requests (and graphs)
    - systematic requests
    - stereoscopic requests
  - Consider other methods for these graphs
    - Constraint programming (PAIS'22)
    - Local search and Min-conflict
- Future work
  - Consider areas (AOI instead of POI)
  - Consider other applications (e.g. NFV)

Acknowledgements

This work has been performed with the support of the French government in the context of the “Programme d’Invertissements d’Avenir”, namely by the BPI PSPC project “LiChIE”

Thank you for your attention!

www.onera.fr