Allocation de tâches et de ressources dans un système multi-agent

AI Mouaddib

Master 1 DECIM,

Université de Caen

Le problème

- Un ensemble d'agents **A**
- Un ensemble de tâches ou ressources **R**
- Allouer les tâches de R aux agents de A
- Chaque agent de A doit **activement décider des choix** dans **la procédure d'allocation**
- Les décisions des agents sont guidées par les **préférences** de ces derniers.

Quelques exemples

- Deux postiers doivent se répartir la distribution des couriers
- Deux satellites d'observations doivent se répartir les tâches d'observation
- Deux entreprises doivent se partager le marché d'un produit.

Les préférences sur les allocations

- Définition : Une allocation **A** est une répartition des tâches de **R** sur les agents de **A**
  - **Exemple :**
    - A(1) = {r1, r3}, A(2) = {r2, r4}
    - A'(1) = {r2, r3}, A'(2) = {r1, r4}
- **Préférences sur les allocations**
  - **Préférences individuelles**
    - **A'(1) > A(1) mais A(2) > A'(2)**
  - **Préférences collectives**
    - **A > A' ou A'> A par rapport à une fonction de préférences collective**
  - **Comment définir des préférences collectives à partir des préférences individuelles ?**

Fonctions d'utilité

- Chaque agent à une fonction d'utilité défnie sur les allocations ou sur les tâches individuelles
  - $\mathbf{U_i} : R \rightarrow \mathcal{R}$
    - $\mathbf{U_i}(r1) = 2$, $\mathbf{U_i}(r2) = 3$, $\mathbf{U_i}(r3) = 3$, $\mathbf{U_i}(r4) = 2$
    - $\mathbf{U_i}(A(i)) = \sum \mathbf{U_i}(r)$, $r \;\; A$
  - $\mathbf{U_i} : 2^R \rightarrow \mathcal{R}$
    - $\mathbf{U_i}(\{r1,r4\}) = 3$, $\mathbf{U_i}(\{r2,3\}) = 5$, $\mathbf{U_i}(\{r1, r2, r3)) = 6$
- On dit qu'un agent **i** préfère une Allocation A à une allocation A' quand $\mathbf{U_i}(A(i)) > \mathbf{U_i}(A'(i))$

De la préférence indvidiuelle à la préférence collective

- Difficulté : soient deux agents **i** et **j** et deux allocations A et A' telles que :
  - $\mathbf{U_i}(A(i)) > \mathbf{U_i}(A'(i))$
  - $\mathbf{U_j}(A'(j)) > \mathbf{U_j}(A(j))$
- Comment atteindre un accord mutuel quand les intérêts individuels divergent ?
- Exemple
  - $\mathbf{U_1}(A) = 2$ $\mathbf{U_1}(B) = 8$
  - $\mathbf{U_2}(A) = 4$ $\mathbf{U_2}(B) = 3$
- Nous avons besoin d'une approche systématique

La fonction d'utilité collective (CUF)

- Vecteur d'utilité :
  - Soit un enemble d'agent {1, 2, …, n}
  - Une allocation A sur les agents produit un vecteur d'utilité, noté, **u** = **< u_1(A), u_2(A), …, u_n(A) >** où **u_i(A)** est l'utilité de l'allocation **A** pour l'agent **i**.
- Pour comparer deux allocations A et A', nous devons comparer les vecteurs d'utilité associées **u** et **v**
- **CUV est une fonction qui a un vecteur d'utilités associe une valeur réelle.**

CUV : $R^n \rightarrow R$

On dit que u > v si CUV(u) > CUV(v)

Social Welfare

- Un ordre social (SWO) ≥ est relation binaire entre deux vecteurs d'utilité.
- SWO peut être définie par la fonction CUF telle que :
  - U ≥ V ssi CUV(U) ≥ CUV(V)
- CUV peut être définie selon différent critère du système
  - **Utilitariste**
  - **Egalitarise**
  - **Elitiste**

Utilitarian Social Welfare

- Un système est dit **utilitariste** est un système qui cherche à **maximiser le profit total**
  - Jeremy Bentham, philosophe anglais (1748-1832)
- **Utilitarian CUF :**

$$\mathbf{SW_u(V) = \sum v_i} \qquad \text{avec i un agent.}$$

- Exemple
  - U = < 19, 3, 5, 2>; SW(U) = 29
  - V = < 9, 9, 9, 9>; SW (V) = 27
  - **U = ARGMAX (SW(U), SW(V)) (SW(U) > SW(V))**
  - Remarque : la majorité des agents sont plus satisfaits avec V qu'avec U mais c'est U qui est préfér d'après l'approche Utilitariste.

Egalitarian Social Welfare

- Un système est dit **égalitariste** est un système qui cherche à **maximiser le profit du membre le plus faible.**
  - John Rwals, philosophe américain (1921-2002)
- **Egalitarian CUF :**

$$\mathbf{SW_u(V) = min_i\ v_i} \qquad \text{avec i un agent.}$$

- Exemple
  - U = < 19, 3, 5, 2>; SW(U) = 2
  - V = < 9, 9, 9, 9>; SW (V) = 9
  - V **= ARGMAX (SW(U), SW(V))= ARGMAX (2, 9)** .
  - Remarque, l'agent 1 n'est pas satisfait.

Approche élitiste

- Un système est dit **élitiste** est un système qui cherche à **maximiser le profit du membre le plus fort.**
- **Elitist CUF :**

  $SW_u(V) = max_i\ v_i$ avec i un agent.

- Exemple
  - U = < 19, 3, 5, 2>; SW(U) = 19
  - V = < 9, 9, 9, 9>; SW (V) = 9
  - **U = ARGMAX (SW(U), SW(V))= ARGMAX (19, 9)** .
  - Remarque, tous les agents sauf le 1 ne sont pas satisfaits

Ordered Utility vector

- Le vecteur d'utilité U est ordonné dans un ordre croissant/décroissant Ū
  - Exemple :
    - Soit U = $<5, 20, 0>$ un vecteur d'utilité, avec le premier a une utilité individuelle 5, le deuxième 20 et le troisième 0.
    - Ū = $<0, 5, 20>$ un vecteur d'utilité où **le plus faible a 0**, **le plus fort a 20**.
- L'intérêt des vecteurs ordonnés est de permettre une analyse sur les éléments les plus forts/faibles qu'une simple somme ou moyenne qui peut cacher une disparité.

Lexmin Ordering

- $\leq_l$ peut être vu comme une amélioration de l'approche égalitariste
- L'ordre ***Lexmin*** $\leq_l$ est défini comme suit :
  - U $\leq_l$ Z si et seulement si **Ŭ** lexicallement précède **Ž Signifie que :**
  - **$\breve{U} = \check{Z}$ ou**
  - **Il existe un $k \leq n$ tel que**
    - **$\breve{U}_i = \check{Z}_i$ pour tout $l < k$ et**
    - **$\breve{U}_k < \check{Z}_k$**
- **Exemple :** $<0,6,20> \ \leq_l \ <0, 6, 24>$

Produit Nash

- **Nash CUF**

  $\mathbf{SW(V) = \prod_i v_i}$ avec i un agent.

- Exemple
  - U = < 10, 3, 5, 2>; SW(U) = 300
  - V = < 9, 9, 9, 9>; SW (V) = 6561
  - **V = ARGMAX (SW(U), SW(V))= ARGMAX (300, 6561)** .
  - Remarque, l'agent 1 n'est pas satisfait.

- Product Nash est comme l'approche utilitariste mais réduit aussi l'inégalité de la distribution.

Rang dictateur

- K-rank dictator CUF, pour un K associe à un vecteur d'utilités la valeur du Kième élément du vecteur :
- K-rank dictator CUF :

$$SW_k(U) = \breve{U}_k$$

- Exemple
  - U = < 10, 3, 5, 2>; $SW_2(\breve{U}) = 3$
  - Z = < 9, 9, 9, 9>; $SW_2\ (\check{Z}) = 9$
  - **Z = ARGMAX (SW(U), SW(V))= ARGMAX (9, 3)** .
- Note : k = 1, approach égalitariste, k = n approach élitiste

Autres fonctions

- Fonction d’utilité à base de regret
- Fonctions d’utilités normalisées
- Fonctions d’utilités généralisées
- Fonction d’utilités multi-dimensionnelles

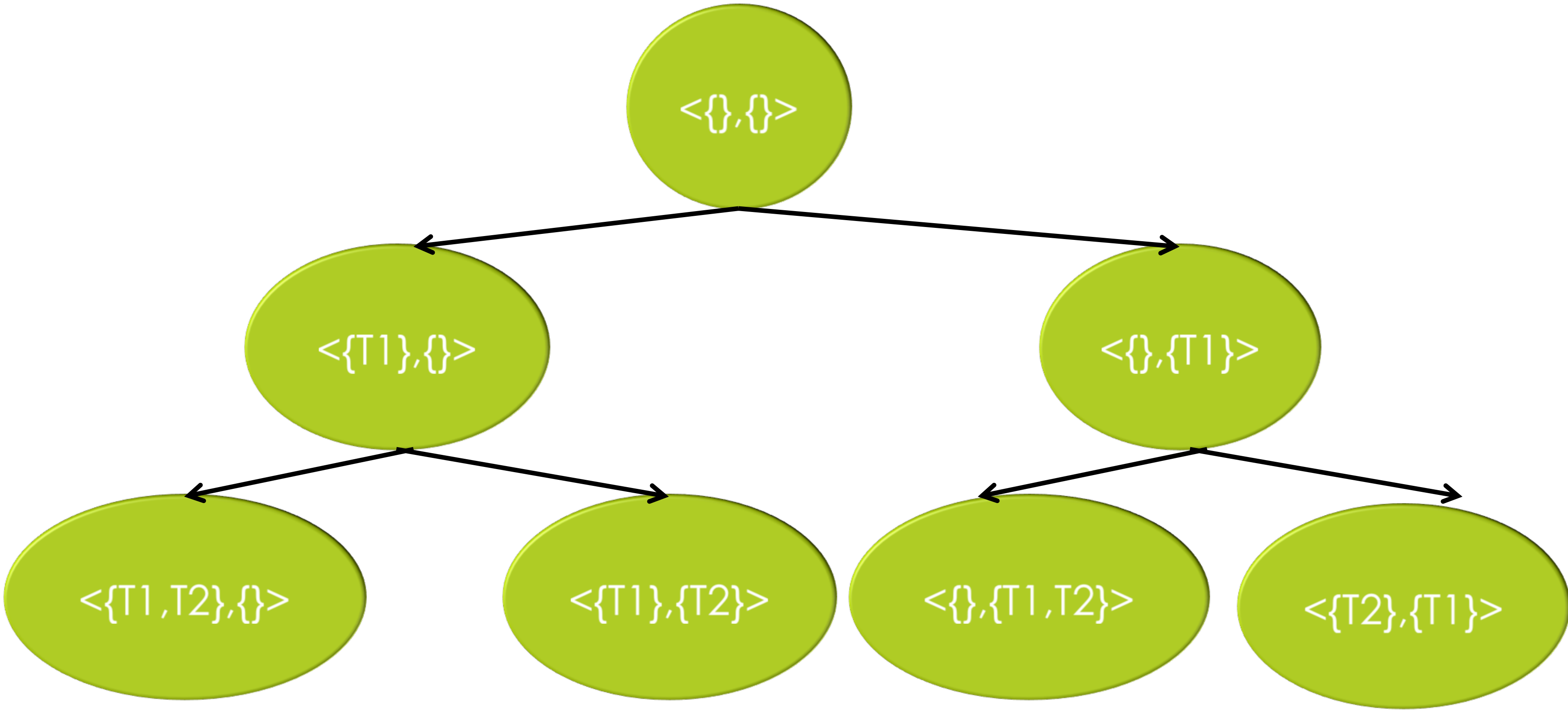
Procédures d'allocations

- Les procédures d'applications permettent de définir des fonctions de décision des agents :
  - Protocol : Les objets de négotiation entre les agents ? Les messages échangés entre agents ?
  - Stratégie : Quelle stratégie un agent doit suivre pour choisir ses teaches/ressources ? Égoisme, altruisme, optimisme, pessimisme, …
  - Algorithmes : traiter la combinatoire des choix

Approche centralisée

- Approche naïve **: arbre de décision**
  - Allocation initiale : <{}, {}, …, {}>
  - A chaque étape, on alloue une tâche à un agent.
  - Les feuilles sont toutes les allocations possibles
  - Choisir la meilleure en utilisant CUF

Illustration

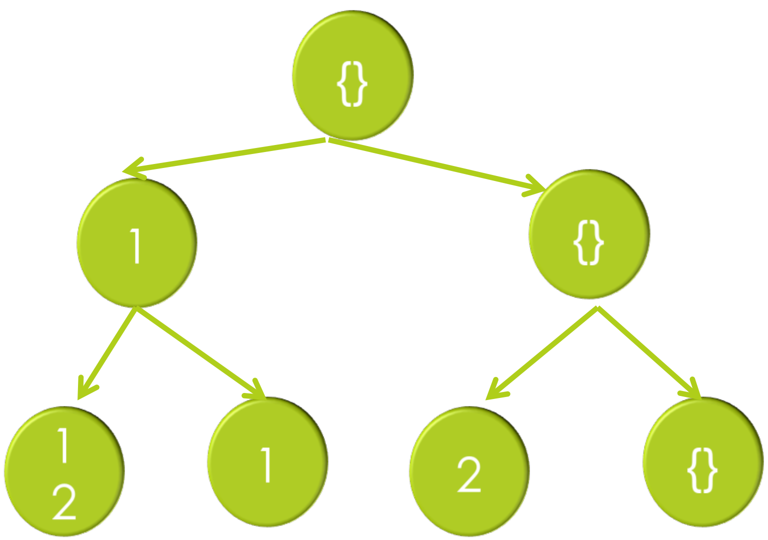


Approche distribuée

- Chaque agent choisit une allocation individuelle
- Il envoie son allocation aux autres agents
- Si refus par un agent,
  - Négotiaition :
    - choisir une autre allocation
    - Demander aux autres de modifier les leurs

Calcul d’une allocation locale

Arbre de décision local



Choix du meilleur noeud feuille.

Préférences sur les allocations

- Soit T un ensemble de tâche (par exempel {1,2})
- Soit U une fonction d’utilité, qui pour tout T’ de T, on a U(T’).
- Choix de A telle que

A = argmax U(T’)

Protocol Contract-Net

- Phases :
  - Annonce
  - Proposition d’une offre
  - Choix de la meilleure offre
  - Allocation
  - Confirmation
- Suppose un manager.
- **Extension à plusieurs manager**
  - **Chaque agent manage son allocation**.

Stratégie égoiste

- Startégie qui consiste à choisir l'allocation qui maximise l'utilité individuelle.
- Risque de plusieurs conflits à résoudre.
- Résolution de conflits
  - Système coopératif : changer l'allocation individuelle qui améliore la CUF.
  - Système compétitif : **théorie des jeux, théorie du choix social**

Stratégie Altruiste

- Startégie qui consiste à choisir l'allocation qui provoquera le moins de conflit.
  - S'adapter aux choix des autres
- Risque d'être moins favorisé dans un environnement compétitif
- Définir une nouvelle fonction d'utilité qui tient compte des autres par la fonction C.

$$U(A) = x.I(A) + (1-x).C(A)$$

  - X = 1 : égoiste
  - X= 0 = altruiste
- Décision distribuée

Agent égoiste dans un système coopératif

- Chaque agent trie dans l'ordre croissant ses allocations
- Il propose sa meilleure allocation
- Celui qui <u>**a le moins d'impact (moins de regret)**</u> change son allocation.

  Regret (A*, A') = U(A') – U(A*)

- Répéter le processus jusqu'à convergence

Agent Altruiste avec une fonction d'utilité distribué

- **U(A) = x.I(A) + (1-x).C(A)**
- Que vaut C(A) :
  - A contient les tâches que les autres agents ne vont pas obtenir.
  - Calculer un coût pour chacune de ces tâches par agent.

Cout(t, A) = U(A-{t}) – U(A)

- **Plus le coût est élevé pour l'impact collectif moins est bonne l'allocation, même d'un point de vue individuel.**

Conclusion

- Fonctions CUF pour évaluer les allocations
- Procédures d'allocation
  - Protocol
  - Stratégies : égoisite, altruiste, coopératif, compétitif
  - Algorithmes : centralisés, distributés
- Prochaine fois : systèmes compétitifs
  - **Théorie des jeux**
  - **Théorie du choix social**