Planification en environnement situé et stratégies d'équipes

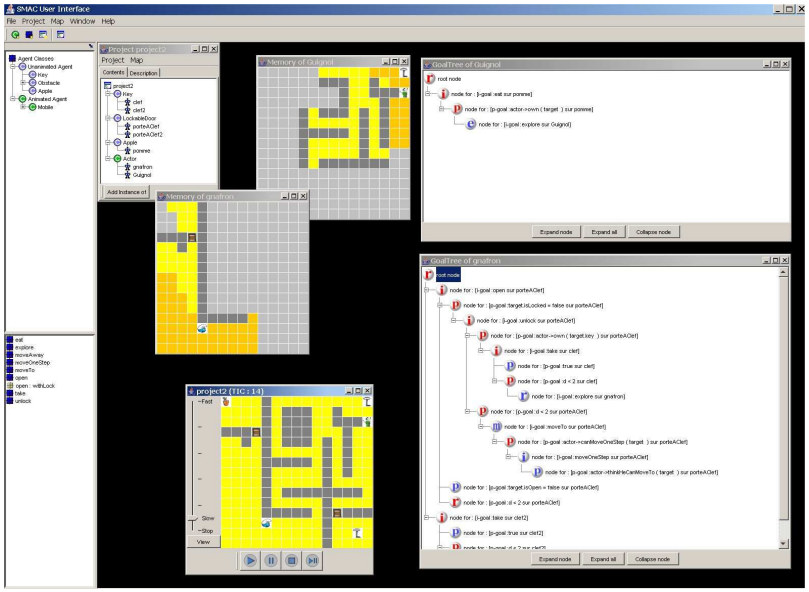
Damien Devigne
Philippe Mathieu
Jean-Christophe Routier
Equipe SMAC, LIFL, LILLE 1
{devigne, mathieu, routier}@lifl.fr

Plan

- CoCoA
- Notre modèle d'agents centré interactions
- Présentation du moteur de planification
- Intégration des équipes dans le modèle
- Conclusion

CoCoA (Collaborative Cognitive Agents)

- Logiciel de simulation de comportements d'équipes s'appuyant sur des agents cognitifs autonomes situés géographiquement dans un environnement
- Séparation
  - Agents
  - Interactions
  - Environnement



Exemples d'application

- Simulations sociales in situ
- Robocup Rescue
- Jeux vidéo

Différents points de vue

- Autres approches
  - Emergence
  - Plan-Merging
  - Plans statiques et réseaux de Pétri
- Notre approche
  - Planification pour agents cognitifs situés avec chef

Notre modèle d'agents centré interactions

- Interactions
  - Connaissance abstraite
  - Acteur, cible
- Exemple

  Ouvrir

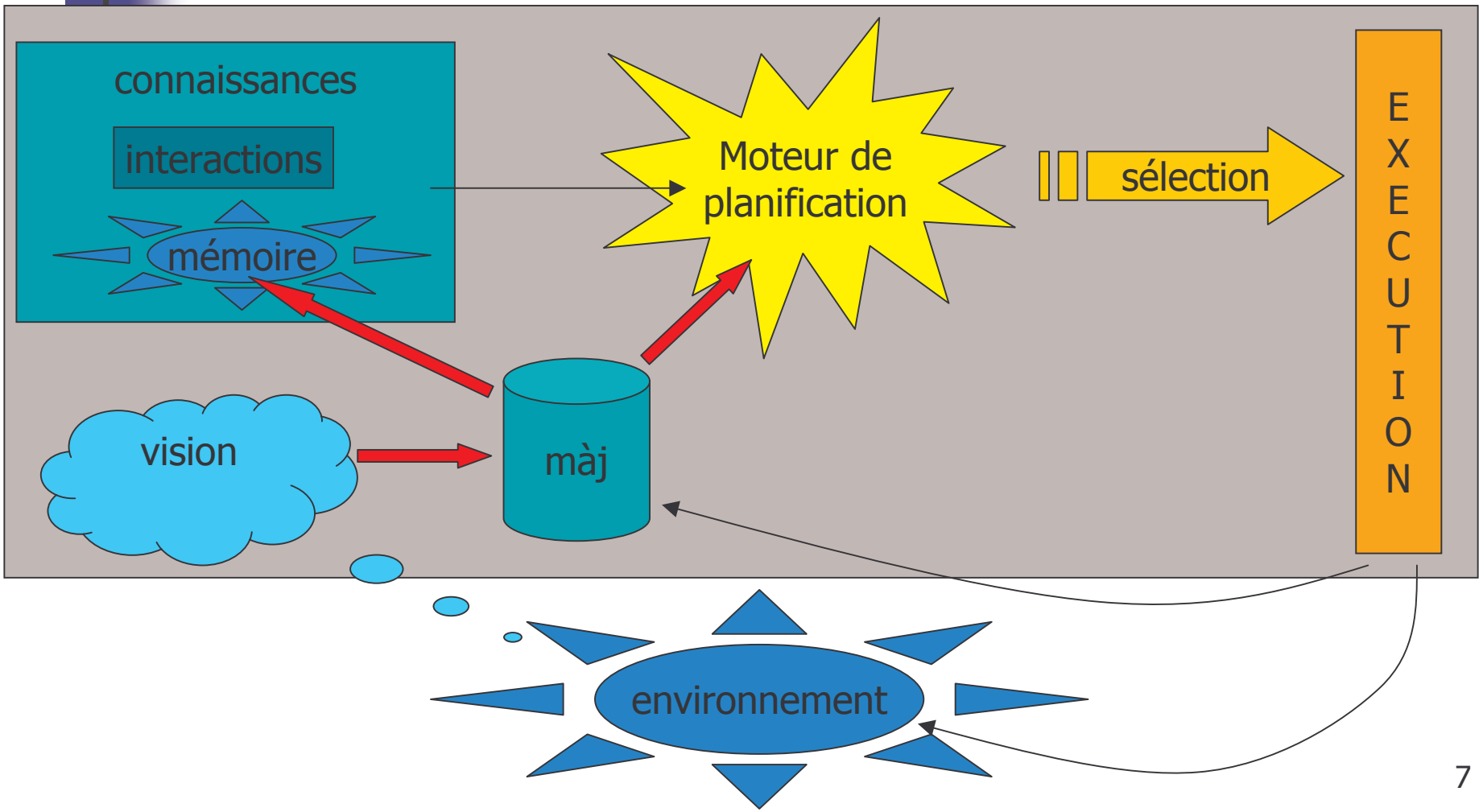
  conditions: acteur.possede(cible.clef)

  garde: distance(acteur, cible) < 1

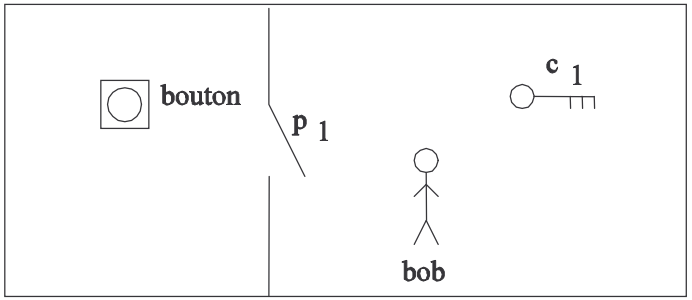
  actions: cible.verrouillee = faux

- Agents
  - propriétés
    - Peut-subir *(cible)*
    - Peut-effectuer *(acteurs)*
  - non omniscients
  - situés géographiquement
  - cognitifs

Le moteur de comportement



Les gardes de distance




```
appuyer bouton
  distance(bob,bouton)<1
    aller bouton
      p1.verrouillee = faux
        ouvrir p1
        +-posseder c1
        |   prendre c1
        |     distance(bob,c1)<1
        |       aller c1
        +-distance(bob,p1)<1
            aller p1
```

Appuyer
c :
g : distance(acteur, cible) < 1
a : bouton.appuyé = vrai

Aller
c : conditions de l'environnement
g :
a : distance(acteur, cible) < 1

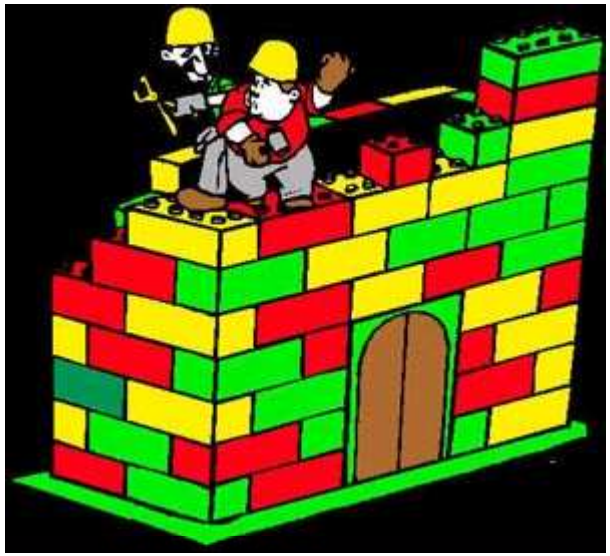
Ouvrir
c : acteur.possede(cible.clef) = vrai
g : distance(acteur, cible) < 1
a : cible.verrouillée = faux

Prendre
c :
g : distance(acteur, cible) < 1
a : acteur.possede(cible) = vrai

Quelques problèmes d'équipes

- Problèmes :
  - Equipes sans chef (auto-organisation)
  - Equipes avec chef (gestion centralisée)
  - Gestion de la synchronisation
  - Equipes antagonistes
  - Dynamicité de l'équipe
- Notre approche :
  - Equipes d'agents autonomes, collaboratifs, avec chef, sans simultanéité

Exemple de problème d'équipe



- maçon
- menuisier
- charpentier
- couvreur
- électricien
- plombier
- peintre
- ...

Collaboration des agents : Nécessité des équipes

Agents animés :
A1
peut-effectuer : {I1}
peut-subir : Ø

A2
peut-effectuer : {I2}
peut-subir : Ø

Agents inanimés :
A3
peut-subir : {I1}
A4
peut-subir : {I2}

Environnement :
¬P0
¬P1
P2

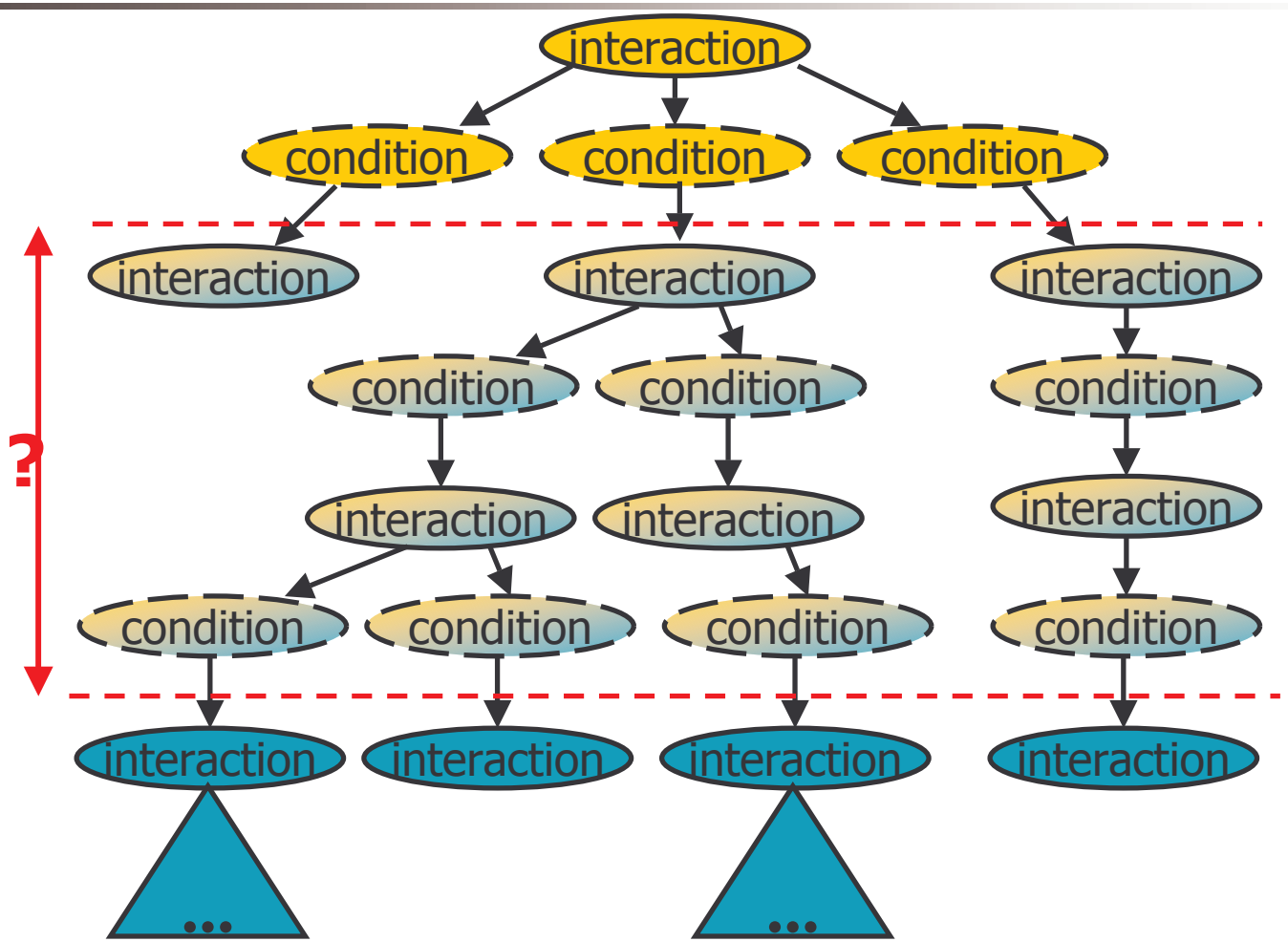
But des agents :
P0

Interaction I1
conditions P1
garde G1
actions P0

Interaction I2
conditions P2
garde G2
actions P1

Problèmes :
A1 ne peut pas déclencher I1 car ¬P1
A2 ne déclenche pas P2 car il ne connait pas I1

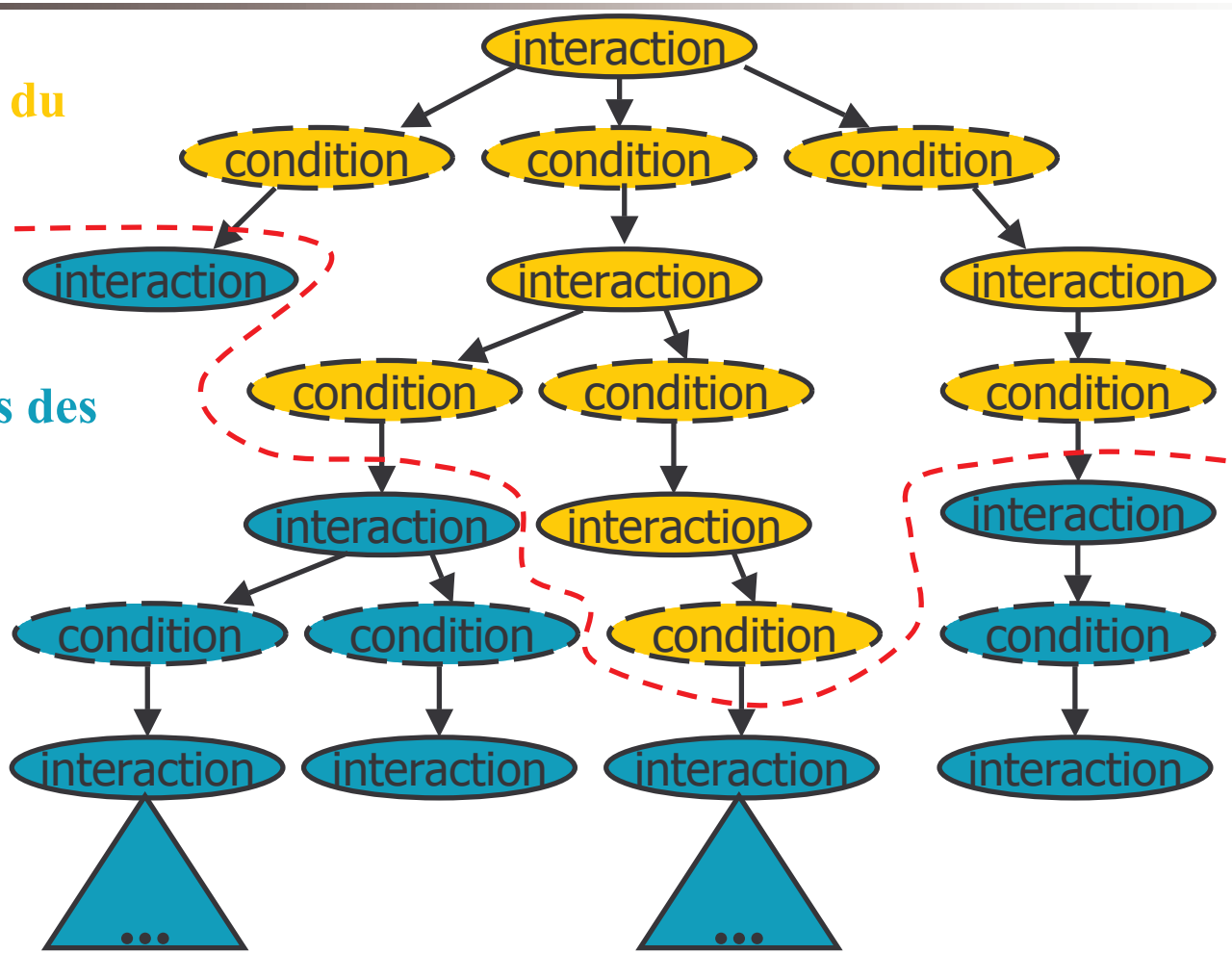
Problème de l'autonomie des agents



Problème de l'autonomie des agents

Plan abstrait du chef

Plans concrets des agents

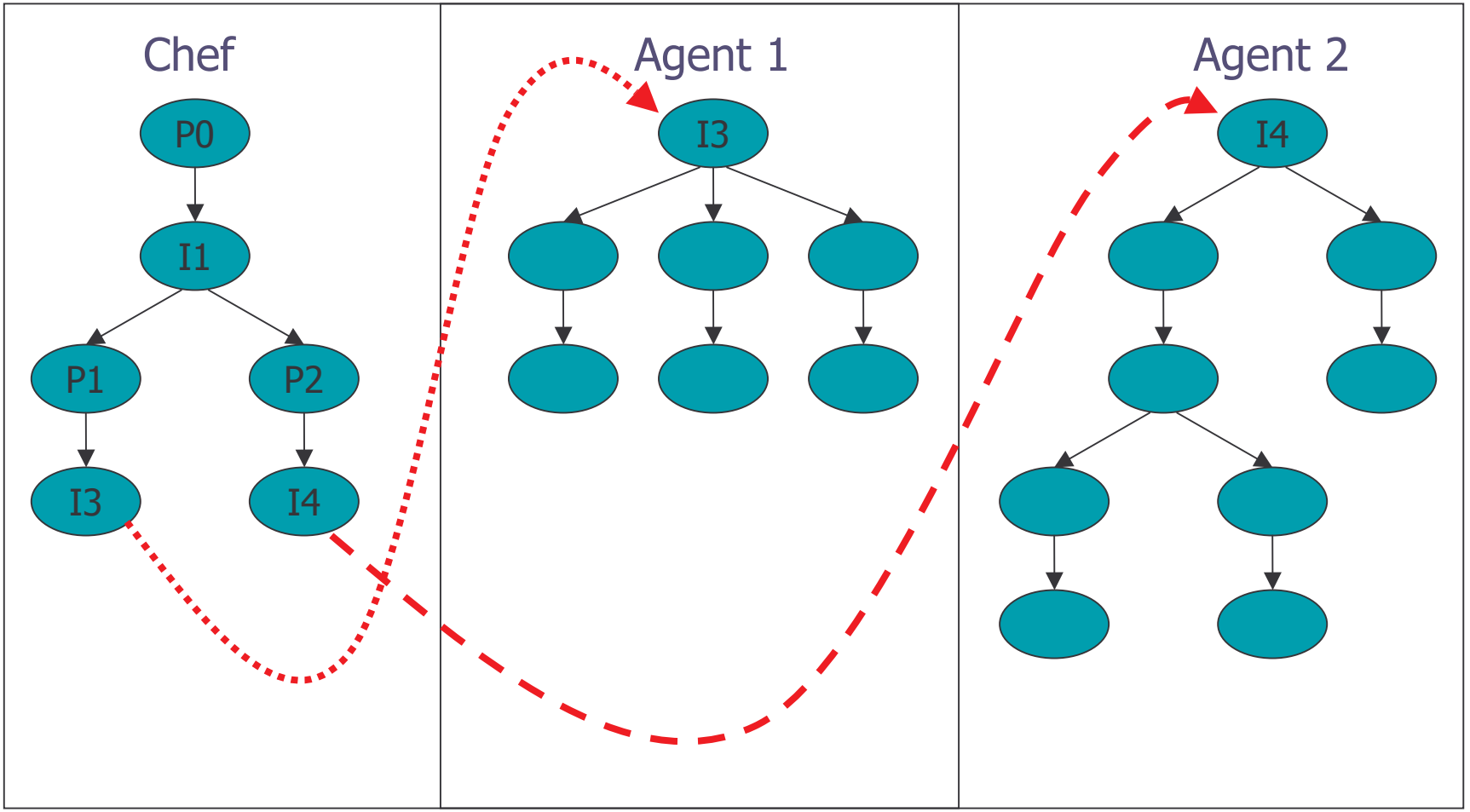


Notre méthode : Interprétation abstraite du plan

- Ajout des interactions des agents aux connaissances du chef SANS les conditions et les gardes

	chef.peut-effectuer	équipe.peut-effectuer		
nom	I1	I2	I3	I4
conditions	P1, P2	-	-	-
garde	G1	-	-	-
actions	P0	P1	P2	P3

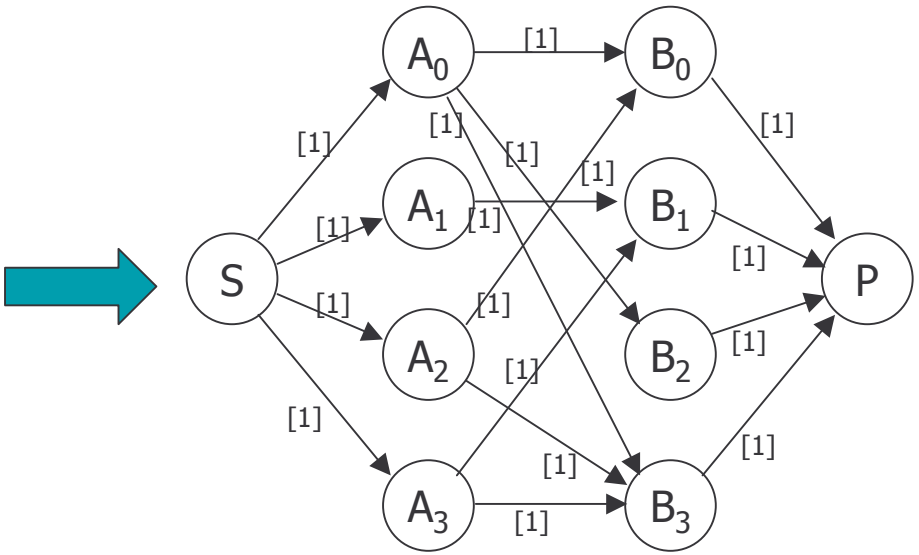
Plan d'équipe, plan des agents



Allocation des tâches

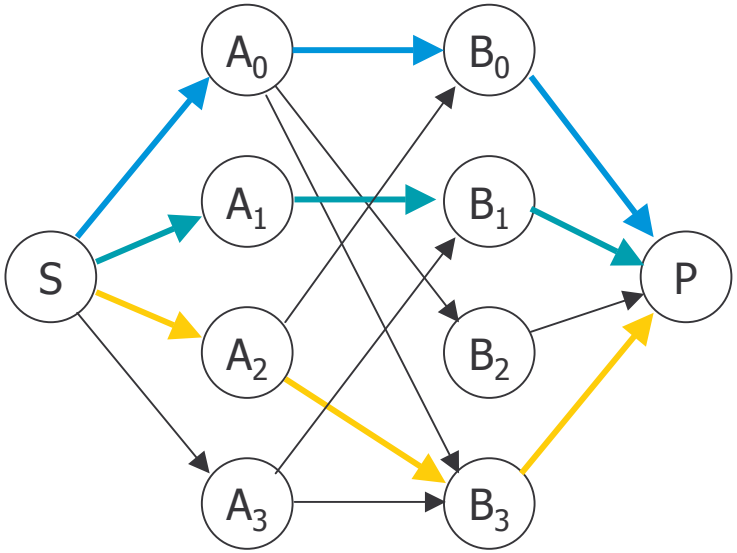
- Transformation en graphe de flot

	B_0	B_1	B_2	B_3
A_0	X		X	X
A_1		X		
A_2	X			X
A_3		X		X



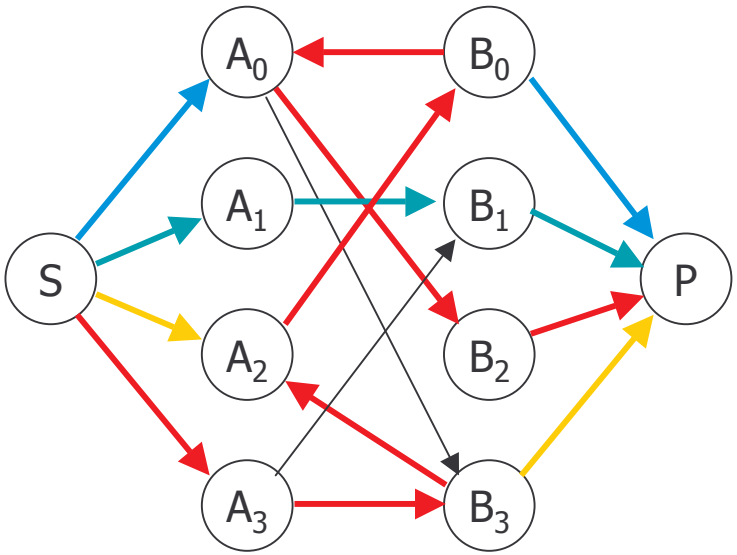
Allocation des tâches

- Algorithme de Ford-Fulkerson



Allocation des tâches

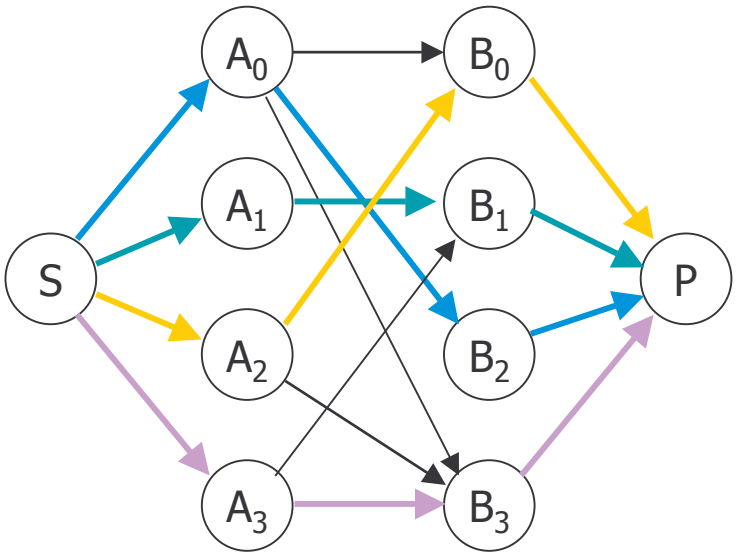
- Algorithme de Ford-Fulkerson



- On avance sur les arcs non saturés et on les sature
- On recule sur les arcs saturés et on les désature

Allocation des tâches

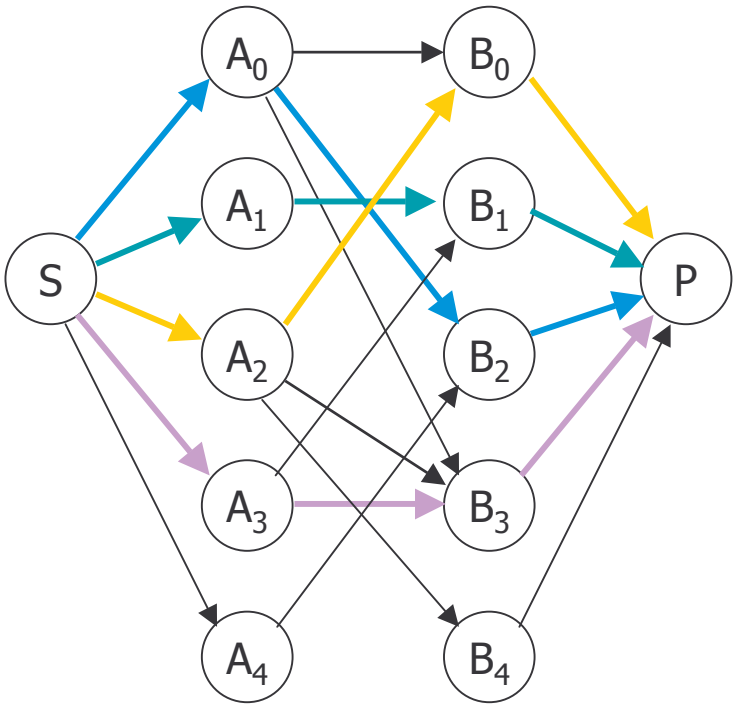
- Fin de l'algo



- Si plus de chaînes augmentantes alors flot maximal
- L'affectation maximale est donnée par les arcs (A_i, B_j)

Réallocation des tâches

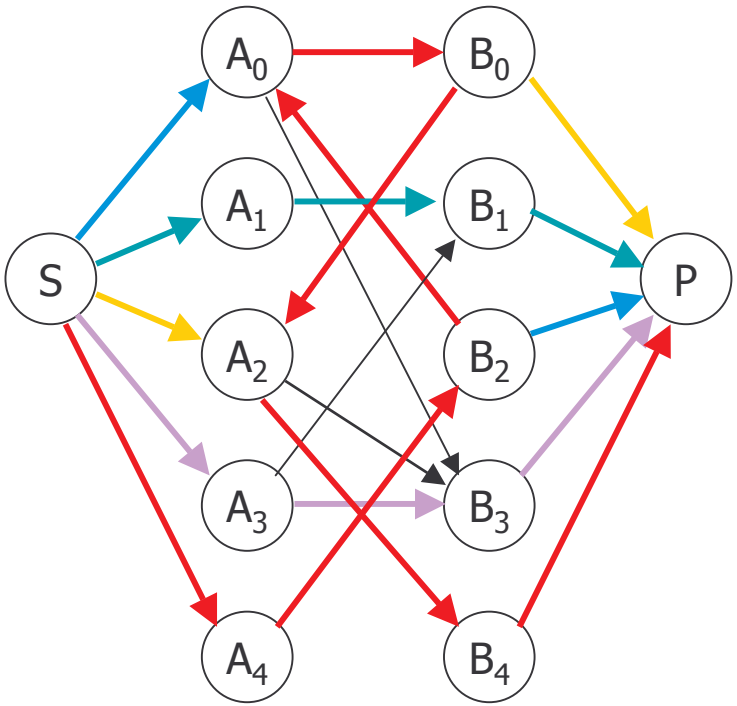
- Ajout de nœuds dans le graphe



- Agent libre :
  - Nouvel agent
  - Agent libéré
- Tâche à affecter :
  - Nouvelle tâche
  - Agent supprimé

Réallocation des tâches

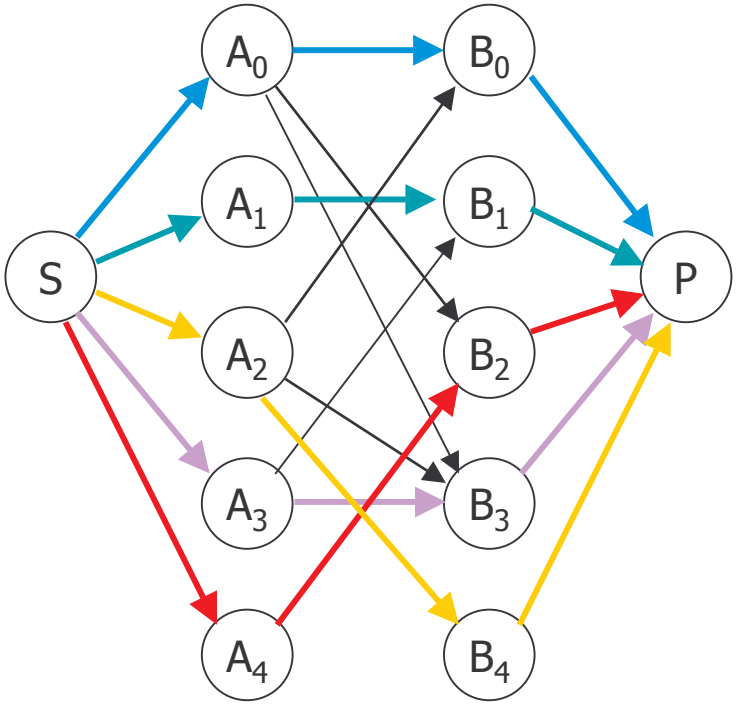
- Algorithme de Edmonds-Karp



- Recherche de chaînes augmentantes de longueur minimale
- Minimisation du nombre de changements

Réallocation des tâches

- Algorithme de Edmonds-Karp

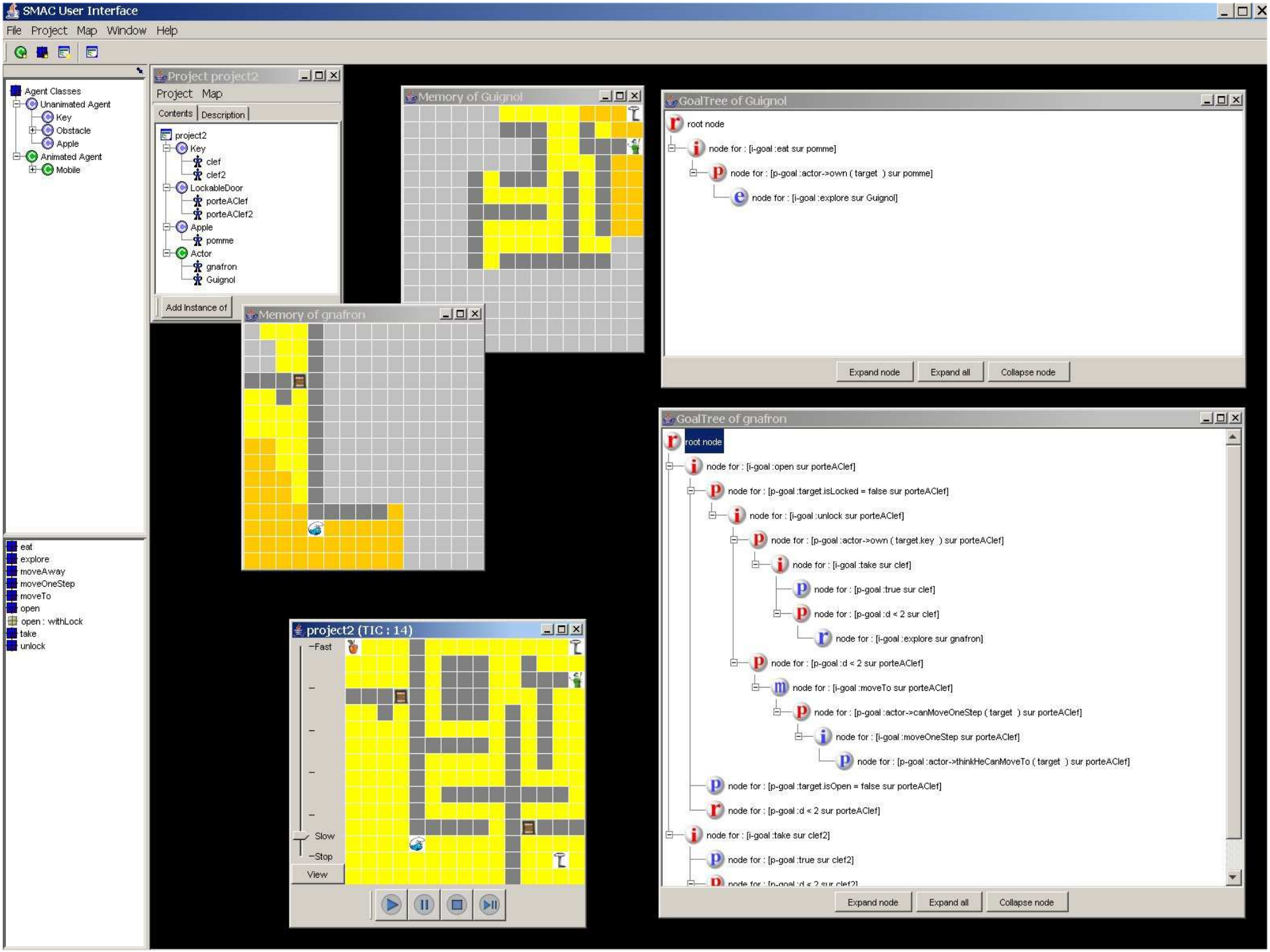


Réallocation des tâches

- Améliorations
  - Prise en compte du « coût » des changements (distance à parcourir pour atteindre la nouvelle cible)
  - Algorithme hongrois

Conclusion

- Notre approche
  - modèle original centré interactions, interprétation abstraite du plan
- Travaux futurs
  - Gestion des échecs en équipe, remontée d'information
  - Gestion dynamique des capacités de l'équipe (recrutement)
  - Minimisation du temps total d'inactivité des agents
  - Synchronisation


SMAC User Interface
File Project Map Window Help
Agent Classes
Unanimated Agent
Key
Obstacle
Apple
Animated Agent
Mobile
eat
explore
moveAway
moveOneStep
moveTo
open
open : withLock
take
unlock
Project project2
Project Map
Contents Description
project2
Key
clef
clef2
LockableDoor
porteAClef
porteAClef2
Apple
pomme
Actor
gnafron
Guignol
Add Instance of
Memory of Guignol
Memory of gnafron
GoalTree of Guignol
root node
node for : [i-goal :eat sur pomme]
node for : [p-goal :actor->own (target) sur pomme]
node for : [i-goal :explore sur Guignol]
Expand node
Expand all
Collapse node
GoalTree of gnafron
root node
node for : [i-goal :open sur porteAClef]
node for : [p-goal :target.isLocked = false sur porteAClef]
node for : [i-goal :unlock sur porteAClef]
node for : [p-goal :actor->own (target.key) sur porteAClef]
node for : [i-goal :take sur clef]
node for : [p-goal :true sur clef]
node for : [p-goal :d < 2 sur clef]
node for : [i-goal :explore sur gnafron]
node for : [p-goal :d < 2 sur porteAClef]
node for : [i-goal :moveTo sur porteAClef]
node for : [p-goal :actor->canMoveOneStep (target) sur porteAClef]
node for : [i-goal :moveOneStep sur porteAClef]
node for : [p-goal :actor->thinkHeCanMoveTo (target) sur porteAClef]
node for : [p-goal :target.isOpen = false sur porteAClef]
node for : [p-goal :d < 2 sur porteAClef]
node for : [i-goal :take sur clef2]
node for : [p-goal :true sur clef2]
Expand node
Expand all
Collapse node
project2 (TIC : 14)
Fast
Slow
Stop
View