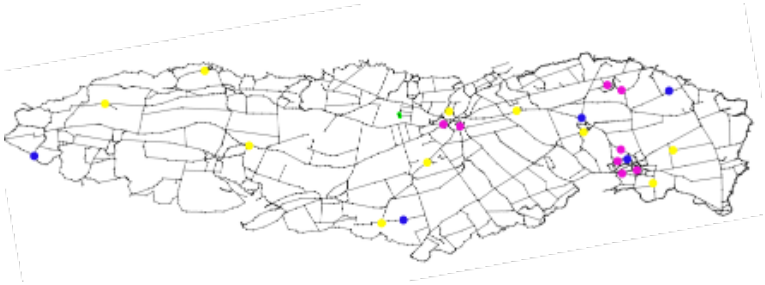
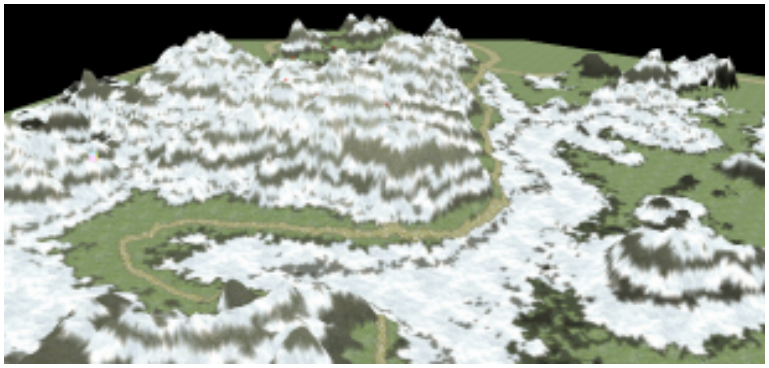
- DEDALE -

Projet FoSyMa

Master 1 ANDROIDE

Environnements discrets ou continus





https://dedale.gitlab.io

- Environnement ouvert, dynamique et partiellement observable
- Asynchronisme des traitements et des communications
- Conçu pour les problèmes de coordination

Dedale

Contexte

- Env. inconnu, ouvert, dynamique et partiellement observable
- Agents et communications asynchrones
- Rayons de communication limités

Objectif 2025

Récolter et déverser un maximum de ressources dans l'agent "Silo"

Dedale

Contexte

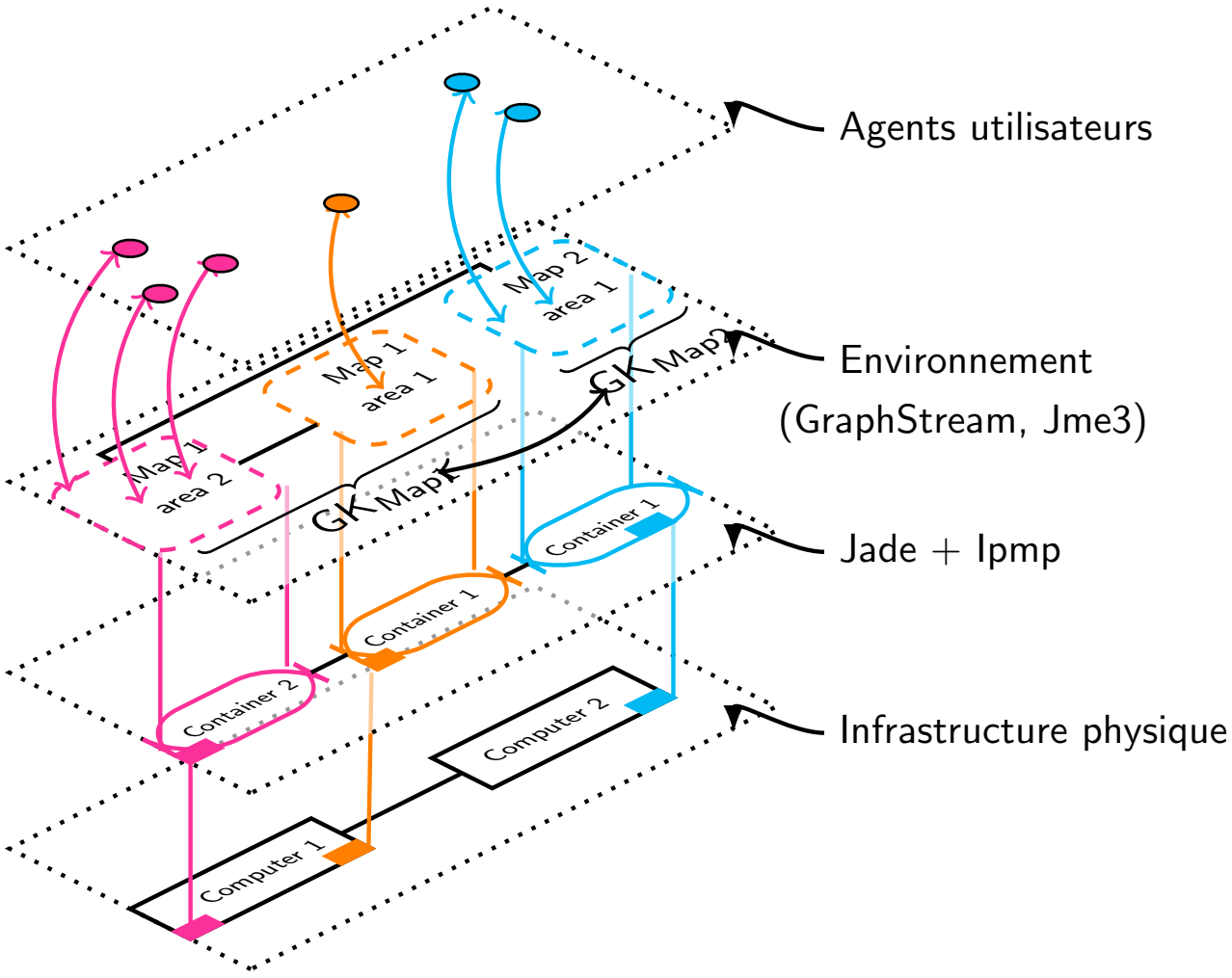
- Env. inconnu, ouvert, dynamique et partiellement observable
- Agents et communications asynchrones
- Rayons de communication limités

Objectif 2025

Récolter et déverser un maximum de ressources dans l'agent "Silo"

- Exploration collaborative efficiente de l'environnement
- Gestion des interblocages
- Stratégie coopérative de récolte

Architecture générale de la plateforme

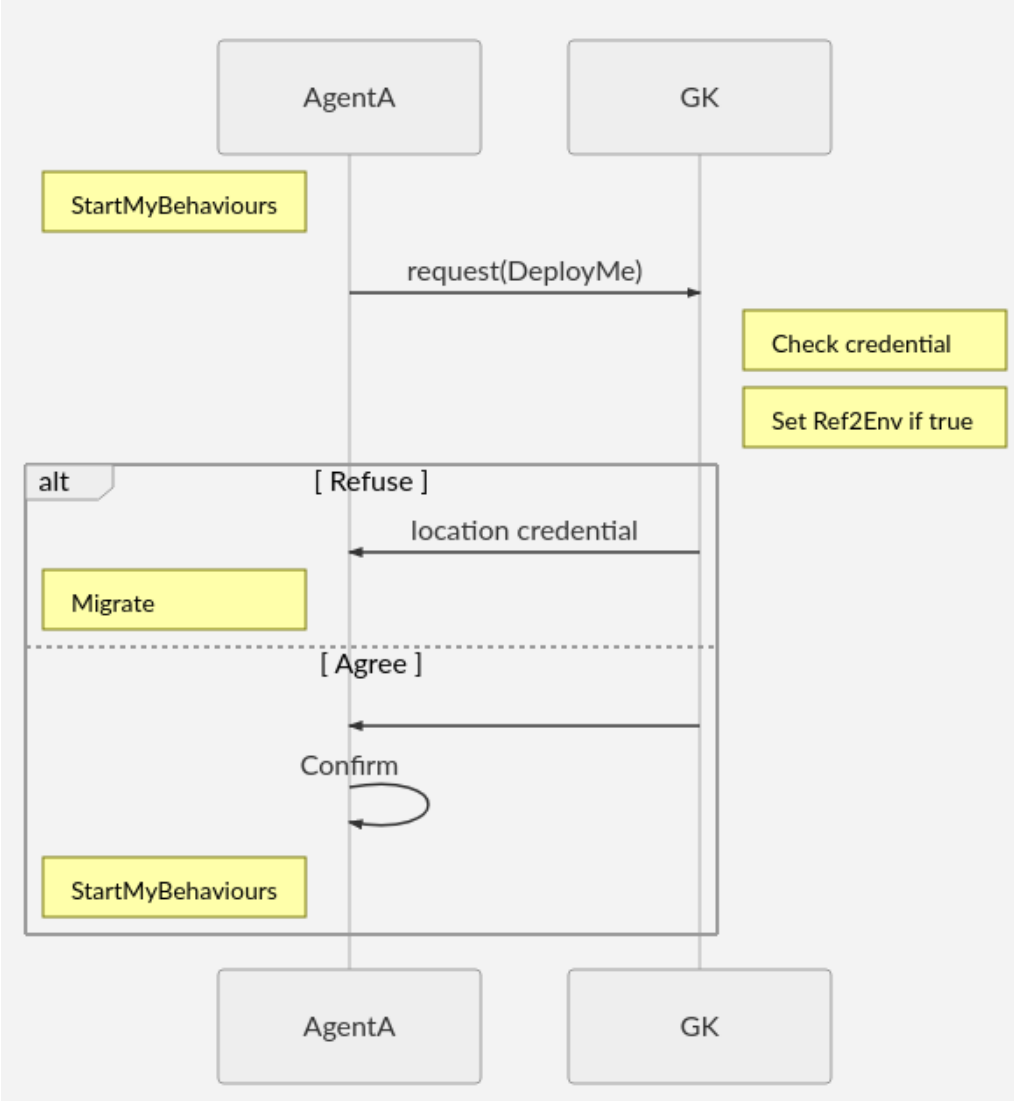


Création et déploiement d'un agent

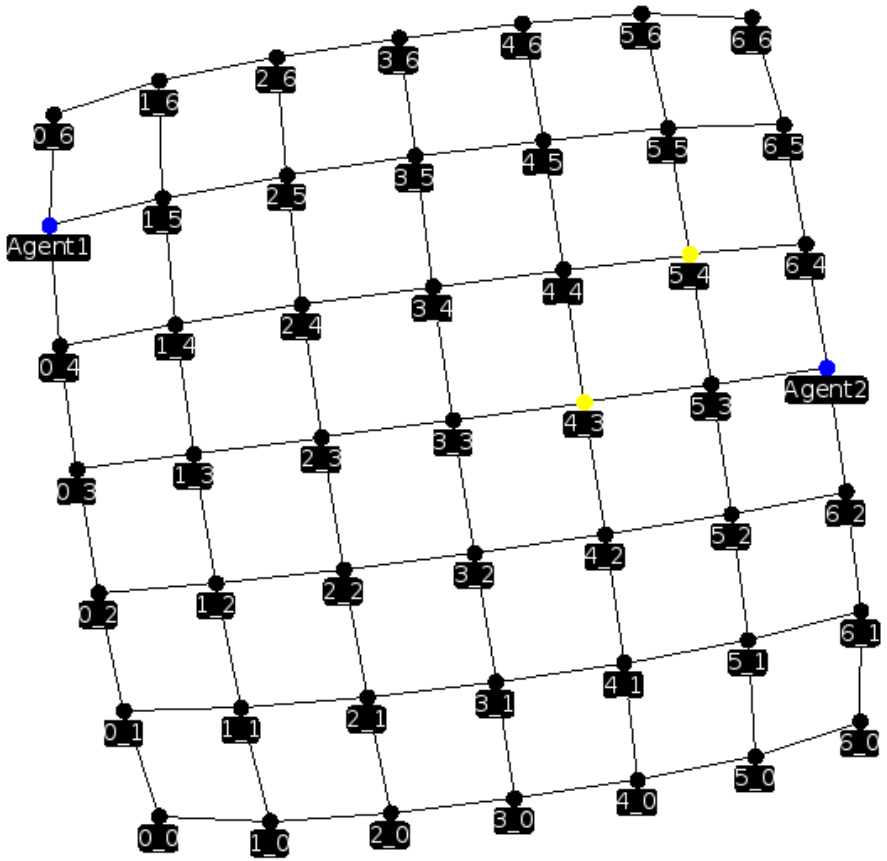
```
public class DummyMovingAgent extends AbstractDedaleAgent{
  protected void setup(){
    super.setup();

    //get the parameters given into the object[]
    //use them as parameters for your behaviours is you want
    final Object[] args = getArguments();

    List<Behaviour> lb=new ArrayList<Behaviour>();

    /************************************************
    * ADD the behaviours of the Dummy Moving Agent
    ************************************************/
    lb.add(new RandomWalkBehaviour(this));
    lb.add(new SayHello(this));
    /***************************************************
    * MANDATORY TO ALLOW YOUR AGENT TO BE DEPLOYED CORRECTLY
    ***************************************************/
    addBehaviour(new startMyBehaviours(this,lb));
}}
```

Création et déploiement d'un agent

Sous le capot, le travail du Gate-Keeper



Environnement : type 1 (Grille)



- Agents
- Coffres

- Agents déployés (aléatoirement ou non).
- Vision de la position courante et des identifiants des nœuds voisins.

Principales méthodes de l'API (1/2)

Déplacement et communication

- **String getCurrentPosition()** : Retourne la position courante de l'agent
- **List<Couple<String,List<Couple<Observation,Integer>>>> observe()** : Retourne l'ensemble des observables depuis la position courante de l'agent sous la forme d'une liste de couple (position, liste attribut/valeur)
- **boolean moveTo(String myDestination)** : Se déplacer jusqu'à la position fournie en paramètre (si atteignable). Cette fonction, lorsque elle est appelée, **doit** être la dernière méthode de votre behaviour.
- **void sendMessage(ACLMessage msg)** : Envoi de message qui gère le rayon de communication des agents. Utiliser **exclusivement** celle-ci.
  - **setContent(String s)** pour envoyer une chaîne de caractère
  - **setContentObject(Serializable o)** pour envoyer un objet **serializable** dans le message. Attention, c'est l'un ou l'autre, pas les deux.

Exemple d'utilisation de l'API par le DummyMovingAgent

Cf le code fourni en Tp

```
//1) Observation
getCurrentPosition() : 6
lobs = observe(): [<6,[(Treasure,43),( Stench,)]>, <7,[(Stench,)]>, <15,[(Stench,)]>, <5,[]>]

//2) Navigate within observations and take one as illustration
Couple onePosit= lobs.get(0);
onePosit.getLeft() : 6
List<Couple<Observation,Integer>> obsOnePosit = onePosit.getRight() : [(Treasure,43),(Stench,)]

obsOnePosit.get(0).getLeft() : Treasure
obsOnePosit.get(0).getRight() : 43
obsOnePosit.get(1).getLeft() : Stench

//3) Move
moveTo(5): true
```

Principales méthodes de l'API (2/2)

Coffres et trésors

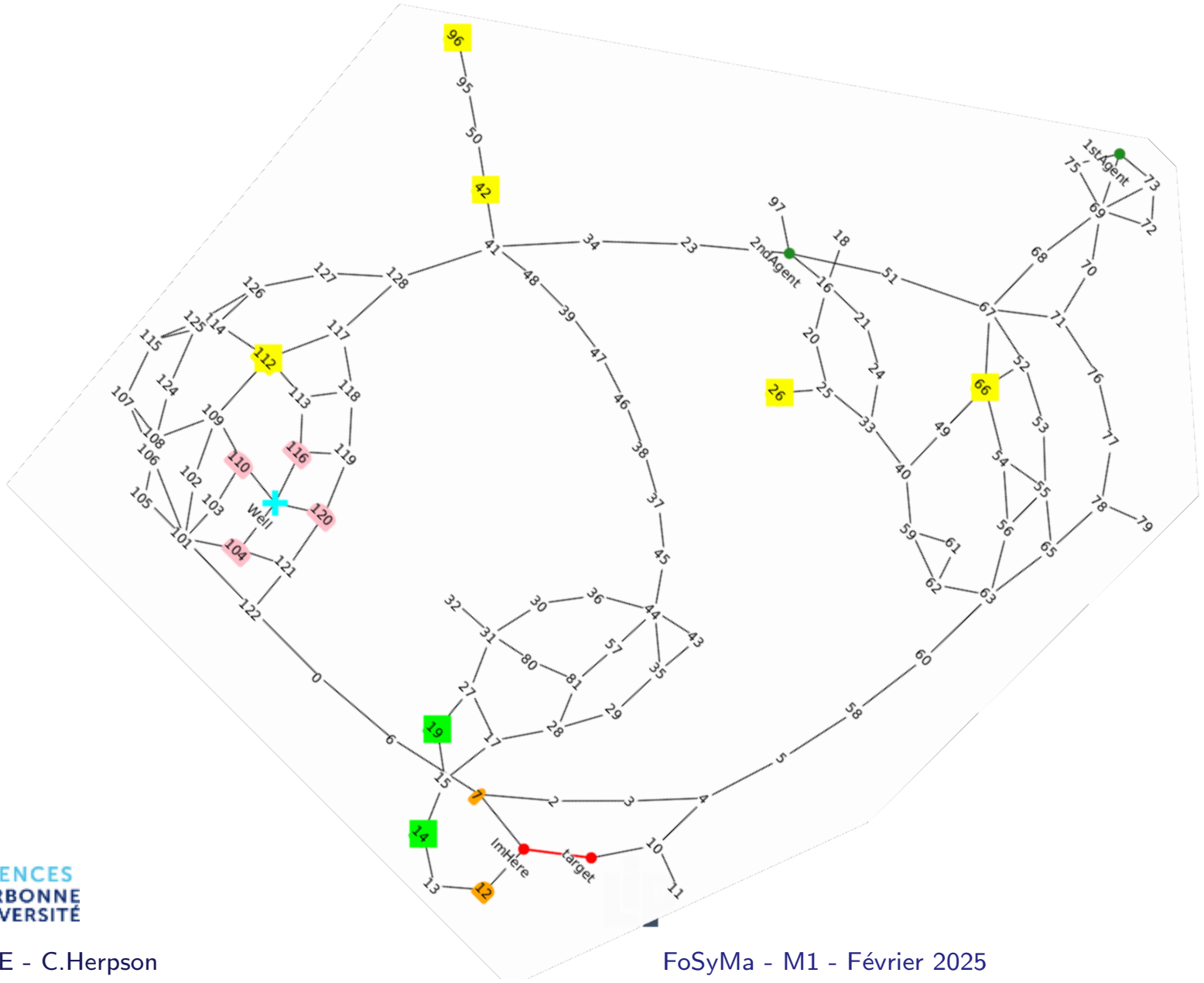
- **String getMyTreasureType()**: Type de ressources que l'agent peut récolter
- **Set<Couple<Observation,Integer>> getMyExpertise()** : Compétences de l'agent
- **boolean openLock(Observation o)**: Ouverture du coffre (type Gold ou Diamond) si les compétences requises sont présentes. Cette méthode agrège les compétences des agents connexes.
- **int pick()**: Permet de récupérer tout ou partie du trésor présent sur la position courante (selon la capacité d'emport de l'agent et l'état du coffre)
- **boolean EmptyMyBackPack(String agentSiloName)** Permet à l'agent de vider son sac dans le silo, sous réserve qu'il soit à portée.

Exemple d'utilisation de l'API pour les coffres

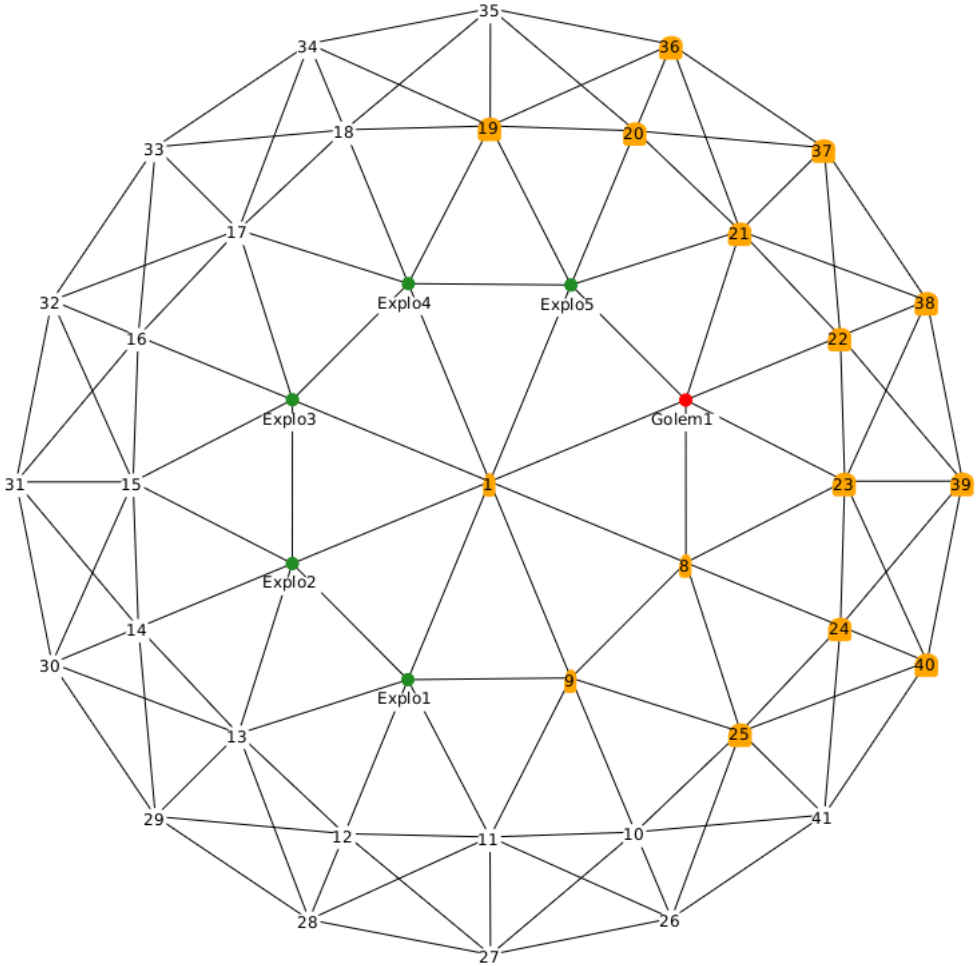
Cf le code fourni en Tp

```
                                Consider the following agent :
Entitytype: AgentCollect; TreasureType : Gold; diamondCapa: 0; goldCapa: 20; comReach: 3;
Expertise: [<Strength, 1>, <LockPicking, 1>]
                                                    ---

getMyTreasureType() --> Gold
getBackPackFreeSpace() --> 20
getMyExpertise() --> [<Strength, 1>, <LockPicking, 1>]

observe() --> [<96, [<Gold, 24>, <LockIsOpen, 0>, <Strength, 1>, <LockPicking, 2>]>, <95, []>]
I try to open the safe:  openLock(Observation.Gold) --> false
The agent tries to grab ressources: pick() --> 0
The remaining backpack capacity is: getBackPackFreeSpace()--> 20
State of the observations after picking :
[<96, [<Gold, 24>, <LockIsOpen, 0>, <Strength, 1>, <LockPicking, 2>]>, <95, []>]

... The agent moved several times...

observe() --> [<42, [<Gold, 60>, <LockIsOpen, 0>, <Strength, 0>, <LockPicking, 1>]>, <50, []>, <41, []>]
I try to open the safe: openLock(Observation.Gold) --> true
The agent tries to grab ressources: pick() --> 20
The remaining backpack capacity is: getBackPackFreeSpace()--> 0
State of the observations after picking :
[<42, [<Gold, 30>, <LockIsOpen, 1>, <Strength, 0>, <LockPicking, 1>]>, <50, []>, <41, []>]
```

Environnement : type 3

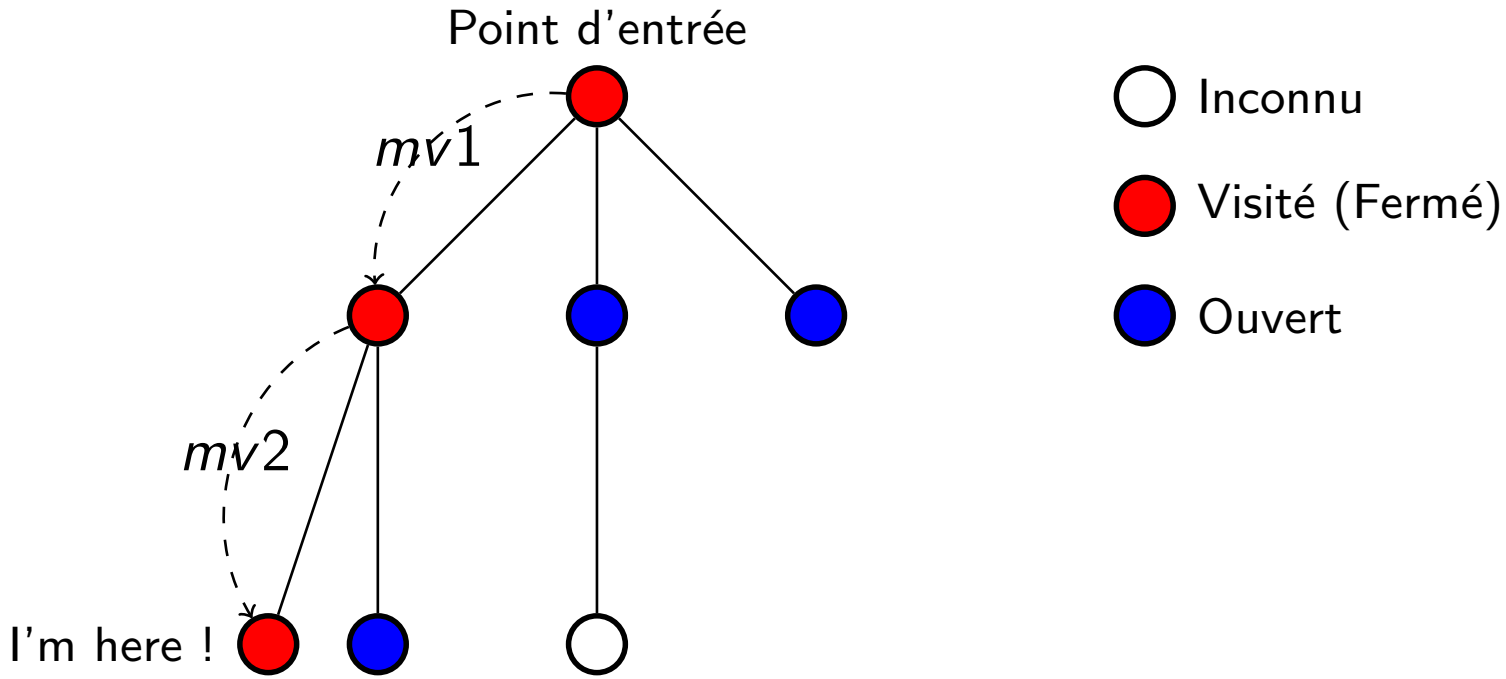


Environnement : type 3



Exploration (mono-agent) d'un graphe

BFS/DFS et équivalents ne sont pas utilisables tels quels..



Exploration (multi-agent) d'un graphe

Partage d'information
Coordination
Pérennité de l'information échangée ?
Cas des interblocages...

Collecte efficace et équitable des trésors

Maximiser la récolte de façon équitable des différentes ressources en un temps fini

- Topologies : Arbre, Graphe
- Rayons de communication : limités, potentiellement hétérogènes
- Nombre d'agents formant l'équipe : X
- Types de ressources : 2
- Types de ressources récoltables par chaque agent : 1
- Nombre d'agents adverses déplaçant et récoltant également des ressources : 2

Environnement non-stationnaire et partiellement observable.

Patrouille et chasse multi-agent

Garantie de trouver et bloquer le ou les adversaires en un temps fini tout en minimisant le nombre d'agents formant la patrouille ?

- Topologies : Arbre, Graphe
- Rayons de communication : limités
- Nombre d'agents formant la patrouille : X
- Nombre d'adversaires : Y

Aujourd'hui pas de solutions optimales connues pour toutes les combinaisons possibles.

Ressources

Dedale

https://dedale.gitlab.io

Remontée de bug et fonctionnalités

https://gitlab.com/dedale/dedale/-/issues/new

Discord

https://discord.gg/JZVz6sR