

TP n°2

Projet Dedale - Exploration de l'Environnement

Dans sa configuration actuelle, les bibliothèques exploitées par le projet sont *Jade* (plateforme agent), *Junit* (tests-unitaires), *GraphStream*¹ (graphes dynamiques) et *Dedale* (environnement du projet). La bibliothèque *Dedale* sera progressivement mise à jour durant le semestre, à mesure que de nouvelles fonctionnalités vous seront nécessaires.

Comme indiqué dans le sujet du projet, l'environnement dans lequel vos agents vont évoluer est un graphe non-orienté. Les nœuds du graphe représentent les pièces, et les arrêtes les couloirs qui les lient. Deux agents ne peuvent être simultanément sur un même nœud.

1 Démarrage de la plateforme fournie

- Sur les machines de la faculté : téléchargez et importez dans eclipse l'archive disponible sur moodle, *DedaleEtuFull* (projet Java).
 - Sur votre machine personnelle : <https://dedale.gitlab.io/page/tutorial/install/>
1. Exécutez la classe *Principal.java*. La configuration courante démarre un unique agent *DummyExploAgent* sur un environnement de type grille généré aléatoirement. Il est rattaché à l'environnement de manière transparente, puis doté de deux comportements :
 - **RandomWalkBehaviour** : Permet à l'agent d'évoluer dans – et d'interagir avec – l'environnement en se déplaçant de manière aléatoire.
 - **SayHello** : Tente de contacter un autre agent pour lui transmettre sa position courante. La distance entre l'émetteur et le récepteur conditionne la réussite de l'acheminement.
 2. Après avoir lu <https://dedale.gitlab.io/page/tutorial/configureenv/> modifiez la configuration du générateur aléatoire de carte dans le fichier *ConfigurationFile.java* de façon à ce que l'environnement généré soit de taille 20, de type dogoronev et que seul de l'or soit présent sur la carte. Relancez la plateforme.

2 Principales caractéristiques de la plateforme

2.1 Création d'un agent et interaction avec l'environnement

Comme la semaine passée, la création d'un agent est déclenchée dans la classe *Principal.java*, au sein de la méthode `createAgents()` par le passage en paramètre de la classe de l'agent souhaité.

```
//4) Give the class name of your agent to let the system instantiate it  
ag=createNewDedaleAgent(c, agentName, DummyMovingAgent.class.getName(), entityParameters2);
```

1. <http://graphstream-project.org/>

Voici le code de l'agent *DummyExploAgent* :

```
public class DummyMovingAgent extends AbstractDedaleAgent{

    /**
     * This method is automatically called when "agent".start() is executed.
     * It considers that the agent is launched for the first time and :
     * 1) set the agent attributes , 2) add the behaviours
     */
    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));
    }
}
```

A l'instar du *DummyMovingAgent*, vos agents devront étendre *AbstractDedaleAgent* et respecter la structure de ce `setup()` en ajoutant leurs comportements initiaux au sein de la liste `lb`.

AbstractDedaleAgent fourni à vos agents les méthodes d'interaction avec les autres agents et l'environnement.

StartMyBehaviours prend en charge les problèmes de distribution que nous verrons plus tard dans le semestre et s'assure que vos comportements ne démarrent qu'une fois l'agent effectivement déployé sur la carte.

2.2 API de l'environnement

Voici les principales méthodes de l'API :

- ***String getCurrentPosition()*** Retourne la position courante de l'agent
- ***List<Couple<String,List<Attribute>>> 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)***
Permet à votre agent de se déplacer dans l'environnement jusqu'à la position fournie en paramètre (si atteignable). Afin de garantir la cohérence de votre agent et de vous faciliter le travail d'implémentation, **cette fonction, lorsqu'elle est appelée, doit être la dernière méthode de votre comportement.**
- ***void sendMessage(ACLMessage msg)***
Fonction d'envoi de message qui gère le rayon de communication de agents. **Vous devez utiliser exclusivement celle-ci.**
- ***int getBackPackFreeSpace()***
Retourne l'espace libre dans le sac à dos de l'agent.
- ***int pick()***
Permet de récupérer tout ou partie du trésor présent sur la position courante (en fonction de la capacité d'emport de l'agent et du respect des conditions d'ouverture du coffre)
- ***String getMyTreasureType()***
Permet à l'agent de connaître son type
- ***boolean EmptyMyBackPack(String agentSiloName)***
Permet à l'agent de vider son sac dans le silo, sous réserve qu'il soit à porté.

Les quatre dernières fonctions (relatives aux trésors) ne seront pas utilisées durant les premières semaines, et d'autres seront ajoutées ultérieurement. La Javadoc vous donnera le détail complet de l'ensemble de ces fonctions, et le code du *DummyExploAgent* illustre l'utilisation des fonctions ci-dessus.

3 Exploration et représentation de l'environnement

3.1 Exploration mono-agent

Dans la classe *Principal.java*, méthode `createAgents`, remplacez la création de l'agent `DummyMovingAgent` par `ExploreSoloAgent`. Rebaseculez sur une carte de type grille de taille 10 puis exécutez le code.

```
//4) Give the class name of your agent to let the system instantiate it
//ag=createNewDedaleAgent(c,agentName,DummyMovingAgent.class.getName(),entityParameters2);
ag=createNewDedaleAgent(c,agentName,ExploreSoloAgent.class.getName(),entityParameters2);
```

`ExploreSoloAgent` explore son environnement et construit sa propre représentation de celui-ci en vous donnant un aperçu temps réel de ses connaissances². Lisez le code de l'agent, son comportement et sa structure de représentation mémoire.

2. Un exemple simpliste de création visuelle de graphe est disponible dans le package `test`

1. Pourquoi est-il préférable de réaliser un parcours en profondeur plutôt qu'en largeur ?
2. Quand celui-ci s'arrête t'il ?
3. Explo1 ne s'intéresse qu'à la topologie. D'autres éléments seront nécessaires lorsque votre agent ira récolter des ressources. Listez-les puis proposez une représentation mémoire adaptée.

3.2 Exploration multi-agent

En vous inspirant (ou pas) de Explo1, proposez un mécanisme permettant à plusieurs agents d'explorer collectivement **et intelligemment** l'environnement. En particulier, réfléchissez à la façon la plus pertinente de représenter et partager l'information.

La solution que vous proposerez devra être en mesure de fonctionner quel que soit le nombre d'agents de votre équipe et le rayon de communication de ceux-ci.

Réfléchissez consciencieusement aux différentes situations que vos agents sont susceptibles de rencontrer, dessinez le protocole de communication envisagé et identifiez les rôles associés. Analysez les différentes façon de décomposer ceux-ci en comportements en tenant compte de l'impact des différentes solutions sur l'information à représenter et partager entre les comportements d'un agent, ainsi qu'entre les agents eux-mêmes (complexité en temps, mémoire, nombre de messages, taille des messages,...).

On rappelle que le contenu d'un message entre agent n'est pas limité à une chaîne de caractère. L'utilisation de la méthode `msg.setContentObject(Object o)` vous permet de transmettre tout objet **sérialisable**.

Déploiement de plusieurs agents sur la carte

1. Lisez <https://dedale.gitlab.io/page/tutorial/deployagents/>.
2. Dans la classe *Principal.java*, `createAgents()`, décommentez l'agent `explo2`.
3. Ouvrez le fichier *ressources/monoAgent-entities* et créez en une copie. Modifier celle-ci de façon à autoriser la création de `Explo2` puis pointez dessus dans *ConfigurationFile.java*
4. Lancez la plateforme. Si vous avez fait ce qu'il faut, vos deux agents se promènent sur la carte. à vous de faire qu'ils l'explorent correctement.