SID Primavera 2022
Práctica
Versión 1 (04/04/2022)

Sergio Álvarez
Ulises Cortés

Abril – Junio 2022

1. Introdución

En esta práctica, se pide implementar un agente inteligente situado con cierto grado de autonomía que pueda trabajar de manera estable y coordinada con otros agentes en un entorno multi-agente con recursos públicos limitados. El agente tendrá dos objetivos: por un lado, (a) deberá maximizar los recursos recogidos por el conjunto de todos los agentes; por el otro, (b) deberá maximizar su empeño individual, que dependerá de su rol.

La evaluación consistirá en la ejecución de los agentes de todos los grupos de laboratorio en diferentes escenarios aleatorios.

2. Resumen del escenario

En esta práctica trabajaremos con entornos que tendrán cierto nivel de aleatoriedad para motivar que los agentes implementados tengan que *adaptarse* al entorno. Cada ejecución del escenario dependerá de un conjunto amplio de parámetros cuyo valor será asignado de manera aleatoria. Algunos ejemplos son: el tamaño del mapa, la distancia de percepción, el rol de cada agente, los recursos disponibles, la duración de la ejecución, entre otros parámetros.

El mundo está circunscrito a un grafo conexo $G = \langle V, E \rangle$ al que denominaremos *mapa*. En un mapa tenemos los diferentes elementos del entorno: agentes, recursos (oro o diamantes) y obstáculos (pozos). Un ejemplo de dicho mapa se puede ver en la Figura 1.

Hay dos tipos de recursos disponibles en el mapa: oro y diamantes. Cada uno de los nodos con recursos contiene una cantidad inicial aleatoria no renovable de un (sólo un) tipo. Los recursos están protegidos por una cerradura que requiere un nivel específico de habilidad para ser abierta.

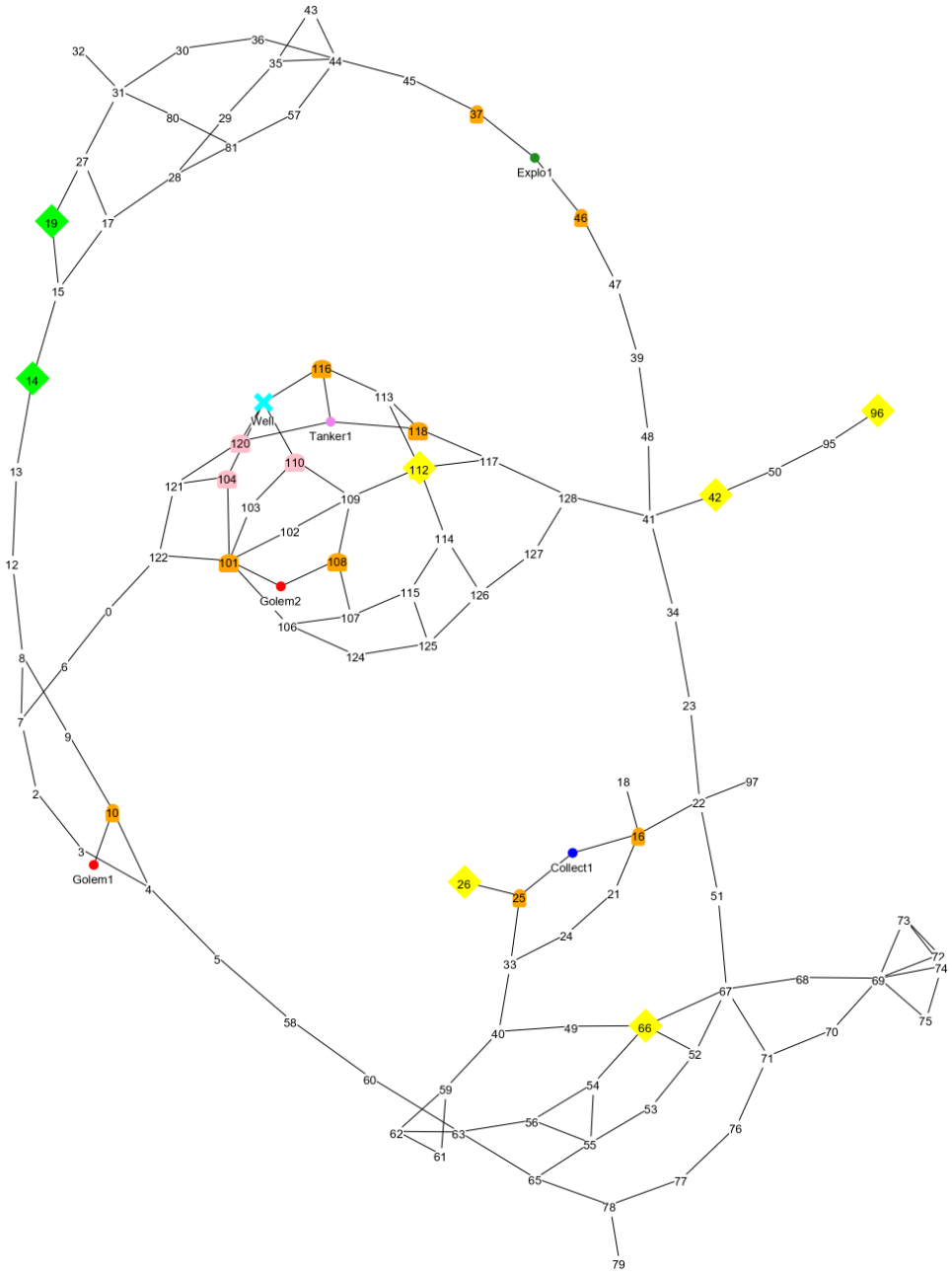

Figura 1: Ejemplo de mapa. En amarillo y verde, nodos que contienen oro y diamantes respectivamente. Los puntos verde, azul y rosa representan agentes explorador, recolector y almacenamiento. El punto rojo representa un gólem. Los círculos naranjas representan el rastro de los agentes, que permite detectar su presencia. La cruz azul representa un pozo, detectable por el viento a su alrededor representado por los círculos rosa.

Alternativamente, los nodos que no contienen recursos pueden tener un pozo (*well*). Si un agente se mueve a un nodo con un pozo, automáticamente muere y desaparece de la plataforma de agentes. Este agente no podrá volver a instanciarse en la plataforma durante el transcurso de la ejecución en marcha.

Los nodos que no contienen recursos ni pozos se consideran vacíos. En cada nodo, independientemente de si contiene un recurso o no, sólo puede haber un agente a la vez. Los agentes no pueden saltar por encima de otros agentes para evitar el bloqueo producido por un agente localizado en un nodo.

Un agente situado en el entorno empieza en un nodo aleatorio y cumple con un rol (y sólo uno en cada ejecución) también aleatorio de entre los siguientes:

- Explorador (*explorer*): no puede recoger recursos ni almacenarlos.
- Recolector (*collector*): dependiendo de la configuración inicial, puede recoger uno de los dos tipos de recurso y tiene un nivel específico de habilidad para abrir cerraduras que protejan el tipo de recurso correspondiente. La capacidad de almacenamiento de recursos es limitada. Además, puede entregar recursos previamente recogidos a agentes de tipo almacenamiento que estén en un nodo contiguo.
- Almacenamiento (*tanker* o *silo*): no puede recoger recursos directamente del escenario pero puede recibirlos de los recolectores. Puede almacenar recursos de uno o de ambos tipos de recurso, dependiendo de la configuración inicial. La capacidad de almacenamiento de recursos es también limitada, pero por lo general esta capacidad será considerablemente mayor que la capacidad de un agente recolector.

Independientemente del rol, los agentes pueden en cada momento consultar el conjunto de percepciones accesibles desde el nodo en el que están. El rango de visibilidad (distancia en nodos) dependerá de la configuración inicial del entorno. Por lo tanto, para saber qué hay más allá de dicho rango, los agentes deben moverse por el mapa o comunicarse con otros agentes.

Los agentes a implementar han de poder cumplir dos o tres[1] de estos roles. Qué rol está asignado a un agente en cada ejecución depende de la configuración inicial del entorno.

Adicionalmente, puede haber un número arbitrario (que puede ser 0) de agentes controlados por el entorno con el rol de *golem*. Estos agentes se mueven de manera aleatoria por el mapa, recogiendo cantidades aleatorias de recursos (pudiendo incluso vaciar nodos) y colocándolos en nodos vacíos al azar.

Todos los agentes, incluyendo los *golems*, dejan un rastro a su alrededor que es visible mediante percepciones. El radio (en número de nodos) de este rastro depende de la configuración del entorno y puede variar para cada agente de forma individual.

Este escenario implica dos tipos de coordinación: competición, ya que el objetivo de cada agente es maximizar su rendimiento con respecto a los demás

[1] Según el tamaño del grupo de alumnos.

agentes; y cooperación, con el objetivo de maximizar los recursos obtenidos del entorno. Por lo tanto, será necesario que los agentes se comuniquen.

La métrica para evaluar el rendimiento cooperativo será minimizar la cantidad de recursos en el mapa remanentes al final de la ejecución.

Las métricas para evaluar el rendimiento de los agentes en base a su rol serán:

- Explorador: maximizar la proporción entre nodos visitados en total (al cuadrado) y la desviación estándar entre los diferentes valores de número de visitas a cada nodo. Con esto se intenta que el agente se mueva rápido a la vez que abarca una porción considerable del mapa:

$$\text{maximizar } \frac{\left(\sum_{\forall v \in V} times\_explored(v)\right)^2}{1+std(times\_explored(v)|v \in V)}$$

  donde:

  - *times_explored* es una función que devuelve, para un nodo determinado, el número de veces que el agente ha visitado ese nodo,
  - *std*() es una función que calcula la desviación estándar de una lista de valores.

- Recolector: maximizar total de recursos recogidos durante todo el transcurso de la ejecución.

- Almacenaje: maximizar cantidad de recursos transportados al final de la ejecución.

Como se describirá en la rúbrica, es posible entregar o bien un agente situado que implemente todo el razonamiento, o bien dos agentes, uno situado puramente reactivo y otro agente deliberativo (usando BDI4JADE) que gestionará la mayor parte del razonamiento y que se comunicará con el agente situado, usándolo como *proxy*.

2.1. Mecanismos de comunicación entre agentes

En cualquier momento, un agente situado puede enviar un mensaje a uno o varios de los agentes también situados en la plataforma. La comunicación entre agentes situados debe hacerse utilizando el método *sendMessage* de la clase *AbstractDedaleAgent*.

En caso de implementar un agente BDI para gestionar la parte deliberativa, este agente únicamente podrá comunicarse con su agente situado correspondiente, usando el método *send* de la clase *Agent* de JADE. Los agentes BDI no podrán comunicarse entre si.

El uso de cualquier otro mecanismo de comunicación entre los agentes supondrá una evaluación con un 0 en la nota de laboratorio.

En esta práctica no es obligatorio utilizar la ontología para comunicarse, pero podéis poneros de acuerdo entre grupos para acordar una si consideráis que eso puede ayudar en la creación e interpretación del contenido de los mensajes. Podéis elegir libremente los protocolos de interacción y los lenguajes de contenido que queráis utilizar siempre y cuando os pongáis de acuerdo para que la comunicación sea posible. No es obligatorio (aunque sí muy recomendable si queréis que los agentes se puedan coordinar correctamente) que todos los agentes se comuniquen de la misma manera.

2.2. Dinámica del entorno

El entorno no es discreto, por lo que no hay un control de modificaciones sobre el entorno basado en ticks de reloj. Por lo tanto, las acciones de los agentes son asíncronas y se pueden realizar en cualquier momento. Esto puede causar que algunas acciones de los agentes fallen (por ejemplo, se envía un mensaje a un agente que justo ha abandonado el rango de comunicación), por lo que los agentes deberán tener esto en cuenta.

2.3. Repertorio de acciones disponibles para los agentes

Un agente situado puede realizar las siguientes acciones sobre el entorno:

- *String getCurrentPosition()*: Devuelve la posición actual del agente.
- *List observe()*: Devuelve el conjunto de percepciones disponibles.
- *boolean moveTo(String myDestination)*: Si el nodo myDestination es accesible desde la posición actual, mueve el agente a este nodo.
- *void sendMessage(ACLMessage msg)*: Envía un mensaje, si el destinatario está en rango.
- *String getMyTreasureType()*: Devuelve el tipo de recurso que el agente puede recoger.
- *Set getMyExpertise()*: Devuelve los niveles de habilidad del agente.
- *int getBackPackFreeSpace()*: Devuelve el espacio libre del agente para recoger recursos.
- *boolean openLock(Observation o)*: Abrir la cerradura del recurso del nodo actual, siempre y cuando el agente tenga suficiente nivel de habilidad.
- *int pick()*: Recoger la máxima cantidad de recurso posible del nodo actual (mínimo entre capacidad del agente y recursos remanentes), siempre y cuando el agente pueda recoger este tipo de recurso.
- *boolean EmptyMyBackPack(String agentSiloName)*: Descargar los recursos transportados por el agente en el agente de almacenamiento correspondiente al nombre dado, siempre y cuando esté en un nodo vecino.

Los agentes pueden intentar ejecutar todas las acciones pero su éxito puede depender de si su rol permite que tengan efecto sobre el entorno o no. Por ejemplo, el método *pick()* siempre retornará 0 para un agente explorador. El agente no recibe información sobre su rol cuando empieza el escenario, por lo que su estrategia debería incluir la detección del rol basándose en los resultados de sus acciones.

3. Plazos y evaluación

Durante las sesiones de laboratorio os introduciremos a las tecnologías que obligatoriamente tendréis que usar: Dedale[2], JADE[3], BDI4JADE[4] y Apache Jena[5], utilizando Java 11. No hay ninguna limitación respecto a los algoritmos que podéis implementar para vuestros agentes. Si para estos algoritmos creéis conveniente utilizar librerías, consultádnoslo con antelación suficiente y de manera justificada para 1) recibir aprobación y 2) comprobar que no haya incompatibilidades con otras librerías propuestas por otros grupos, ya que la ejecución se hará por nuestra parte de manera local por lo que tendremos que importarlas todas en la misma instancia de JVM.

Si sois un grupo de 4, deberéis implementar un agente que sea capaz de detectar y ejecutar los 3 roles de agente (explorador, recolector, almacenamiento). En caso contrario, deberéis escoger e implementar 2 de ellos. Para planificar adecuadamente el conjunto de configuraciones con las que se evaluarán los agentes y evitar posibles desequilibrios entre grupos, tenéis que enviar los roles escogidos antes del 30 de abril a salvarez@cs.upc.edu.

La fecha límite de entrega de la práctica será el 6 de junio via Racó y consistirá en:

- El código del agente o agentes implementados (ver rúbrica).
- Un documento (.pdf) en el cual se explique:
  - La estrategia diseñada para cada rol.
  - Cómo se traducen esas estrategias en los diferentes elementos del agente: arquitectura, objetivos, planes, behaviours, protocolos de comunicación, etc.
  - Cómo habéis organizado las tareas a hacer y cómo las habéis distribuido.

La nota base de la práctica se evaluará según la rúbrica descrita en la Figura 2. Esta nota tendrá un valor máximo de 10. Aparte de esta nota base, existe la posibilidad de sumar puntos extra que permitirían tener una nota mayor de 10:

[2] https://dedale.gitlab.io
[3] https://jade.tilab.com
[4] https://github.com/ingridnunes/bdi4jade
[5] https://jena.apache.org

Criterio	Peso	Expectativa
Implementación del agente	50.00 %	El agente funciona bien y no falla en la plataforma. El agente implementa una estrategia adecuada para cumplir con sus objetivos individuales. El agente es proactivo a la vez que reactivo.
BDI	20.00 %	El agente situado en el entorno es meramente un proxy para obtener percepciones y ejecutar acciones sobre el entorno. Además del agente situado, se entrega un agente implementado con BDI4JADE para implementar el razonamiento, utilizando el agente situado como proxy (sensor/actuador). La descomposición en objetivos y planes está correctamente diseñada para implementar la estrategia del agente para cada uno de los roles que implementa.
Coordinación	15.00 %	El comportamiento del agente persigue establecer algún mecanismo de coordinación con otros agentes.
Ontología	10.00 %	Se utiliza una ontología para hacer la revisión de creencias del agente.
Comunicación	5.00 %	La comunicación con otros agentes es correcta. La generación e interpretación de los mensajes es correcta. Se utilizan protocolos FIPA de manera adecuada para cada tipo de conversación.

Figura 2: Rúbrica de evaluación de la práctica

- 1 punto extra para todos los grupos si en todos los escenarios sin *golems* los agentes consiguen recoger todos los recursos.
- 2 puntos extra al 25 % de grupos (redondeando hacia abajo) que implementen los agentes que consigan el mejor rendimiento en el rol asignado en cada escenario, en más escenarios.

La nota final de laboratorio, sumando la nota base y los puntos extra, se multiplicará por 0.3 para el cálculo de la nota final de la asignatura.