Esplorazione di ambienti 2D tramite agenti eterogenei

Andrea Monaldini

13 gennaio 2006

1 Obiettivi

L'obiettivo del progetto [3] è quello di sviluppare un MAS [4] capace di effettuare l'esplorazione di un ambiente virtuale per mezzo di agenti software. Nelle sezioni successive dell'elaborato si suppone che l'ambiente virtuale sia composto da uno spazio bidimensionale che rappresenta un territorio da esplorare. Si ipotizza che in questo ambiente siano presenti alcune zone inaccessibili ad alcuni tipi di "esploratori", ma agibili ad altri.

2 Analisi

L'esistenza di aree agibili a determinate classi di esploratori e non ad altre implica l'esistenza di almeno due tipologie di agente.

Lo spazio da esplorare è caratterizzato da una forma semplice e non variabile: gli agenti che sono situati in questo ambiente devono avere dei comportamenti, da esso dipendenti, che gli permettano di muoversi ed operare in modo corretto.

Si considera utile attribuire diversi compiti ai diversi tipi di agenti: ciò consente di rappresentare una situazione reale nella quale un tipo di agente hardware "economico" svolge operazioni più rischiose e meno complesse, mentre un altro tipo, più avanzato, svolge le operazioni più delicate.

La necessità di esplorare lo spazio bidimensionale rende indispensabile l'individuazione di un algoritmo che porti gli agenti-esploratori a non trascurare porzioni di spazio. È evidente come emerga la possibilità di impostare l'esplorazione come un compito di ciascun agente oppure dell'intera comunità. Nel secondo caso

è necessario prevedere, anche tra agenti dello stesso tipo, particolari meccanismi di coordinazione che rendano efficiente l'esplorazione.

La scelta di attribuire compiti diversi alle diverse tipologie di agenti rende indispensabile implementare modelli di coordinazione anche tra agenti eterogenei, al fine di raggiungere i risultati desiderati con la collaborazione di tutte le classi di entità (o di un loro membro).

3 Progetto

Lo sviluppo del sistema software ad agenti prevede la progettazione dell'ambiente virtuale che rappresenta lo spazio da esplorare, degli agenti che vivranno ed opereranno in questo spazio e dei meccanismi di coordinazione per supportare l'esplorazione.

3.1 Ambiente virtuale

L'ambiente virtuale deve rappresentare una superficie reale da esplorare: una griglia bidimensionale permette di effettuare questa rappresentazione. La griglia deve essere caratterizzata dalle dimensioni e composta da celle, univocamente identificabili, dotate di diversi attributi (figura 1):

- muro
- proibita
- accessibile

Tramite gli attributi si deve poter configurare l'accessibilità della cella, al fine di creare celle accessibili a tutti gli agenti, ad un solo tipo o a nessuno (rappresentazioni di muri / delimitatori della griglia).

3.2 Agenti esploratori

Come emerso in fase di analisi, per il funzionamento del sistema è indispensabile che esistano almeno due diverse tipologie di agenti. Per lo sviluppo del sistema si è quindi scelto di progettare due diversi tipi di agenti, ai quali attribuire compiti diversi in funzione della classe di appartenenza. Tali agenti saranno la versione software di ipotetici "rover" utilizzati per l'esplorazione di un ambiente reale. Essi

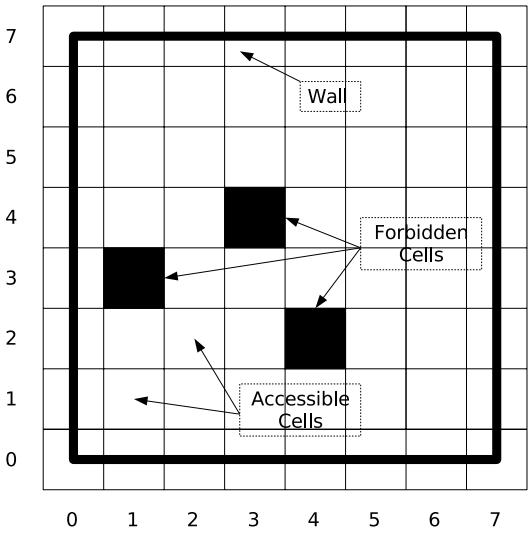

Figura 1: Esempio di ambiente virtuale nel quale operano gli agenti esploratori.

saranno quindi caratterizzati dalla possibilità di conoscere la propria posizione, di muoversi di moto rettilineo in direzione frontale e di eseguire rotazioni di 90°.

3.2.1 Semplici

Questi agenti rappresentano gli esploratori veri e propri: il loro compito è quello di esplorare le celle della griglia a loro accessibili e di richiedere l'intervento di agenti avanzati sulle celle nelle quali i primi non possono entrare.

Gli agenti semplici devono essere dotati di un flusso interno di controllo che gli consenta di esplorare le celle della griglia tenendo conto delle azioni precedentemente svolte e dell'operato di altri agenti dello stesso tipo.

Data la tipologia di task assegnato agli agenti di questa classe e alla sua specificità si è deciso di rappresentarlo implicitamente: tale scelta limita notevolmente la flessibilità degli agenti ed influenza radicalmente il loro design. Si è comunque effettuata questa scelta in quanto non si reputa la flessibilità (in questi termini) particolarmente rilevante in questo contesto.

I passi fondamentali sui quali si basa l'esplorazione dello spazio virtuale effettuato dagli agenti semplici sono rappresentati in figura 2.

Il semplice algoritmo di esplorazione si basa sul principio di passare da celle

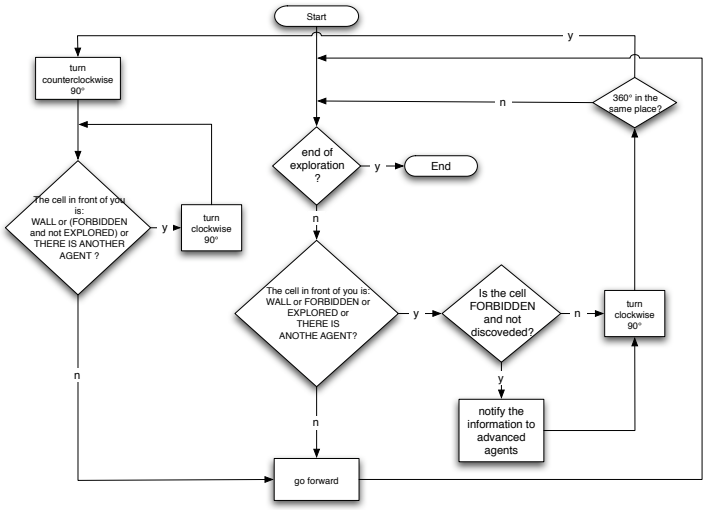

Figura 2: Schema di funzionamento agente esploratore semplice

già esplorate solo in caso di impossibilità di passare da zone non ancora scoperte e di ritornare sui propri passi solo se ciò è indispensabile. Nel caso in cui le possibilità siano tornare sulla strada appena percorsa o andare su una cella già esplorata si predilige la seconda scelta.

È evidente che quando un agente semplice rileverà una cella inaccessibile dovrà notificare l'informazione ad un diverso tipo di agente, capace di accedervi. Analogamente, ogni agente dovrà reificare l'informazione di esplorazione di una cella in modo tale da permettere agli altri agenti di conoscere lo stato di quella specifica porzione di spazio. È inoltre indispensabile che questi agenti possano conoscere le posizioni reciproche, al fine di evitare di occupare inutilmente la stessa area.

3.2.2 Avanzati

Gli agenti avanzati rappresentano una forma più evoluta di esploratore il cui compito è quello di perlustrare le celle inaccessibili agli agenti semplici. Uno schema di funzionamento di questo tipo di agente è presente in figura 3.

Essi sono normalmente fermi ed è l'individuazione di una cella inaccessibile ad attivarne il funzionamento: dopo aver calcolato un percorso dal punto di partenza a quello di destinazione (avvalendosi unicamente di celle già esplorate

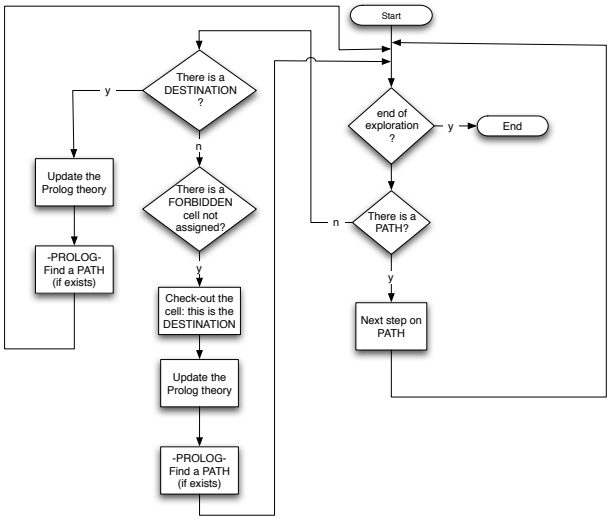

Figura 3: Schema di funzionamento agente esploratore avanzato

dal primo tipo di agente o da agenti avanzati) partono in quella direzione. Considerando che questo tipo di agente si può muovere solo su celle già controllate è necessario che esista un percorso non partizionato tra partenza e arrivo affinché si possa muovere.

Al fine di presentare anche forme semplici di reattività degli agenti e non solo quelli di pianificazione, si configura come compito degli agenti avanzati che hanno una meta ma non ancora un percorso a causa della mancanza di continuità di celle esplorate, quello di ricercare, ad ogni passo di simulazione, l'eventuale esistenza di un percorso fruibile grazie alle nuove celle rilevate agibili.

3.3 Coordinazione

Si decide di basare la coordinazione degli agenti del sistema su tuple centres programmabili + ACC offerti da TuCSoN [1], assumendo quindi valido per il progetto il suo meta-modello e modello di coordinazione (figura 4).

Gli agenti semplici dovranno reificare l'informazione raccolta sulle celle esplorate in uno spazio di tuple: essi inseriranno tuple del tipo *cell(X,Y)* quando esplorano (entrano) in una cella libera, mentre sarà prodotta una tupla del tipo *forbiddenCell(X,Y)* quando individuano celle inaccessibili adiacenti a quella di appartenenza.

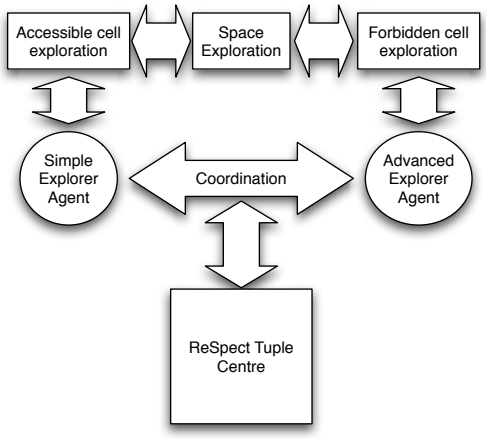

Figura 4: Coordinazione tra agenti eterogenei, con task sociale suddiviso in due task individuali.

Gli agenti avanzati consumeranno le tuple *forbiddenCell(X,Y)* facendosi carico dell'esplorazione di queste celle e si baseranno sulle tuple *cell(X,Y)* per pianificare il percorso tra la posizione attuale e quella di destinazione. In risposta all'esplorazione di una cella dichiarata inagibile sarà prodotta una tupla *cell(X,Y)*.

La figura 5 mostra graficamente la coordinazione tra agenti semplici ed avanzati necessaria per effettuare l'esplorazione dell'ambiente.

La terminazione del processo di esplorazione è garantito da una sincronizzazione a barriera ottenuta tramite programmazione della reazione del tuple centre (*ReSpecT*) all'inserimento delle tuple *cell(X,Y)*: quando il numero di queste tuple è pari al numero di celle presenti nella griglia l'esplorazione è terminata e si ha, come reazione del tuple centre, l'inserimento di una tupla *virtualSpaceExplored*.

La posizione degli agenti nella griglia sarà indicato da tuple *explorerAt(X, Y, AgentID)*.

3.4 Prolog

Al fine di semplificare la pianificazione del percorso dalla cella di appartenenza a quella di destinazione si decide di rendere disponibile agli agenti avanzati un motore prolog [2] e una opportuna teoria.

Inizialmente la teoria conterrà solo le clausole necessarie a:

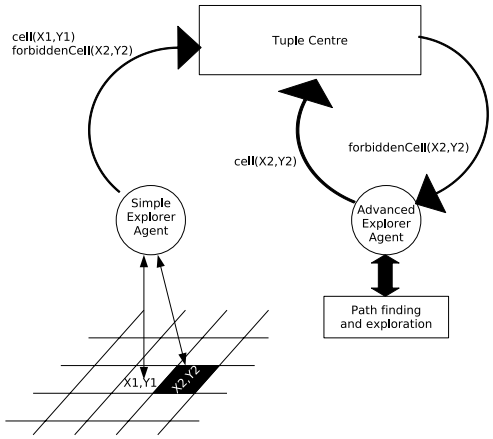

Figura 5: Coordinazione tra agenti semplici ed avanzati necessaria per raggiungere la completa esplorazione dell'ambiente.

- calcolo del percorso (dati inizio e fine)
  - *findPath(A,B,Path) :- route(A,B,[A],Q), reverse(Q,Path).*
- calcolo del percorso tra celle intermedie
  - *route(A,B,P,[B|P]):- link(A,B), !.*
  - *route(A,B,Visited,Path):- link(A,C), not(member(C,Visited)), route(C,B,[C|Visited],Path).*
- individuazione di celle adiacenti (data una cella indicare quali sono adiacenti)
  - *link(Cell1,Cell2):- fromCell(Cell1,[-1,0],Cell2); fromCell(Cell1,[1,0],Cell2); fromCell(Cell1,[0,-1],Cell2); fromCell(Cell1,[0,1],Cell2).*
  - *fromCell(cell(X,Y), [Xoffset,Yoffset], cell(X1,Y1)):- X1 is X + Xoffset, Y1 is Y + Yoffset, cell(X1,Y1).*

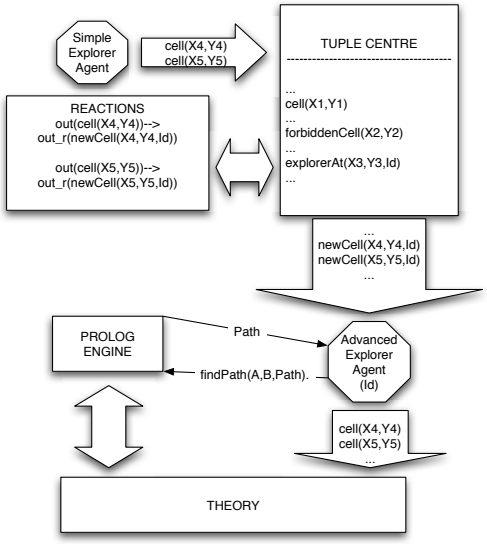

Figura 6: Integrazione del motore prolog nell'agente avanzato.

Per permettere al motore prolog di eseguire il suo compito si deve creare un meccanismo di aggiornamento della teoria in funzione delle nuove celle esplorate. Questo risultato si può ottenere per mezzo del funzionamento coordinato di agenti semplici - reazioni del tuple centre - agenti avanzati. Il meccanismo prevede che ad ogni inserimento di tuple *cell(X,Y)* il tuple centre reagisca creandone una copia *newCell(X,Y,Id)* per ogni Id di agente avanzato nel sistema. A loro volta, prima di effettuare un piano (quindi prima di utilizzare il motore prolog), gli agenti avanzati aggiorneranno la propria teoria consumando le tuple *newCell(X,Y,Id)*, rendendo quindi consistente la teoria con lo stato di esplorazione del sistema. Il funzionamento è riassunto in figura 6.

3.5 ReSpecT

Come già citato in precedenza, sono presenti casi in cui è opportuno che il tuple centre sia caratterizzato da particolari reazioni.

Il primo caso consiste nell'implementare una sincronizzazione "a barriera" che consenta di interrompere il lavoro degli agenti quando questo è terminato: ciò si ottiene inserendo inizialmente una tupla *exploredCell(0)* e una tupla *gridDimension(N)* il cui valore numerico viene incrementato ad ogni inserimento di tuple

cell(X,Y). Quando il valore di *exploredCell(N)* diventa pari a quello indicato da *gridDimension(N)* viene inserita la tupla *virtualSpaceExplored*. Su quest'ultima si baseranno gli agenti per eseguire la propria terminazione.

Il secondo caso prevede di creare tuple *newCell(X,Y,Id)* all'inserimento delle analoghe *cell(X,Y)*. Questo risultato è ottenibile basandosi su una tupla *advancedAgentNumber(N)*, inserita inizialmente, che viene incrementata ad ogni arrivo di un esploratore avanzato (che dichiara il proprio inserimento nel sistema tramite la tupla *advancedAgent(Id))* e permette di assegnare un numero progressivo all'id di ogni agente. Basandosi sul numero progressivo e sul numero totale di agenti avanzati presenti è possibile creare una nuova tupla *newCell(X,Y,Id)* per ciascun Id di agente avanzato presente.

Si decide di basare su reazioni del tuple centre anche la gestione dei turni degli agenti nel sistema. Ciascun agente (id) attende il proprio turno (*in* della tupla *resource_token(id)*) e ne notifica il completamento restituendo il token (*out* della tupla *resource_token(id)*). Tramite opportune reazioni del tuple centre è possibile creare algoritmi di gestione dei turni complessi a piacere.

Riassunto configurazione reazioni:

- sincronizzazione a barriera
  - *reaction(out(cell(X,Y)), (in_r(exploredCell(N)), N1 is N+1, out_r(exploredCell(N1))))*.
  - *reaction(out_r(exploredCell(N)), (rd_r(exploredCell(N)), rd_r(gridDimension(N)), out_r(virtualSpaceExplored)))*.
- indicizzazione agenti avanzati e generazione delle tuple per la sincronizzazione delle loro teorie
  - *reaction(out(advancedAgent(X)), (in_r(advancedAgentNumber(N)), N1 is N+1, out_r(advancedAgentNumber(N1)), out_r(advancedAgent(X,N1))))*.
  - *reaction(out(cell(X,Y)), (rd_r(advancedAgentNumber(N)), rd_r(advancedAgent(Id,N)), out_r(newCell(X,Y,Id)), N1 is N-1, out_r(tmp(X,Y,N1))))*.
  - *reaction(out_r(tmp(X,Y,N)), (in_r(tmp(X,Y,N)), rd_r(advancedAgent(Id,N)), out_r(newCell(X,Y,Id)), N1 is N-1, out_r(tmp(X,Y,N1))))*.
  - *reaction(out_r(tmp(X,Y,0)), (in_r(tmp(X,Y,0))))*.

- gestione dei turni

  - *reaction(in(resource_token(Who)), (pre, in_r(tickets(N)), N1 is N + 1, out_r(tickets(N1)), out_r(turn(Who,N))))*.
  - *reaction(out_r(turn(Who,N)), (rd_r(current_turn(N)), out_r(resource_token(Who))))*.
  - *reaction(out(resource_token(Who)), (in_r(resource_token(Who)), in_r(turn(Who,N)), in_r(current_turn(N)), N1 is N+1, out_r(current_turn(N1))))*.
  - *reaction(out_r(current_turn(N)), (rd_r(turn(Who,N)), out_r(resource_token(Who))))*.

3.6 Riassunto del sistema progettato

In figura 7 è presente un riassunto grafico del sistema che si è progettato.

4 Implementazione

Per l'implementazione si rimanda al codice java, opportunamente commentato, e alla documentazione "javadoc" prodotta.

5 Risultati

In questa sezione si riportano i risultati ottenuti dal sistema di esplorazione a partire dalla configurazione iniziale dell'ambiente presente in figura 8.

In questa configurazione sono presenti due agenti esploratori semplici e due avanzati, posizionati in modo tale da mostrare una molteplicità di comportamenti del sistema:

- l'agente semplice in (3,1), orientato verso ovest, scoprirà prima la cella proibita (1,3) poi quella in (4,2)
- le due celle proibite saranno scelte come destinazione dai due agenti avanzati

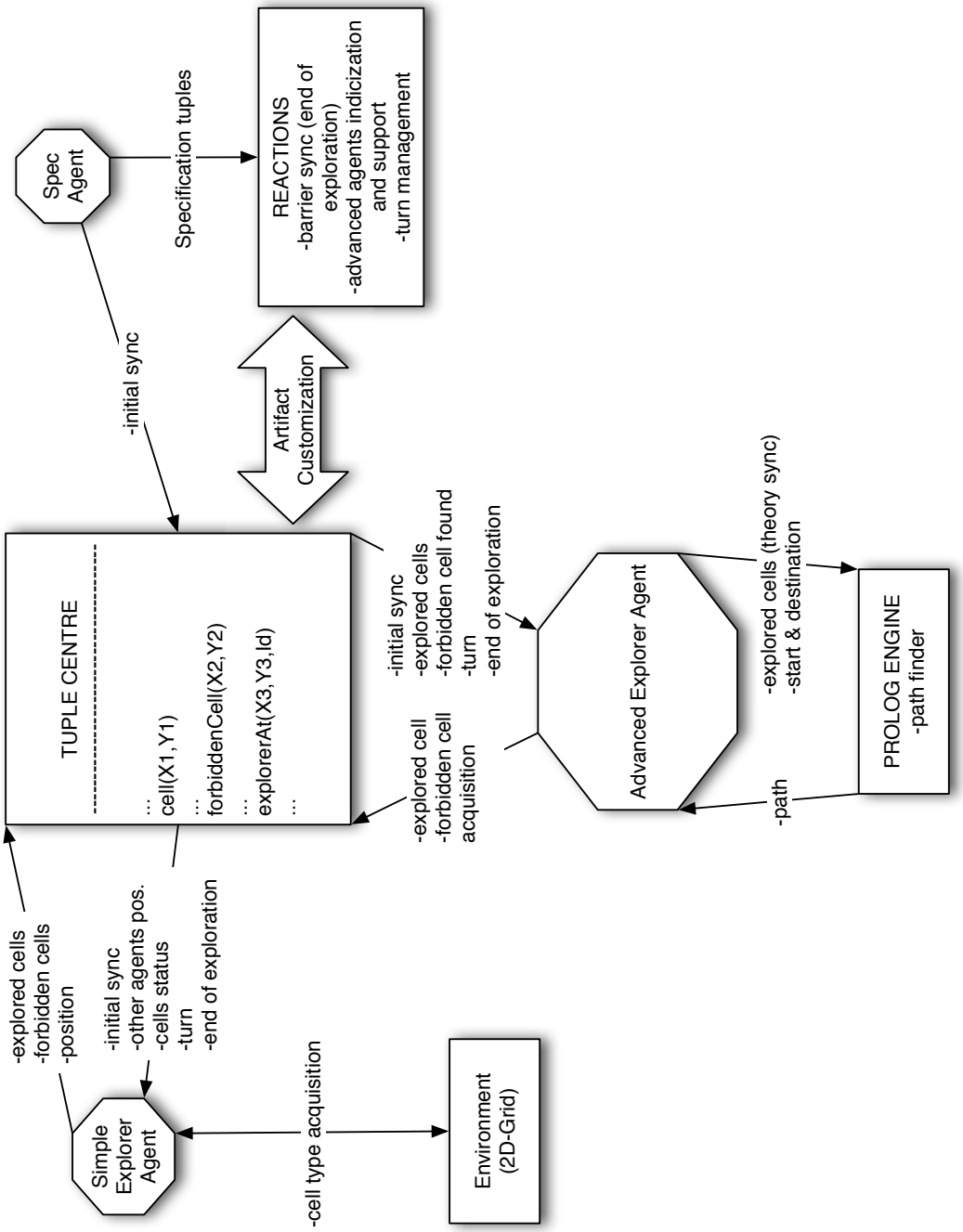

Figura 7: Sistema completo

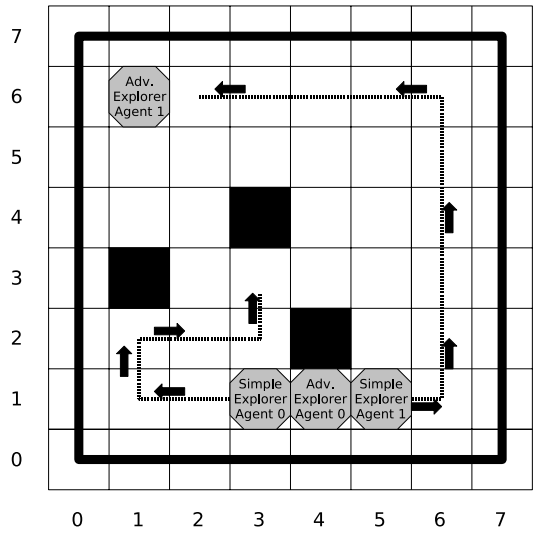

Figura 8: Posizionamento iniziale degli esploratori nella simulazione e percorsi degli agenti semplici

- l'agente avanzato in (1,6) potrà raggiungere la sua destinazione solo quando l'agente semplice inizialmente collocato in (5,1) avrà esplorato la strada che va dalla sua posizione iniziale alla cella (2,6)
- l'agente avanzato in (4,1) potrà immediatamente pianificare un percorso per la cella proibita che gli è stata assegnata

```
TupleCentreId used: testTC
Set up Specification Tuples (ReSpect)...
There are 36 cells to be explored.
SimpleExplorerAgent simpleExplorerAgent_0 placed at 3,1
SimpleExplorerAgent simpleExplorerAgent_1 placed at 5,1
AdvancedExplorerAgent advancedExplorerAgent_1 placed at 1,6
AdvancedExplorerAgent advancedExplorerAgent_0 placed at 4,1
Exploration is started - all explorers spawn
SimpleExplorerAgent simpleExplorerAgent_0 moved to 2,1
SimpleExplorerAgent simpleExplorerAgent_1 moved to 6,1
SimpleExplorerAgent simpleExplorerAgent_0 moved to 1,1
SimpleExplorerAgent simpleExplorerAgent_1 moved to 6,2
SimpleExplorerAgent simpleExplorerAgent_0 moved to 1,2
SimpleExplorerAgent simpleExplorerAgent_1 moved to 6,3
Forbidden Cell found at 1,3
SimpleExplorerAgent simpleExplorerAgent_0 moved to 2,2
SimpleExplorerAgent simpleExplorerAgent_1 moved to 6,4
Forbidden Cell 1,3 has been check-out by advancedExplorerAgent_1
```

```
SimpleExplorerAgent simpleExplorerAgent_0 moved to 3,2
SimpleExplorerAgent simpleExplorerAgent_1 moved to 6,5

AdvancedExplorerAgent advancedExplorerAgent_1 at 1,6 has
a destination (1,3) but not a plan

Forbidden Cell found at 4,2
SimpleExplorerAgent simpleExplorerAgent_0 moved to 3,3
SimpleExplorerAgent simpleExplorerAgent_1 moved to 6,6

AdvancedExplorerAgent advancedExplorerAgent_1 at 1,6 has
a destination (1,3) but not a plan

Forbidden Cell 4,2 has been check-out by advancedExplorerAgent_0

AdvancedExplorerAgent advancedExplorerAgent_0 at 4,1 has
a plan: [cell(4,1),cell(4,2)]
...
...
SimpleExplorerAgent simpleExplorerAgent_0 moved to 6,2
...
...
AdvancedExplorerAgent advancedExplorerAgent_1 at 1,6 has
a plan: [cell(1,6),cell(2,6),cell(3,6),cell(4,6),cell(5,6),
cell(6,6),cell(6,5),cell(6,4),cell(6,3),cell(5,3),cell(4,3),
cell(3,3),cell(3,2),cell(2,2),cell(1,2),cell(1,3)]
...
...
AdvancedExplorerAgent advancedExplorerAgent_1 exited - total explored cells = 36
AdvancedExplorerAgent advancedExplorerAgent_0 exited - total explored cells = 36
SimpleExplorerAgent simpleExplorerAgent_0 moved to 2,6
SimpleExplorerAgent simpleExplorerAgent_0 exited - total explored cells = 36
SimpleExplorerAgent simpleExplorerAgent_1 moved to 5,5
SimpleExplorerAgent simpleExplorerAgent_1 exited - total explored cells = 36
```

6 Conclusioni e sviluppi futuri

Gli artefatti di coordinazione e il linguaggio logico utilizzati forniscono un valido supporto per lo sviluppo di software dotato di caratteristiche “agent oriented” e consentono di creare comportamenti osservabili “intelligenti”.

Gli agenti sviluppati sono di due tipologie ma entrambe mettono in luce, anche se con sfumature diverse, caratteristiche proprie del concetto di agente-MAS:

- il *controllo* è *incapsulato* e volto al compimento di *task individuali* (esplorazione di celle agibili per l’esploratore semplice, di quelle proibite per l’esploratore avanzato) , inquadrabili in un task di più alto livello e attribuito all’intera *società* (esplorazione dell’intero ambiente);

- l'esistenza di una *società* e di *goal sociali* richiede *coordinazione* per raggiungerli;
- il comportamento è di tipo *proattivo*, e non unicamente legato alla reazione a stimoli esterni: in modo particolare, gli esploratori semplici hanno l'obiettivo di esplorare quante più celle possibili e agiscono sulla base di questo obiettivo;
- c'è *interazione* con l'ambiente nel quale gli esploratori sono *situati* ed emerge, evidente, l'*impossibilità di disaccorpare* gli agenti sviluppati dall'ambiente nel quale vivono e operano;
- proprietà di *reattività* (reazioni alla presenza di configurazioni ambientali, di informazioni provenienti da altri agenti), di capacità di *imparare* (gli esploratori avanzati hanno la capacità di acquisire informazioni su nuove celle accessibili -nuove "leggi del mondo"- e da queste informazioni dedurne l'esistenza di percorsi non presenti in precedenza), di *pianificare* (l'agente avanzato pianifica il proprio percorso con lo scopo di raggiungere dalla posizione attuale quella di destinazione) sono circoscritte all'ambiente nel quale gli agenti sono *situati*.

Nel progetto si è cercato di fare uso del numero più elevato possibile di strumenti messi a disposizione del corso, sacrificando, in certe situazioni, la generalità e completezza delle soluzioni individuate per lasciare spazio a importanti concetti.

La ricerca del percorso tra due celle eseguita per mezzo della teoria prolog proposta soffre, in presenza di un numero elevato di celle (griglie di dimensione superiore a 10x10), di problemi di esplosione della complessità di risoluzione, in quanto da ogni cella partono tre diversi percorsi possibili: è quindi opportuno, per il corretto funzionamento del sistema, mantenere la dimensione della griglia al di sotto delle 10x10 celle.

Sono esplorabili strade che aggiungano la possibilità di introdurre meccanismi di pattern recognition agli agenti per individuare cluster di celle inagibili caratterizzati da particolare forma, algoritmi più efficienti di ricerca del percorso e comportamenti maggiormente reattivi degli agenti avanzati. Essi potrebbero, ad esempio, ricalcolare il piano migliore ad ogni istante di simulazione.

Riferimenti bibliografici

[1] aliCE Team. Homepage TuCSoN. http://tucson.sourceforge.net/.

[2] aliCE Team. Homepage tuProlog. http://tuprolog.sourceforge.net/.

[3] A. Omicini. Homepage corso di Sistemi Intelligenti Distribuiti L-S, 2005-2006. http://www-lia.deis.unibo.it/corsi/2005-2006/SID-LS-CE/home.shtml.

[4] Michael J. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons Ltd., 2002.