Relazione Progetto PDDL

Vendramini Simone 866229

Maggioni Giacomo 869265

Il Problema

Un labirinto di NXN caselle contiene un mago e un invurgus, un mostro invisibile che si nutre di barbe di mago. Alcune celle sono dei pozzi senza fondo da cui l'invurgus non può passare, mentre il mago può sorpassare volando. Altre contengono delle colate di lava che bloccano il passaggio sia al mago che all'invurgus. Per volare il mago deve raccogliere un particolare fungo che cresce in alcune celle del labirinto. Ogni fungo gli permetterà di volare due volte. Ci sono due uscite nella parte nord del labirinto e il mago parte da una posizione a sud. Il mago, per fuggire e preservare la sua barba, utilizza un incantesimo che gli permette di scegliere ad ogni passo la prossima mossa migliore.

Considerando queste condizioni:

1. L'invurgus ad ogni passo si muoverà scegliendo la posizione adiacente migliore per diminure la distanza dal mago |x(mago)-x(invurgus)| +|y(mago)-y(invurgus)|, o la successiva in senso anti orario. L'incantesimo conosce la posizione dei funghi, delle

uscite, e del invurgus anche se invisibile. L’incantesimo viene lanciato una sola volta.

2. L’invurgus ad ogni passo si muoverà la traiettoria permissibile con meno passi per raggiungere il mago. L’incantesimo conosce la posizione dei funghi, delle uscite, e dell’Invurgus anche se invisibile. Il mago per lanciare l’incantesimo perde un turno. L’incantesimo può dare al mago altri suggerimenti come “vai a nord e usa di nuovo l’incantesimo in tre passi”
3. L’invurgus ad ogni passo si muoverà la traiettoria permissibile con meno passi per raggiungere il mago. L’incantesimo crede che l’invurgus si muoverà come nell’esercizio 1 conosce la posizione dei funghi, delle uscite, ma riconosce la posizione dell’ invurgus solo se dista meno 3 passi, e lo ignora in altri momenti. Il mago per lanciare l’incantesimo perde un turno. L’incantesimo può dare al mago altri suggerimenti come “vai a nord e usa di nuovo l’incantesimo in tre passi”

Per soddisfare la richiesta serve prima conoscere il tema su cui è incentrato questo progetto, ovvero il PDDL.

Il PDDL

Il Planning Domain Definition Language (PDDL) è un linguaggio formale per definire i domini di pianificazione e le istanze di problemi. Viene utilizzato per descrivere i problemi e le loro soluzioni in modo che i planner possano utilizzarli per generare piani.

Per utilizzare il linguaggio PDDL, è necessario definire il dominio del problema e la sua istanza, specificando gli operatori del problema e le condizioni iniziali e finali.

Il problema sarà quindi rappresentato da una serie di concetti

I predicati

La dichiarazione dei predicati è necessaria per rappresentare i vari componenti del mondo, ovvero la posizione dei vari ostacoli quali le colate di lava ed i buchi, le posizioni dei funghi, delle uscite, del mago e del mostro.

Un altro aspetto importante è la dichiarazione dei fatti che coordinano il mondo, ossia lo spostamento tra le varie caselle, il numero di cariche che possiede il mago ed un

meccanismo semplice ma efficace per la gestione dei turni mago-invurgus.

Infine per gestire meglio l’arrivo ad un goal abbiamo aggiunto un predicato che indica se il mago è uscito dal labirinto, che sarà successivamente la condizione da verificare.

Le azioni

Le azioni nel PDDL definiscono le operazioni che possono essere eseguite sulle istanze del problema. Ogni azione specifica delle precondizioni, ovvero ciò che deve essere vero affinché l'azione possa essere eseguita, e il suo effetto, ovvero il cambiamento di stato che l'azione provoca. Le azioni sono quindi utilizzate dai planner per generare una sequenza di azioni che risolve il problema.

In questo problema specifico sono necessarie una serie di azioni per gestire ogni possibile mossa:

- Azioni di movimento del mago:

  Queste azioni di movimento comprendono tutto quel set che permette di far muovere il mago nelle 4 direzioni (up-down-left-right) sia normalmente che sui buchi spendendo una carica.

- Azioni di movimento del mostro:

  Queste azioni si basano sulla ricerca del percorso migliore per raggiungere il mago, dovranno considerare una metrica per avvicinarsi e decidere come comportarsi nel caso in cui sia inagibile la casella che permette di accorciare la distanza.

- Azioni di interazione:

  Sono azioni più semplici che permettono al mago di interagire con il mondo, come prendere le uscite o mangiare i funghi.

Gli oggetti

Gli oggetti in PDDL sono elementi che rappresentano le istanze specifiche del dominio e del problema. Gli oggetti sono dichiarati all'inizio del file PDDL del dominio e vengono utilizzati nelle precondizioni e negli effetti delle azioni per specificare quali istanze specifiche degli oggetti sono interessate da quella particolare azione.

In questo caso specifico sono necessari solamente gli oggetti che rappresentano le coordinate della tabella e il numero di cariche possibili da assumere.

L’inizializzazione del problema

L’inizializzazione nel PDDL (Planning Domain Definition Language) serve a definire lo stato iniziale del problema. In pratica, è una lista di fatti o predicati che rappresentano le condizioni iniziali del mondo. Questi fatti possono includere informazioni sulle posizioni dei vari ostacoli, sui funghi disponibili, sulla posizione del mago e del mostro, sulle cariche del mago e altro ancora. Queste informazioni vengono utilizzate dai planner per generare una sequenza di azioni che risolve il problema.

Il goal

Il goal è l’obbiettivo del problema, ossia la serie di condizioni da rendere vere per le quali viene realizzato il plan. In questo caso il goal è che il mago si trovi all’uscita e fugga dal labirinto.

La nostra implementazione

Il problema propostoci consiste quindi in tre punti principali:

- L’implementazione PDDL di una prima versione dell’incantesimo, che abbiamo effettuato secondo due punti di vista, quali la considerazione della staticità del mostro e, al contrario, una versione dinamica in cui è considerato il progressivo movimento del mostro sulla base della distanza L1 (Distanza di Manhattan).
- La spiegazione di una possibile soluzione considerando che il movimento del mostro deve considerare il percorso migliore che lo colleghi al mago e la possibilità di utilizzare l’incantesimo più volte nel corso di una partita, che fornirà una serie di azioni al posto di una soluzione effettiva.
- La spiegazione di una possibile soluzione considerando il secondo punto e che l’incantesimo sia in grado di conoscere la posizione del mostro soltanto se si trova nell’intorno di tre caselle.

Link Github

https://github.com/MaggioniGiacomo/ProjectPDDL

Realizzazione prima richiesta

Come già detto in precedenza per il primo punto abbiamo deciso di approcciarlo suddividendolo in due obbiettivi rilevanti:

- Un primo obbiettivo statico, più semplice ma funzionante, per concentrarci nell'individuazione di un percorso dal punto di partenza ad un punto di arrivo senza troppe complicazioni fornite dal mostro.
- Un secondo obbiettivo dinamico, in cui verrà aggiunto il movimento del mostro

Versione statica

La versione statica del problema è stata affrontata anch'essa a step, partendo da problemi più semplici come il trovare una soluzione dal mago all'uscita senza ostacoli e mano a mano aggiungendo complicazioni.

Inizialmente ci siamo interrogati sul come implementare la matrice, avevamo pensato ad una soluzione numerica, ma facendo qualche ricerca abbiamo notato che come soluzione si utilizza una rappresentazione che incapsula le coordinate in stati.

Inoltre è sorto un problema sulla gestione delle cariche fornite dai funghi, che inizialmente abbiamo pensato di gestire tramite una funzione che andava a cambiare dinamicamente il valore delle cariche ad ogni azione in cui erano interessate, ma che non funzionava in quanto il solver che utilizzavamo non era in grado di gestire i :fluents, che sono necessari per effettuare comparazioni numeriche.

Per risolvere questi problemi abbiamo quindi adoperato anche noi l'utilizzo dell'incapsulazione, dato che i solver spesso e volentieri impongono dei vincoli per limitare le possibili combinazioni di stati, come nel nostro caso in cui l'utilizzo dei numeri li avrebbe incrementati drasticamente.

Versione dinamica

La versione dinamica aggiunge la problematica del dover implementare un meccanismo per gestire i turni Invurgus/mago e l'utilizzo della metrica L1 per compiere i movimenti del mostro.

Per il problema della gestione dei turni abbiamo aggiunto due predicati che serviranno ad identificare di chi è il turno.

Per quanto riguarda invece il movimento del mostro abbiamo pensato di decidere il movimento in base alla posizione del mostro rispetto al mago, ma in questo modo è risorto il problema del solver che non ci permetteva l'utilizzo dei numeri.

A questo punto è iniziata una fase di ricerca di nuovi planner, ma tutte fallimentari. Nella tabella a fine relazione ne indichiamo alcuni.

Essendo che tutt'ora ci sfugge come sia possibile implementarlo senza l'utilizzo dei numeri, di seguito illustriamo come avremmo voluto implementare la soluzione.

```
(:action inv-right
                :parameters (?xinv ?yinv ?xn ?xwiz ?ywiz)
                :precondition (and
                        (inv-turn)
                        (at-invirgus ?xinv ?yinv) (at-wizard ?xwiz ?ywiz)
                        (> ?xwiz ?xinv) (inc ?xinv ?xn)
                        (not (LAVA ?xn ?yinv)) (not (HOLE ?xn ?yinv))
                )
                :effect (and
                        (not(inv-turn)) (wiz-turn)
                        (at-invirgus ?xn ?yinv)
                )
        )
```

Oltre a questi quattro casi per le azioni di Up-Left-Right-Down, serve gestire il caso in cui la nuova casella sia inagibile.

La parte di codice seguente mostra la possibile implementazione di gestione dell'impossibilità del movimento.

```
(:action inv-direzione-occupato
                :parameters (?xinv ?yinv ?posoccupato ?posn ?xwiz ?ywiz)
                :precondition (and
                        (inv-turn)
                        (at-invirgus ?xinv ?yinv) (at-wizard ?xwiz ?ywiz)
//verifico che la posizione nuova e quella da occupare siano valide
                        (> ?posoccupato ?xinv) (inc ?xinv ?posn)
                        (inc ?xinv ?posoccupato)
                        (not (LAVA ?posoccupato ?yinv)) (not (HOLE ?posoccupato ?yinv))
                        (not (LAVA ?posn ?yinv)) (not (HOLE ?posn ?yinv))
                )
                :effect (and
//cambio il turno e aggiorno la posizione del mostro a quella in senso orario
                        (not(inv-turn)) (wiz-turn)
                        (at-invirgus ?posoccupato ?yinv)
```

```
                )
        )
```

Inoltre andrebbero gestiti pure i casi in cui le posizioni adiacenti siano occupate andando a ruotare quelle da occupare

Studio seconda richiesta

Analizzando la seconda richiesta ci sono due modifiche da applicare:

- Il mostro dovrà passare da un movimento secondo la metrica L1 a una più ottimale
- L'incantesimo dovrà fornire una serie di step da eseguire e potrà essere utilizzato più volte all'interno della stessa partita

Per attuare il secondo punto l'idea era quella di gestire l'output del solver in un secondo momento: dato che i solver hanno capacità di restituire più piani, se disponibili, pensavamo di mostrare all'utente solo gli step considerati "safe", ossia fino a quando la posizione del mostro risulti deterministica.

Come esempio basta pensare a due canali con la stessa distanza dal mago. In questo caso non si è in grado di dire con certezza quale dei due percorsi il mostro prenderà, ma, essendo che il planner dovrebbe restituire due soluzioni diverse, basterà effettuare un confronto su esse per individuare il punto in cui i plan si differenziano che determina lo step in cui sarà consigliato riutilizzare l'incantesimo.

Per quanto riguarda invece il primo punto volevamo sfruttare il multiagent, in modo da differenziare il mago e il mostro, ognuno con i propri obbiettivi, proprietà e azioni.

Studio terza richiesta

La complicazione del terzo punto sta nella modifica dell'incantesimo che cambia dall'essere onnisciente al non sapere dove si torvi il mostro a meno che è situato entro tre caselle dal mago.

Per fare ciò avevamo pensato di dividere il compito dell'incantesimo in due parti:

- Una prima parte in cui l'incantesimo si occupa del verificare se il mostro si trova entro 3 caselle dal mago

- Una seconda parte in cui in base alla presenza del mostro decide come comportarsi

Per il meccanismo con il quale il mago cerca un piano per trovare l'uscita non serve effettuare modifiche se non per sdoppiare i due casi, in cui la posizione del mostro è conosciuta oppure no.

L'aggiunta principale sta nel trovare quindi un metodo per identificare la presenza del mostro.

La nostra idea era quella di far sapere sempre la posizione del mostro all'incantesimo, ma di decidere di considerarla solo quando effettivamente si trova entro le tre caselle dal mago al momento del lancio dell'incantesimo.

In questo modo è possibile verificare, sapendo la posizione del mago, se nelle varie caselle adiacenti con distanza massima tre è presente l'invurgus e successivamente scegliere opportunamente se considerare o meno il mostro.

Rimane aperto il problema legato alla politica secondo cui l'incantesimo possa decidere di essere riutilizzato per verificare se l'invurgus è vicino o meno.

Fonti utilizzate

Creazione tabella	https://verificationglasses.wordpress.com/2020/11/02/pddl/	Utile
Utilizzo Numeri	https://github.com/yarox/pddl-examples	Sorto problema fluents
Spiegazione problema fluents	https://stackoverflow.com/questions/61402519/pddl-using-numeric-fluents-using-solver-planning-domains	Utile
Problem Solving con ChatGPT	https://chat.openai.com	Utile

Metric-FF	https://github.com/Vidminas/metric-ff-crossplatform	Inutile - Impossibile usare comando make su windows per conflitti con variabili d'ambiente e files .sh - Linux restituisce errori su multipli assegnamenti nei vari file.c

pddl4j	http://pddl4j.imag.fr/getting_started.html#creating-the-executable-jar	Inutile - non supporta :functions - restituisce errore al momento della compilazione
Installazione LPG	http://lpg.unibs.it/lpg/lpgtd-win.tar.gz	Inutile - Durante l'esecuzione si impianta e non parte neanche
Planutils	https://github.com/AI-Planning/planutils	Inutile - Alla creazione del server ci sono dei problemi con la connessione
PDDL Planners	https://www.ai4europe.eu/research/ai-catalog/pddl-planners	Inutile - Problemi con docker
POPF	https://github.com/Vidminas/metric-ff-crossplatform	Inutile - Per essere buildato serve CBC ma il link per scaricarlo da 404