Intelligenza Artificiale e Laboratorio:
CLIPS
Progetto: Rescue2017

Andrea Forgione: 809289
Arthur Capozzi: 802586
Leonardo Zanchi: 800935

Indice

Capitolo 1

Progetto: Rescue2017

Il progetto Rescue2017 si occupa dello sviluppo di un agente intelligente che possa contribuire al salvataggio di persone disperse sotto le macerie di un edificio crollato. In particolare l'agente dovrà fornire informazioni utili ad agenti umani presenti nello stesso ambiente, andando ad individuare la posizione delle persone disperse sotto le macerie con l'ausilio delle percezioni a sua disposizione, ed esplorando l'ambiente in modo efficiente sfruttando le conoscenze acquisite sullo stesso. È inoltre preferibile che esso non intralci le operazioni di soccorso ad opera degli agenti umani, evitando di limitarne gli spostamenti.

È utile elencare le caratteristiche dell'ambiente, così che siano note le problematiche da considerare durante la progettazione dell'agente.

- **Parzialmente osservabile:** l'agente inizialmente ha conoscenza del solo perimetro dell'edificio, della sua posizione e se è carico di detriti. Sarà poi tenuto ad ampliare il suo modello del mondo in base alle percezioni che riceve dai suoi sensori visivi, auditivi, o in seguito ad una delle azioni volte alla raccolta di informazioni.

- **Multi-agente:** oltre all'agente robotico nello stesso ambiente agiscono anche dei soccorritori umani. È difficile caratterizzare in modo netto la natura della convivenza di più agenti. Si può dire che sia *parzialmente cooperativo* dal momento che l'agente robotico riceve delle penalità nell'intralciare il cammino dei soccorritori, facendoli fermare ad aspettare. È tuttavia anche *parzialmente competitivo*, in quanto le celle non possono essere occupate da più di un agente alla volta, e sono quindi una risorsa incondivisibile per la quale gli agenti devono competere per soddisfare i rispettivi obiettivi.

- **Deterministico:** lo stato successivo in cui si troverà l'ambiente è completamente definito dallo stato attuale e dall'azione intrapresa dall'agente robotico.

- **Sequenziale:** l'esperienza dell'agente non è divisa in episodi separati, ogni azione può avere grosse ripercussioni sullo sviluppo a lungo termine delle azioni che egli sceglierà di compiere. Ad esempio, l'aver esplorato l'edificio mediante delle azioni precedenti può modificare molto la sequenza delle azioni che l'agente sceglierà di eseguire successivamente per il raggiungimento di un obiettivo.
- **Statico:** l'ambiente non evolve tra un'azione e l'altra dell'agente robotico.
- **Discreto:** per quanto nell'ambiente siano codificati sia un contatore di passi che un contatore del tempo, è solo il primo a scandire lo sviluppo dell'ambiente, incrementandosi dopo l'esecuzione di un'azione da parte dell'agente. Il contatore del tempo è utilizzato al fine di calcolare il tempo a disposizione dell'agente e le penalità evolutive accumulate; infatti le azioni hanno durate differenti e per ogni unità di tempo trascorsa vengono assegnate delle penalità in base allo stato dell'ambiente. Oltre alla discretizzazione del tempo, anche le posizioni spaziali sono codificate in modo discreto, rappresentando l'ambiente con una griglia rettangolare.
- **Conosciuto:** questa caratteristica non si riferisce all'ambiente stesso, ma alla conoscenza dell'agente sulle implicazioni evolutive sull'ambiente delle proprie azioni. In seguito ad un'azione l'agente sa quali saranno i possibili esiti e come questi andranno a modificare lo stato dell'ambiente.

Capitolo 2

Architettura dell'agente robotico

A fronte delle caratteristiche del problema, abbiamo deciso di realizzare un agente di tipo goal-based la cui struttura segue il modello Belief-Desire-Intention (BDI). Questo modello vede il susseguirsi di tre fasi:

- **Belief:** ad ogni nuovo passo l'agente riceve delle percezioni, che possono essere visive, acustiche o derivanti dall'azione scelta al passo precedente, come ad esempio nel caso dell'azione drill che produce una percezione che informa l'agente sull'eventuale presenza di una persona sotto le macerie; o nel caso delle azioni di load ed unload che informano l'agente della riuscita dell'esito dell'azione, così che esso possa aggiornare lo stato di conoscenza sul fatto che sia carico o meno; o ancora, nel caso di una azione forward che ha portato alla collisione, viene generata una percezione di bump. In base a queste percezioni l'agente aggiorna il suo modello dell'ambiente, il così detto belief state.

- **Desire:** una volta aggiornato il modello interno, l'agente valuta i possibili obiettivi che potrebbe perseguire. Abbiamo deciso di definire vari tipi di obiettivi che modellano azioni ad alto livello che l'agente può eseguire al fine di svolgere i suoi compiti. Questi sono: il desiderio di esplorare celle sconosciute, il desiderio di depositare dei detriti, di ricevere percezioni acustiche così da determinare la presenza di una persona sotto le macerie o di controllare mediante una sonda, ed infine il desiderio di dichiarare di aver svolto il suo lavoro e non avere più nulla da fare, eventualmente a causa del poco tempo rimasto.

- **Intention:** definiti i possibili obiettivi, l'agente deve scegliere quale di questi perseguire.

Va notato che non tutte le azioni possibili sono codificate nei desideri. Questa scelta è stata fatta poiché in alcuni stati alcune azioni vengono effettuate dall'agente di riflesso, senza bisogno di deliberazione, o sono codificate implicitamente nel piano che viene calcolato per raggiungere un obiettivo. Queste sono le azioni per la comunicazione di informazioni riguardanti l'ambiente al centro operativo che coordina le operazioni di soccorso, o l'azione di carico dei detriti che avviene quando il percorso calcolato implica il passaggio in una cella occupata dalle macerie.

2.1 Struttura del programma CLIPS

Passiamo ora a discutere le caratteristiche dell'implementazione dell'agente in CLIPS. La struttura fornita come base su cui sviluppare il progetto prevede vari moduli.

- **ENV:** contiene i fatti e le regole che gestiscono l'evoluzione dell'ambiente, e calcolano le penalità accumulate dall'agente. Comunica con l'agente tramite il modulo MAIN.
- **MAIN:** fa da tramite tra il modulo ENV ed il modulo AGENT. Contiene le definizioni dei modelli dei fatti che codificano lo stato delle celle, le percezioni, le azioni scelte dall'agente, e si occupa di gestire la terminazione del programma in caso di disastro o allo scadere del tempo.
- **AGENT:** è il modulo che contiene tutta la logica della strategia. Nel suo interno sono codificati i fatti per la rappresentazione dell'ambiente, per il suo aggiornamento, e per la gestione del ciclo BDI. È diviso in vari sotto-moduli a cui AGENT passa ciclicamente il focus.
  - **PERCEPTION:** è il modulo che si occupa di tradurre le percezioni ottenute dal modulo MAIN ed aggiornare la base di conoscenza dell'agente. Non ha subito modifiche rispetto a quello consegnato dai docenti.
  - **REASONING:** è il modulo principale dell'agente. È a sua volta suddiviso in altri moduli: **REFLEX**, **SEARCH**, **PICK-GOAL**. Il primo sotto-modulo aggiorna i desideri, fornendo quindi l'implementazione alla fase Desire. Se disponibili nello stato attuale, asserisce un'azione di riflesso che rende non necessario il passaggio ai due moduli successivi. Il modulo SEARCH si occupa di calcolare i piani per il raggiungimento degli obiettivi. Infine, il modulo PICK-GOAL si occupa di scegliere quale tra i piani ottenuti mediante la ricerca eseguire, andando ad implementare di fatto la fase Intention.

- **ACTION:** una volta selezionato il piano da eseguire, il modulo ACTION traduce le azioni, che sono fornite ad un livello d'astrazione superiore, nelle azioni base eseguibili dall'agente: forward, turnleft, turnright, load_debris, unload_debris, drill, done.

Ora che abbiamo un'idea generale della struttura del programma CLIPS possiamo andare ad analizzare i vari moduli nel dettaglio. Come abbiamo detto, il modulo principale è AGENT con i suoi sotto-moduli, ed è su questo che si concentrerà la relazione, per poi passare al modulo ACTION, con una breve descrizione delle regole che portano all'effettiva asserzione del fatto relativo all'azione scelta. I moduli ENV e MAIN sono rimasti immutati rispetto a quelli consegnati dai docenti e quindi non richiedono ulteriori dissertazioni.

2.2 MODULO AGENT

Nel modulo AGENT sono definiti i template per i fatti non ordinati che rappresentano il suo modello dell'ambiente. Il template **K-cell** codifica lo stato delle celle, le cui informazioni son limitate a ciò che è conosciuto dall'agente in quel momento. Il template **K-agent** rappresenta lo stato dell'agente, ovvero la sua posizione, il fatto che sia o meno carico di detriti, la sua direzione attuale, oltre ad informazioni rispetto al tempo della simulazione, espressa in step ed in unità di tempo. Infine nel modello dell'ambiente vengono codificati gli eventuali soccorritori, il cui stato è rappresentato con fatti **K-person** che ne contengono la posizione.

In questo modulo sono definiti i template utilizzati per comunicare tra i vari sotto-moduli.

```
(deftemplate goal-candidate (slot id) (slot goal-type (allowed-
   values explore debris unload_debris drill done))
      (slot pos-r) (slot pos-c) (slot param1) (slot param2)
      (slot activated (allowed-values yes no)))
(deftemplate current-goal-candidate (slot id))
```

Il template **goal-candidate** rappresenta gli obiettivi candidati tra cui l'agente sceglie quello da perseguire, indicato da **current-goal-candidate**.

Sono definiti alcuni template che verranno utilizzati nel modulo SEARCH in cui viene calcolato un cammino verso il completamento dell'obiettivo.

```
(deftemplate path (slot id) (slot created-at-step) (slot goal-
   type) (slot pos-r) (slot pos-c) (slot cost) (slot biased-
   cost))
```

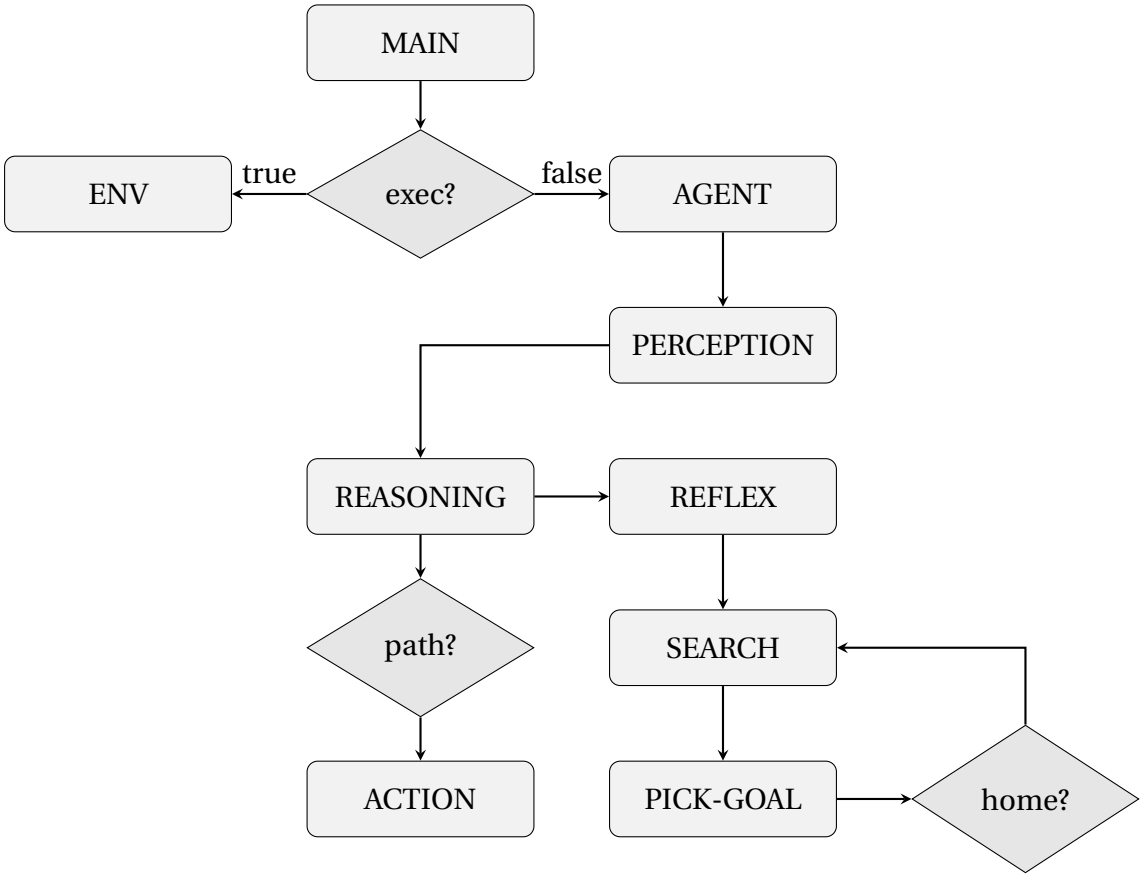
```
(deftemplate path-step (slot path-id) (slot step) (slot
   direction (allowed-values north south east west drill
   unload_debris done)))
(deftemplate path-picked (slot path-id))
(deftemplate home-plan (slot id) (slot cost) (slot calculated-
   at-time))
```

I fatti di tipo **path** rappresentano un cammino prodotto dal modulo SEARCH, indicando il tipo e la posizione dell'obiettivo a cui conducono, il costo del cammino, ed un identificatore che li lega ai corrispettivi fatti **path-step** che definiscono la sequenza da azioni ad alto livello da eseguire. **home-plan** può essere considerato come un percorso particolare utilizzato per condurre l'agente in un nodo gate, qualora non sia rimasto tempo sufficiente per soddisfare altri obiettivi. Se ne parlerà nel dettaglio in seguito.

Abbiamo visto i template dei fatti che compongono la struttura comunicativa tra i vari moduli. Oltre a questi, in AGENT sono presenti altri template di fatti di utilità, come ad esempio **offset-around** che si occupa di fornire fatti per la traduzione delle posizioni relative all'agente a quelle del sistema di riferimento principale della griglia. **turn-direction** offre un modello per fatti che permettono di esprimere l'azione che l'agente deve compiere per andare in una determinata direzione. Vi sono inoltre template per l'assegnazione dei costi alle varie azioni, sia per le azioni ad alto livello, in **move-cost**, che per le azioni atomiche che l'agente può compiere ad un passo di esecuzione, **action-cost**.

Al fine di comprendere il modo in cui si susseguono i vari moduli, qui di seguito è riportato un diagramma di flusso. Si può notare la mancanza di frecce all'indietro in modo da creare il ciclo di ragionamento e azioni che ci si aspetterebbe. Questo è dovuto al fatto che i moduli si susseguono andando ad aggiungersi in cima ad una pila. Quando il diagramma raggiunge un modulo dal quale non escono frecce, una volta che non sono disponibili più azioni, il modulo in cima alla pila viene rimosso, andando a seguire a ritroso il cammino che aveva seguito.

Un'iterazione tipica, partendo dal modulo **MAIN**, non essendo ancora presenti fatti exec, che asseriscono la scelta di un'azione, passa il controllo al modulo **AGENT**. Questo a sua volta lo cede al modulo **PERCEPTION** per ricevere le percezioni create nal modulo ENV all'esecuzione dell'azione al passo precedente. Si passa dunque al modulo **REASONING** che esegue il focus su REFLEX in cui vengono aggiornati gli obiettivi candidati, ed eventualmente viene scelta un'azione di riflesso. Al fine di non appesantire troppo la rappresentazione grafica non è presente un nodo di scelta, tuttavia in seguito all'esecuzione delle regole del modulo REFLEX, qualora sia stata selezionata un'azione, il modulo REASONING potrebbe



saltare il passaggio al modulo SEARCH ed andare direttamente al modulo ACTION. Nel modulo **SEARCH** viene eseguita la pianificazione dei percorsi che conducono agli obiettivi candidati, dopodiché si passa al modulo **PICK-GOAL** che compie la deliberazione su quale di questi percorsi utilizzare effettivamente andando ad asserire un fatto **path-picked**. Una volta tornato il controllo al modulo REASONING, viene eseguito il focus al modulo **ACTION** che asserisce l'azione **exec**. A questo punto, in seguito ad una serie di pop sulla pila dei moduli, il controllo passa al modulo **ENV** che modifica lo stato del mondo in accordo con l'azione scelta.

Un'ultima cosa prima di descrivere dettagliatamente gli altri moduli: l'agente è stato progettato in modo da avere a disposizione varie opzioni per modificarne comodamente il comportamento, riportiamo qui di seguito un esempio di una configurazione dei fatti che definiscono tali parametri. Vedremo come questi influenzano il comportamento dell'agente nei moduli successivi.

```
(deffacts AGENT-OPTIONS
      (AGENT-OPTION_inform-clear)
```

```
      (AGENT-OPTION_home-plan)
      (AGENT-OPTION_single-goal)
      (AGENT-OPTION_max-plans 10)

      (bias-cost (goal-type explore) (cost 0))
      (bias-cost (goal-type unload_debris) (cost -20))
      (bias-cost (goal-type debris) (cost -8))
      (bias-cost (goal-type drill) (cost -65))
      (bias-cost (goal-type done) (cost 0))
)
```

2.3 Modulo PERCEPTION

Il modulo PERCEPTION non ha subito modifiche rispetto a quello consegnato dai docenti, di conseguenza evitiamo di aggiungere altro a quanto già detto in precedenza.

2.4 Modulo REASONING

Il modulo REASONING si occupa di gestire il loop del ragionamento dell'automa passando il focus al modulo REFLEX e, qualora non siano state asserite azioni di riflesso, al modulo SEARCH, ed infine al modulo PICK-GOAL. È in questi moduli che è contenuta la logica del ragionamento dell'automa.

2.5 Modulo REFLEX

Il modulo REFLEX si occupa di fare delle inferenze sullo stato delle celle debris in base alle percezioni sonore ricevute. A partire da una cella in cui è stato percepito un suono, se c'è solo una cella adiacente (secondo la definizione di 4-adiacenza) con dei detriti che possano contenere delle persone ferite, allora si può affermare con certezza la presenza di un ferito sotto le macerie di tale cella.

```
(defrule cell-is-discovered-north (declare (salience 200))
      (K-agent (pos-r ?r) (pos-c ?c))
      (K-cell (pos-r ?r) (pos-c ?c) (sound yes))
```

```
      (or (K-cell (pos-r (- ?r 1)) (pos-c ?c) (contains ~
         debris)) (K-cell (pos-r (- ?r 1)) (pos-c ?c) (
         contains debris) (injured no)))
      ...
      ?cell <- (K-cell (pos-r (+ ?r 1)) (pos-c ?c) (contains
         debris) (injured unknown))
=>
      (modify ?cell (injured yes))
)
```

Quando invece non si è ricevuta una percezione sonora in una cella adiacente a dei detriti, si può asserire con certezza che non ci siano feriti al di sotto delle macerie.

```
(defrule cell-is-checked (declare (salience 200))
      (K-agent (pos-r ?r) (pos-c ?c))
      (direction ?direction)
      (direction-offset (direction ?direction) (pos-r ?offset-
         r) (pos-c ?offset-c))
      (K-cell (pos-r ?r) (pos-c ?c) (sound no))
      ?cell <- (K-cell (pos-r (+ ?r ?offset-r)) (pos-c (+ ?c
          ?offset-c)) (contains debris) (injured unknown))
=>
      (modify ?cell (injured no))
)
```

In seguito a queste asserzioni, il modulo REFLEX si occupa dell'aggiornamento degli obiettivi candidati, fatti che seguono il template **goal-candidate** definito nel modulo AGENT. Vi sono regole per la creazione, la disattivazione e la riattivazione dei vari candidati che seguono logiche particolari a seconda del tipo di obiettivo.

- **Explore:** si tratta del desiderio dell'agente di esplorare celle sconosciute, così da ampliare la sua conoscenza dell'ambiente. Gli obiettivi di questo tipo vengono attivati per ogni cella unknown e vengono disattivati nel momento in cui tale desiderio viene soddisfatto, ovvero questa cella viene percepita visivamente dall'agente.

- **Debris:** è un desiderio che spinge l'agente a visitare le celle adiacenti a detriti sotto i quali non sa ancora con certezza se si trovino o meno dei feriti. Questo obiettivo si attiva per le celle adiacenti ad una cella detrito e si disattiva quando esplorare tale cella non darebbe più informazioni utili riguardo alla presenza o meno di un ferito, ovvero nei casi in cui nelle celle adiacenti non ci sia più incertezza sulla presenza di feriti, la cella sia inoccupabile, o sia adiacente ad una cella contenente un ferito e quindi riceveremmo una percezione sonora a causa di esso senza poter discernere la direzionalità.

- **Drill:** esprime il desiderio di utilizzare la sonda per venire a conoscenza della presenza o meno di feriti sotto le macerie. Viene attivato per ogni cella contenente detriti di cui non si conosca la presenza o meno di feriti, viene disattivato nel momento in cui tale incertezza viene meno.

- **Unload debris:** l'agente durante le sue operazioni può doversi liberare la strada caricando dei detriti. L'agente individua posti strategici in cui è possibile scaricare le macerie senza poi andare ad intralciarne in seguito i movimenti. Nel momento in cui l'agente si trova carico di detriti, i desideri di tipo unload si riattivano per poi disattivarsi una volta eseguita l'azione di scarico.

- **Done:** sono i desideri che portano l'agente a tornare ad una cella di tipo gate nel momento in cui non siano più presenti altri desideri soddisfacibili a causa dell'irraggiungibilità degli stessi o del tempo residuo a disposizione dell'agente. Viene asserito un desiderio di tipo done per ogni cella gate.

Figura 2.1

```
(defrule goal-candidate-unload-pattern-3 (declare (salience
   200))
      (direction ?direction)
```

```
      (offset-around (direction ?direction) (position forward-
        left) (offset-r ?offset-A-r) (offset-c ?offset-A-c))
      (offset-around (direction ?direction) (position forward-
        right) (offset-r ?offset-C-r) (offset-c ?offset-C-c))
      (offset-around (direction ?direction) (position left) (
        offset-r ?offset-D-r) (offset-c ?offset-D-c))
      (offset-around (direction ?direction) (position right) (
        offset-r ?offset-F-r) (offset-c ?offset-F-c))
      (offset-around (direction ?direction) (position behind-
        left) (offset-r ?offset-G-r) (offset-c ?offset-G-c))
      (offset-around (direction ?direction) (position behind-
        right) (offset-r ?offset-I-r) (offset-c ?offset-I-c))

      (K-cell (pos-r ?r) (pos-c ?c) (contains empty|gate))
      (K-cell (pos-r =(+ ?r ?offset-A-r)) (pos-c =(+ ?c ?
        offset-A-c)) (contains wall))
      (K-cell (pos-r =(+ ?r ?offset-C-r)) (pos-c =(+ ?c ?
        offset-C-c)) (contains empty|gate))
      (K-cell (pos-r =(+ ?r ?offset-D-r)) (pos-c =(+ ?c ?
        offset-D-c)) (contains wall))

      (K-cell (pos-r =(+ ?r ?offset-F-r)) (pos-c =(+ ?c ?
        offset-F-c)) (contains empty|gate))
      (K-cell (pos-r =(+ ?r ?offset-G-r)) (pos-c =(+ ?c ?
        offset-G-c)) (contains wall))
      (K-cell (pos-r =(+ ?r ?offset-I-r)) (pos-c =(+ ?c ?
        offset-I-c)) (contains empty|gate))
=>
      (assert (unload-candidate-spot (pos-r ?r) (pos-c ?c)))
)
```

Sono state definite varie regole per l'individuazione di celle adatte allo scarico che richiedono una descrizione più approfondita. Le celle di tipo outdoor adiacenti a celle di tipo gate o empty vengono selezionate con una semplice regola. Per celle interne all'edificio si utilizzano delle regole più complesse per determinare quali celle soddisfino dei determinati

pattern di celle 3x3 per cui depositare i detriti nella cella centrale non blocca il passaggio alle altre celle adiacenti. Nel listato precedente è riportata una regola per il matching dell'ultimo pattern in figura 2.1. Tale regola aggiunge come possibile candidato per un desiderio di tipo unload.

Infine il modulo REFLEX mediante delle regole asserisce le azioni di riflesso. Sono state scelte come azioni di riflesso quelle che inviano informazioni al centro operativo. Vengono scelte in ordine di priorità decrescente informazioni relative alla presenza di un ferito (inform di tipo **discover**), l'assenza di un ferito in una cella contenente macerie (inform **checked**), e l'informazione del fatto che una cella sia libera da macerie (inform **clear**). Quest'ultimo tipo di inform è reso opzionale mediante l'uso del fatto **AGENT-OPTION_inform-clear**. La scelta di rendere opzionale tale strategia risiede nel fatto che la mancata comunicazione della presenza di celle vuote implica penalità di tipo evolutivo di un solo punto per unità di tempo trascorsa, senza penalità al termine delle operazioni. A seconda della natura dell'ambiente le penalità per la mancata inform su celle vuote può risultare minore delle penalità che si potrebbero evitare sfruttando il tempo compiendo altre azioni. Inoltre il dichiarare una cella clear ne preclude l'utilizzo come cella per uno scarico dal momento che tale azione porterebbe ad una penalità di 200000 punti. È dunque interessante andare ad analizzare come al variare di tale opzione cambiano le performance dell'agente.

2.6 MODULO SEARCH

Il modulo SEARCH implementa l'algoritmo di ricerca **A***. Gli stati che compongono lo spazio di ricerca è definito dalla posizione dell'agente, dalla sua direzione e dal fatto che sia carico o meno. Le azioni disponibili ad ogni passo della ricerca sono il movimento in una delle celle 4-adiacenti. Tali azioni si differenziano rispetto alle azioni reali a disposizione dell'agente che sono forward, turnright, e turnleft; ciò va ad incrementare il fattore di ramificazione della ricerca riducendone però la profondità. Abbiamo fatto questa scelta principalmente per ragioni di debugging: un percorso definito in termini di spostamenti assoluti ne permetteva una più facile lettura. Il costo di tale scelta è la necessità di una successiva traduzione delle azioni, attuata nel modulo ACTION.

La scelta dell'euristica è ricaduta sulla misura della distanza di Manhattan definita come:

$$Manhattan(x_1, y_1, x_2, y_2) = |x_1 - x_2| + |y_1 - y_2|$$

dove (x_1, y_1) e (x_2, y_2) sono le coordinate delle due celle, ovvero il numero di riga e colonna. Questa euristica sembrava la soluzione più naturale per stimare i costi del movimento in un ambiente modellato con una griglia in cui i movimenti avvengono tra celle 4-adiacenti. Si tratta sicuramente di un'euristica ammissibile e consistente in quanto stima la distanza minima che l'agente deve percorrere, considerando un costo unitario per ogni spostamento.

L'agente seleziona i vari candidati per la ricerca in ordine crescente di distanza, stimata con la distanza di Manhattan. Vengono presi in considerazione tutti i tipi di obiettivi attualmente attivi ad eccezione di quelli di tipo done che sono gestiti in modo particolare. Verrà eseguita la ricerca per un numero di obiettivi pari al valore del fatto **AGENT-OPTION_max-plans**; in caso nessuna delle ricerche abbia esito positivo l'agente continua a selezionare un nuovo obiettivo e ad eseguirne la pianificazione fino ad ottenere una soluzione. La ricerca vera e propria inizia una volta selezionato l'obiettivo candidato. Viene selezionato il nodo iniziale, quello in cui si trova l'agente. La logica del planning è suddivisa in tre moduli.

- **SEACH-EXPAND:** seleziona le azioni disponibili e ne esegue una, andando a determinare il nodo a cui essa conduce, passando il controllo al modulo successivo. Una volta terminate le azioni disponibili dal nodo corrente, passa a selezionare il nodo con costo minore.

- **SEARCH-CHECK:** controlla se il nuovo nodo coincide con il nodo obiettivo. In caso positivo si passa alla creazione della soluzione nel modulo CREATE-SOLUTION, altrimenti si passa al modulo successivo.

- **SEARCH-NEW:** verifica se il nuovo nodo è già presente nella frontiera o se è già stato esplorato. In caso faccia parte della frontiera, se ha un costo minore, va a sostituire il nodo preesistente, altrimenti viene scartato. Se si trattasse di un nodo nuovo, viene aggiunto alla frontiera.

Come accennato poc'anzi, in caso si sia ottenuta una soluzione, viene passato il controllo al modulo **CREATE-SOLUTION** che si occupa di asserire i fatti path e path-step i cui template sono stati definiti nel modulo AGENT.

Per motivi di efficienza, viene mantenuta nel fatto **current-lowest-cost** il costo del cammino completo con costo minore trovato fino a quel momento, così da poter effettuare pruning sui nodi che superano tale costo durante le ricerche di piani per gli obiettivi selezionati successivamente. A tale fine, la regola **node-worse-than-global** nel modulo SEARCH-EXPAND va a chiudere i nodi il cui costo supera il minore attuale. Questa tecnica, in congiunzione

col fatto che gli obiettivi vengono selezionati in base ad una stima del loro costo, porta a ricerche più brevi, rendendo possibile la pianificazione di più obiettivi ad ogni passo in tempi relativamente ridotti.

Vedremo ora delle varianti del modulo SEARCH in grado da alterare il comportamento dell'agente.

2.6.1 Single goal

L'agente descritto fin ora effettua la pianificazione selezionando tra i candidati gli obiettivi più vicini, fino ad un massimo definito da AGENT-OPTION_max-plans. Ciò può portare alla selezione di un nuovo obiettivo ad ogni passo. Può risultare interessante valutare il comportamento dell'agente quando esso persegue l'obiettivo scelto fino al completamento o al fallimento. È possibile attivare questa strategia aggiungendo alla base di conoscenza il fatto **AGENT-OPTION_single-goal**. Questa opzione va ad attivare delle regole che portano ad una deliberazione per selezionare un nuovo obiettivo candidato solo nel caso in cui quello attuale sia stato soddisfatto o che, in seguito ad una nuova pianificazione, non sia risultato raggiungibile.

2.6.2 Home Plan

In alcuni casi il tempo a disposizione dell'agente non è sufficiente a soddisfare tutti gli obiettivi prefissatisi. L'agente è tenuto ad effettuare un'azione finale, denominata done, per non incorrere in penalità. In particolare, se non viene fatta la done, si riceve una penalità di 10 milioni di punti. Se la done non viene effettuata su un nodo di tipo gate la penalità è di 100000 punti. Per questo può risultare fruttuoso assicurarsi di eseguire sempre la done su un nodo gate. A tale scopo è possibile attivare una strategia che tenga sempre a disposizione un piano per il raggiungimento di una cella gate aggiungendo il fatto **AGENT-OPTION_home-plan** alla base di conoscenza.

Come abbiamo detto in precedenza, gli obiettivi di tipo done non vengono considerati durante la ricerca principale, essi infatti vengono considerati solo nel momento in cui non ci siano più obiettivi raggiungibili o, nel caso sia attivata l'opzione AGENT-OPTION_home-plan, sia richiesto un ricalcolo del piano. Questo secondo caso avviene nel momento in cui la stima del tempo necessario per arrivare ad una cella gate è inferiore rispetto al tempo di esecuzione del piano attualmente selezionato. Questa stima viene fatta in modo pigro, per motivi di efficienza: ogni volta che viene richiesto un ricalcolo vengono valutati tutti i cammini per

soddisfare gli obiettivi done, viene selezionato quello di costo minore e viene asserito il fatto home-plan che contiene tale costo e il tempo in cui il cammino è stato calcolato; la stima quindi sarà il costo più il tempo trascorso dall'ultimo ricalcolo, aggiungendo una piccola soglia di sicurezza per tener conto degli eventuali cambi di direzione necessari per fare retro front e tornare sui propri passi. Le ricerche per gli obiettivi di tipo done vengono fatte in modo sicuro, andando a considerare le sole celle conosciute così da non incappare in brutte sorprese, ad eccezione dell'eventuale operatore umano, la cui posizione non è nota all'agente, se non nel momento in cui si trovi in una delle celle 8-adiacenti, ed in modo tale che il costo del cammino sia quello effettivo.

2.6.3 Biased Costs

I costi dei percorsi includono il costo dell'azione finale corrispondente al tipo di obiettivo. Questo valore ci è necessario, ad esempio, per calcolare i costi per tornare ad una cella gate nell'Home Plan. Tuttavia, utilizzando questo valore per la scelta del piano, azioni molto utili come la drill, che richiede 60 unità di tempo, verrebbero selezionate con precedenza bassissima rispetto, ad esempio, ad un'azione di esplorazione. Al fine di esprimere l'importanza di determinate azioni rispetto ad altre sono stati introdotti dei bias sui costi. Questi valori vengono sommati al costo reale di un cammino, ed è tale somma ad essere presa in considerazione durante la pianificazione e, come vedremo nel prossimo paragrafo, durante la selezione del percorso da seguire.

Al fine di chiarire l'utilità dei bias, facciamo l'esempio di una configurazione utilizzata durante la valutazione sperimentale delle performance dell'agente. È stato selezionato un bias per gli obiettivi drill di -60, così da metterli alla pari di una explore, contando sul fatto che nel momento in cui si va a valutare un obiettivo di tipo drill, generalmente ci si trova in un'area già esplorata. Un bias di -10 sugli obiettivi debris, fa in modo che non venga eseguita una drill qualora sia ancora possibile evincere la presenza di feriti tramite le percezioni acustiche attraverso un percorso di costo ragionevolmente basso.

2.7 Modulo PICK-GOAL

Il modulo PICK-GOAL si occupa di selezionare il percorso con costo inferiore tra quelli per cui è stata trovata una soluzione.

```
(defrule pick-best-path (declare (salience 100))
      (not (path-picked))
```

```
      ?path <- (path (id ?best-id) (biased-cost ?best-cost) (
         goal-type ?goal-type&:(neq ?goal-type done)))
      (not (path (id ?other-id&:(neq ?other-id ?best-id)) (
         biased-cost ?other-cost&:(< ?other-cost ?best-cost))))
=>
      (assert (path-picked (path-id ?best-id)))
)
```

Oltre a questa regola, è in questo modulo che avviene la valutazione sulla necessita del ricalcolo del piano per arrivare ad una cella gate, o della necessità di valutare anche gli obiettivi di tipo done in caso non siano presenti altri obiettivi realizzabili.

2.8 MODULO ACTION

Il modulo ACTION, come abbiamo già accennato in precedenza, si occupa di tradurre le azioni di alto livello generate durante la ricerca in azioni eseguibili dall'agente.

```
(deftemplate turn-direction (slot direction-from) (slot
   direction-to) (slot action (allowed-values forward turnright
    turnleft)))
```

Mediante l'utilizzo di fatti che seguono il template turn-directin, è possibile inferire l'azione che dovrà compiere l'agente per muoversi in una determinata cella adiacente conoscendo la sua direzione attuale e la direzione di destinazione, espresse relativamente al sistema di riferimento del mondo, ovvero utilizzando i valori *north*, *south*, *west*, *east*.

Come meccanismo di sicurezza, l'esecuzione dell'azione avviene in due fasi: viene prima asserito un fatto di tipo **test-exec** per dichiarare l'intenzione di compiere tale azione, dopodiché si effettua un ultimo controllo per assicurarsi che il costo dell'azione lasci il tempo di effettuare un'azione di tipo done "di emergenza" al passo successivo.

Capitolo 3

Risultati sperimentali

Al fine di compiere un'analisi sperimentale dell'efficacia delle strategie implementate, abbiamo eseguito dei test con varie configurazioni dell'agente. Attraverso l'analisi dei risultati potremmo dedurre come differenti parametri vadano ad influire sulle scelte compiute dall'agente.

Per avere un confronto diretto con strategie implementate da altri colleghi, i test sono svolti sulle mappe create da loro, e verranno riportati i loro risultati a confronto con quelli prodotti dai nostri agenti.

Le mappe scelte si differenziano in base a:

- **Grandezza:** sono state analizzate mappe 10x10, quelle fornite come test dai docenti, 15x15, e 20x20.
- **Tempo a disposizione:** per ogni mappa vi è una versione a tempo illimitato, in cui l'agente è tenuto a compiere tutti gli obiettivi raggiungibili, ed una versione in cui il tempo a disposizione non è sufficiente a soddisfarli. In questi casi è possibile valutare l'impatto della strategia di Home Plan.
- **Rapporto detriti e feriti:** a parità di grandezza e tempo a disposizione le mappe sono state differenziate così da avere un rapporto tra detriti contenenti feriti e detriti senza persone intrappolate. Questo, come vedremo, darà modo di fare delle considerazioni sulla strategia AGENT-OPTION_inform-clear.

3.1 Configurazioni degli agenti

Abbiamo scelto di analizzare otto differenti configurazioni, tenendo invariati i bias. I valori scelti sono riportati in tabella 3.1. Si è scelto di considerare come azione neutra l'explore.

	Bias
Explore	0
Unload	-25
Debris	-10
Drill	-60

Tabella 3.1

Si prediligono le azioni unload dando un bias netto di -5 così che tale azione sia preferita ad una semplice explore quando si transita vicino ad una zona favorevole allo scarico di detriti. Dati i considerevoli risparmi di tempo di una deduzione della presenza di feriti per mezzo delle percezioni acustiche, agli obiettivi debris è stato assegnato un bias di -10, sufficiente ad aggirare una cella. Infine, si è azzerato il costo dell'azione di drill, così da non ignorare tale azione se non per dedurre la posizione di feriti in altro modo.

	Alpha	AlphaNoInf	AlphaNoHome	Beta	BetaNoInf
Inform Clear	Sì	No	Sì	Sì	No
Home Plan	Sì	Sì	No	Sì	Sì
Single Goal	No	No	No	Sì	Sì
Max Plans	100	100	100	100	100

	Charlie	CharlieNoHome	RENGA
Inform Clear	Sì	Si	Sì
Home Plan	Sì	No	Sì
Single Goal	No	No	Sì
Max Plans	10	10	1

- **Agente Alpha:** è configurato con un numero di obiettivi (100) che dovrebbe essere sufficiente a selezionare ad ogni passo il piano con il costo minore tra tutti quelli disponibili. Riesegue delle pianificazioni complete dopo ogni azione, e seleziona quella con costo minore. Ovviamente ciò comporta dei tempi di delibera abbastanza sostenuti, specialmente in mappe di grandi dimensioni. E' utile analizzare quanto queste computazioni extra influiscono sul rendimento dell'agente, in modo da trovare un giusto compromesso. Sono presentate inoltre due varianti: **AlphaNoInfor** ed **AlphaNoHome**, che rispettivamente non compiono azioni di inform sulle celle vuote, o non tengono conto di un piano per tornare al gate qualora il tempo a disposizione stesse per scadere.

- **Agente Beta:** come l'Agente Alpha, durante la delibera, pianifica 100 obiettivi candidati. Tuttavia, nel momento in cui sceglie quello da seguire, continuerà a pianificare per

soddisfarlo fin quando l'obiettivo non sarà giunto a compimento o sarà irraggiungibile. Per questo agente proponiamo la variante **BetaNoInfor**.

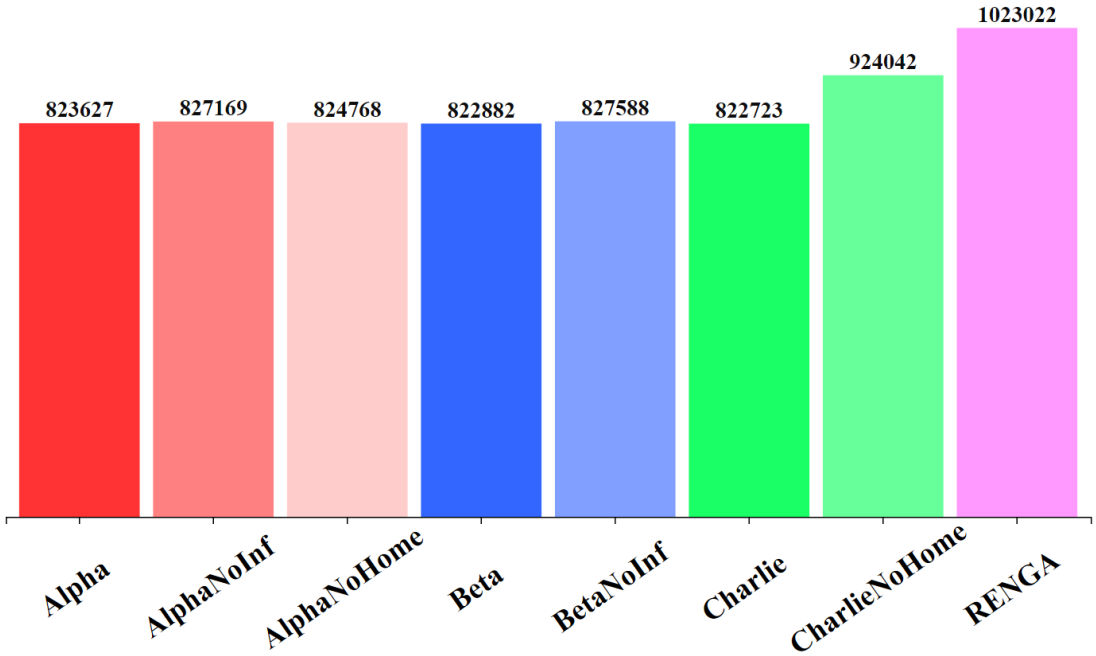
- **Agente Charlie:** al fine di determinare l'impatto del numero massimo di obiettivi considerati sul rendimento dell'agente, in questa implementazione tale parametro è settato a 10. Ciò dovrebbe attivare una ricerca locale più minuziosa. Per questo agente è proposta anche la variante **CharlieNoHome**.
- **Regular Everyday Normal Guy Agent (RENGA):** è l'agente più semplice. Ad ogni passo compie un solo planning (o più, fin quando uno non abbia successo) e, come l'Agente Beta, segue l'obiettivo prefissatosi fin quando gli è possibile.

3.2 First World

Figura 3.1

Si tratta della mappa di prova fornita dai docenti, di dimensioni 10x10, con due soccorritori umani in movimento. Nella mappa sono presenti obiettivi insoddisfacibili, infatti l'agente non può in nessun modo raggiungere le celle delimitate da detriti con feriti, e muri.

Tempo Limitato (300)



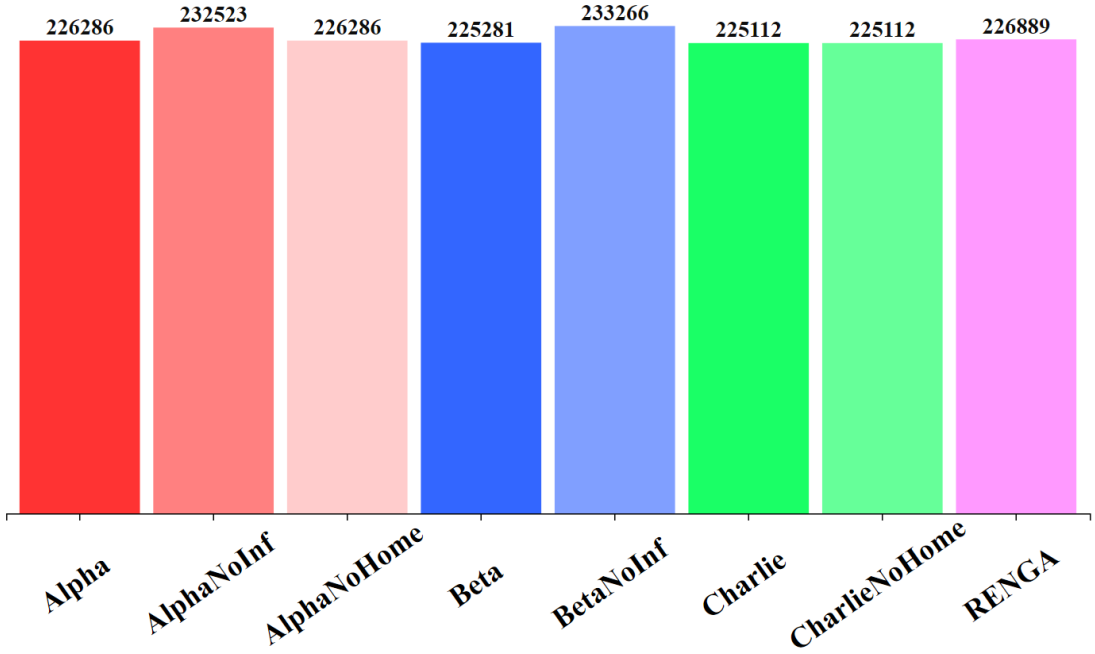
In questa mappa il bias sugli obiettivi di tipo debris, utile per evitare drill dispendiose, non volge a nostro favore, portando l'agente ad una serie di nuove percezioni inconcludenti. Se avesse fatto immediatamente una drill avrebbe risparmiato tempo e gliene sarebbe rimasto a disposizione per esplorare la zona ovest della mappa. Ovviamente queste considerazioni a posteriori non sono un indizio dell'inefficienza della strategia che porta all'esplorazione mediante percezioni auditive, che in generale risulta fruttuosa, almeno in base a risultati sperimentali.

I risultati ottenuti sono abbastanza simili, ad eccezione degli agenti CharlieNoHome e RENGA, dove le scelte, specialmente nel caso del secondo agente, risultano poco ottimali, ed in un contesto a tempo limitato ciò si ripercuote pesantemente sulle penalità.

Tempo Illimitato

Nella versione a tempo illimitato, l'agente compie in ogni caso un buon lavoro, ottenendo punteggi abbastanza simili tra loro. La semplicità di questa mappa non permette di individuare i vantaggi delle varie strategie, che si differenziano nei comportamenti poco l'una dall'altra, ed anche azioni decisamente sub-ottimali non vanno ad incidere in modo significativo sui risultati. In modo poco sorprendente, le varianti **NoHome** coincidono con le strategie base,

dal momento che l'obiettivo done viene scelto esclusivamente in caso tutti gli altri obiettivi siano stati soddisfatti o siano diventati insoddisfacibili.



World 15x15 10d 30f 1p

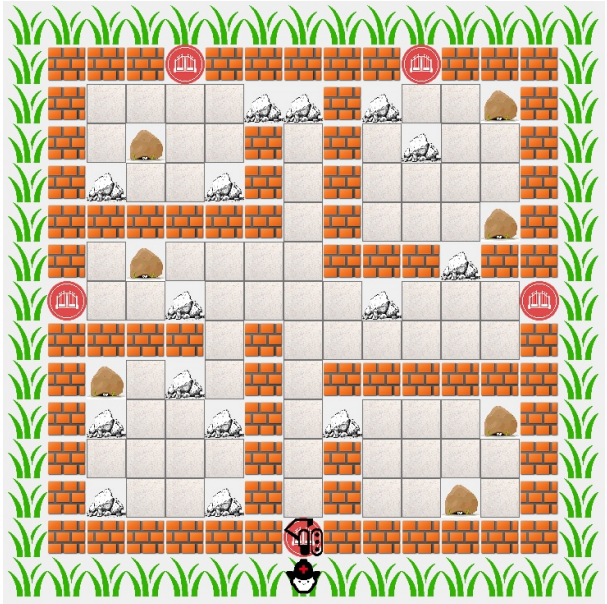
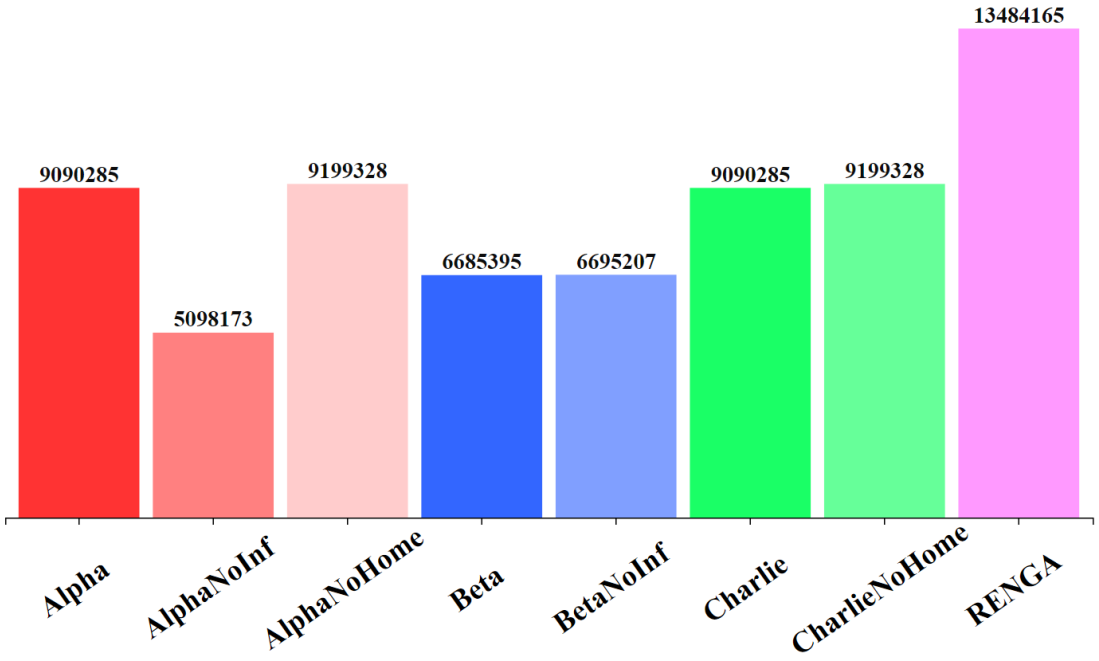


Figura 3.2

Tempo Limitato (300)

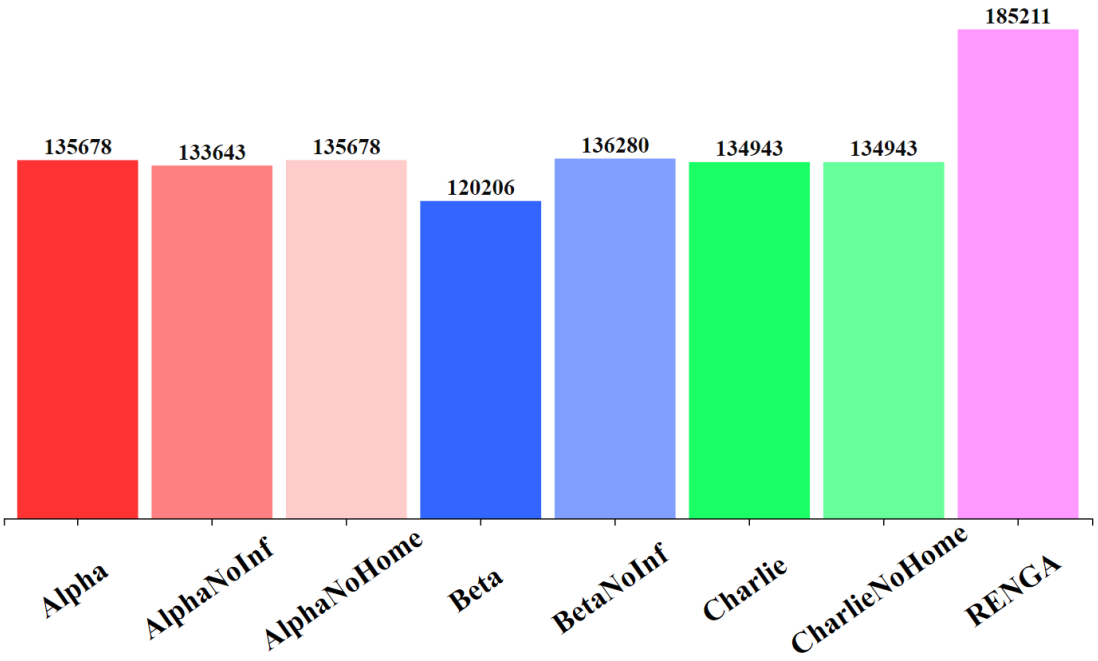
In questo scenario possiamo apprezzare l'utilità della strategia **NoInform**, in particolar modo per l'Agente Alpha, ma anche per l'Agente Beta. Questo risultato ci suggerisce che nel momento in cui il tempo è una risorsa limitata, e c'è un rapporto relativamente alto tra detriti con feriti e senza, ogni secondo è importante, e non risulta ben speso per comunicare la presenza di celle vuote.

Non eseguire l'inform ha varie implicazioni favorevoli: si lasciano più celle a disposizione delle eventuali operazioni di scarico, rendendole molto più rapide; per mappe di media grandezza, con tempi brevi, e con grande concentrazione di feriti, le penalità evolutive per la mancata inform risultano inferiori rispetto alle altre penalità evitate.



Tempo Illimitato

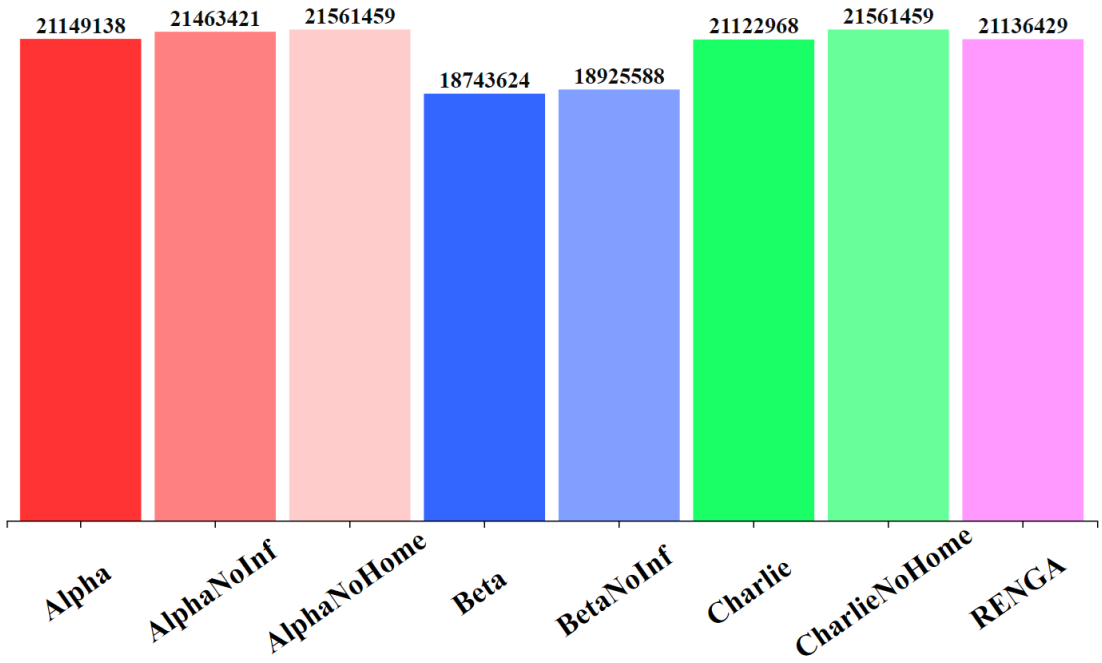
Quando il tempo è illimitato e le operazioni di soccorso si dilungano, le penalità evolutive vanno a colmare il divario che si era creato tra le strategie NoInform e le altre. C'è essenzialmente poco divario tra le varie implementazioni, se non per RENGA che si conferma il peggiore.



3.3 World 15x15 30d 10f 1p

Figura 3.3

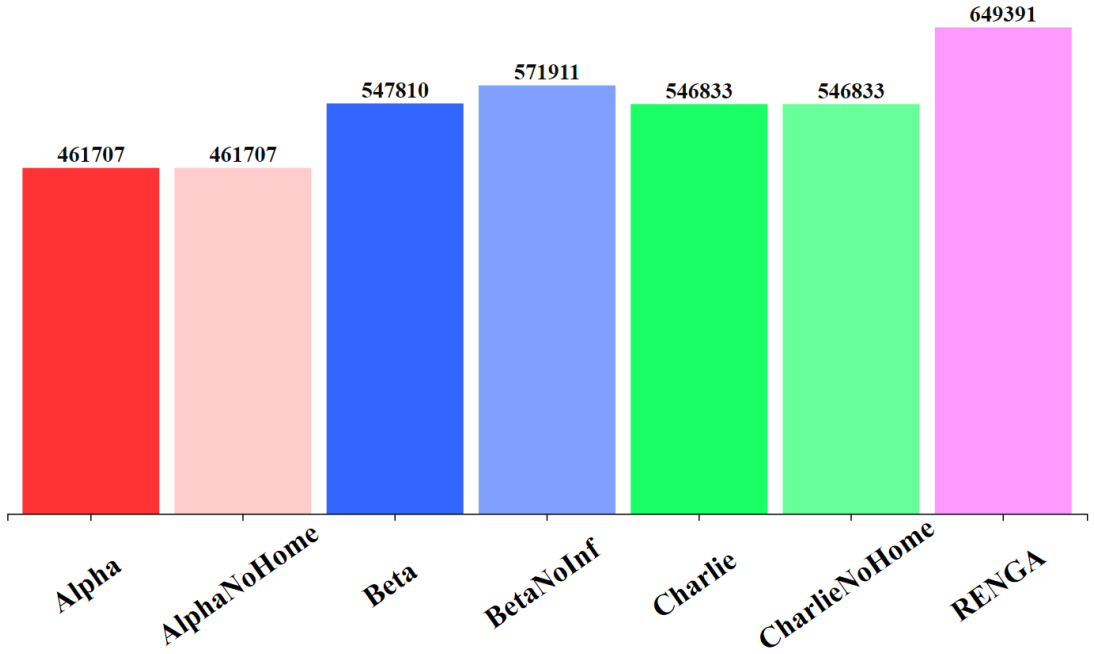
Tempo Limitato (300)



Con una mappa delle stesse dimensioni della precedente, ma con una densità di feriti minore, non si osservano vantaggi delle strategie NoInform. Una minore concentrazione di feriti corrisponde ad un minor "costo" del tempo. Le penalità per il mancato ritrovamento di feriti viene a pesar meno e non si ha più il grande risparmio del caso precedente. Tali strategie rimangono comunque competitive rispetto alle strategie base.

L'**Agente Beta** ottiene il minor numero di penalità. Persistere nel suo intento di soddisfare un obiettivo a tutti i costi lo conduce in una delle stanze dove sono presenti feriti, anziché controllare prima i corridoi che risulterebbero più facilmente visitabili.

Tempo Illimitato



Nell'istogramma manca il dato relativo alle penalità dell'Agente **AlphaNoInf**. In questo caso si è assistito ad un episodio di sabotaggio da parte dei creatori della mappa. Un soccorritore umano si è avvicinato all'agente, senza entrare mai nel suo campo visivo, e lo ha colto di sorpresa nel momento in cui stava effettuando una unload in una cella gate senza via d'uscita.

Episodi simili si sono verificati anche con altri agenti, che sono riusciti però a non rimaner bloccati e, compiendo un'azione a vuoto (nel nostro caso scegliendo turnleft in modo arbitrario) sono riusciti a far transitare il soccorritore fuori dall'edificio.

3.4 WORLD 20X20 10D 30F 1P

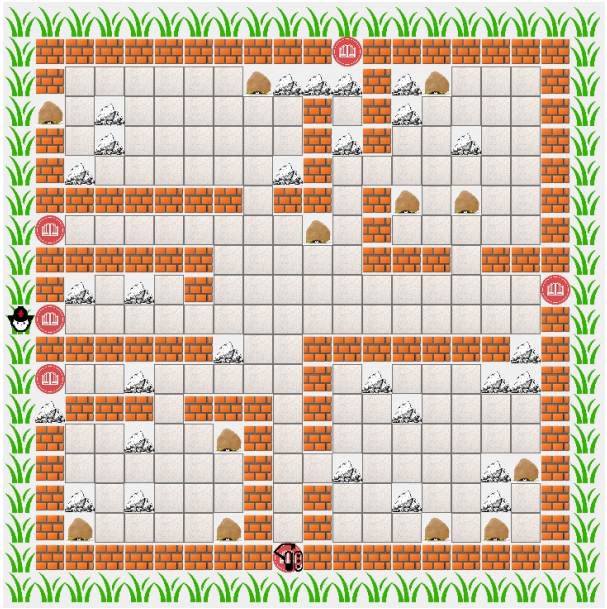
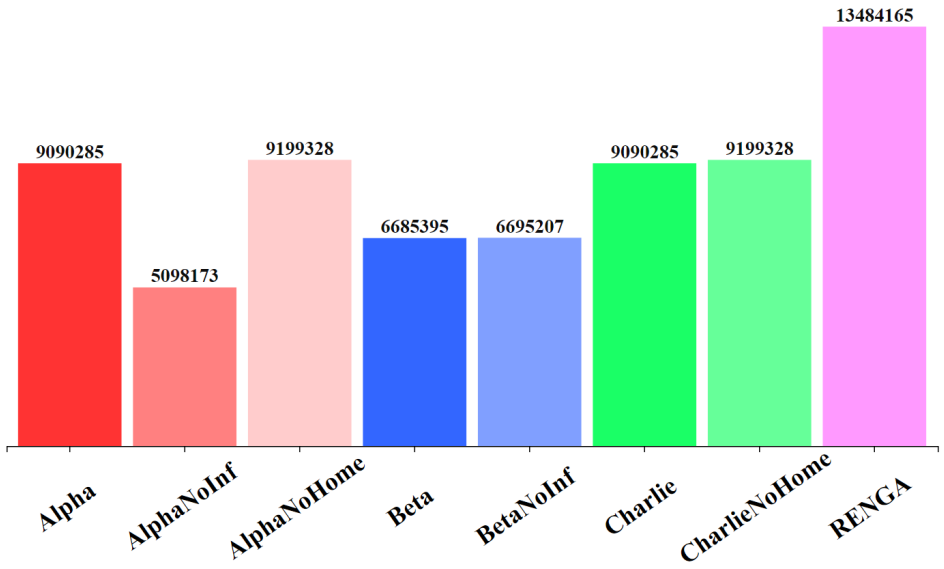


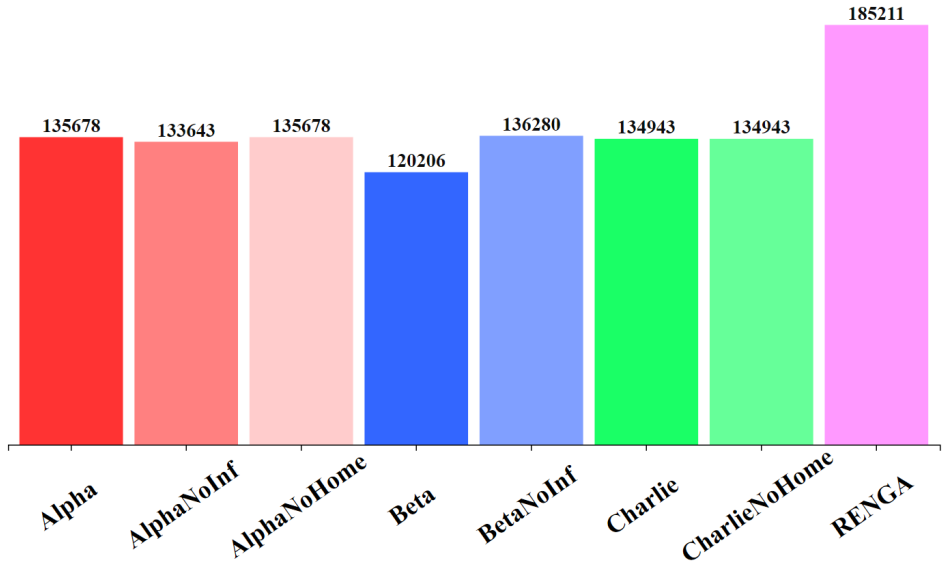
Figura 3.4

Possiamo osservare lo stesso tipo di comportamento delle strategie NoInform anche in queste mappe con un alto rapporto feriti-detriti. Nella versione a tempo limitato i benefici dell'evitare l'inform clear sono evidenti, tuttavia a lungo termine questi vengono colmati dal continuo accumularsi di penalità evolutive.

Tempo Limitato (400)



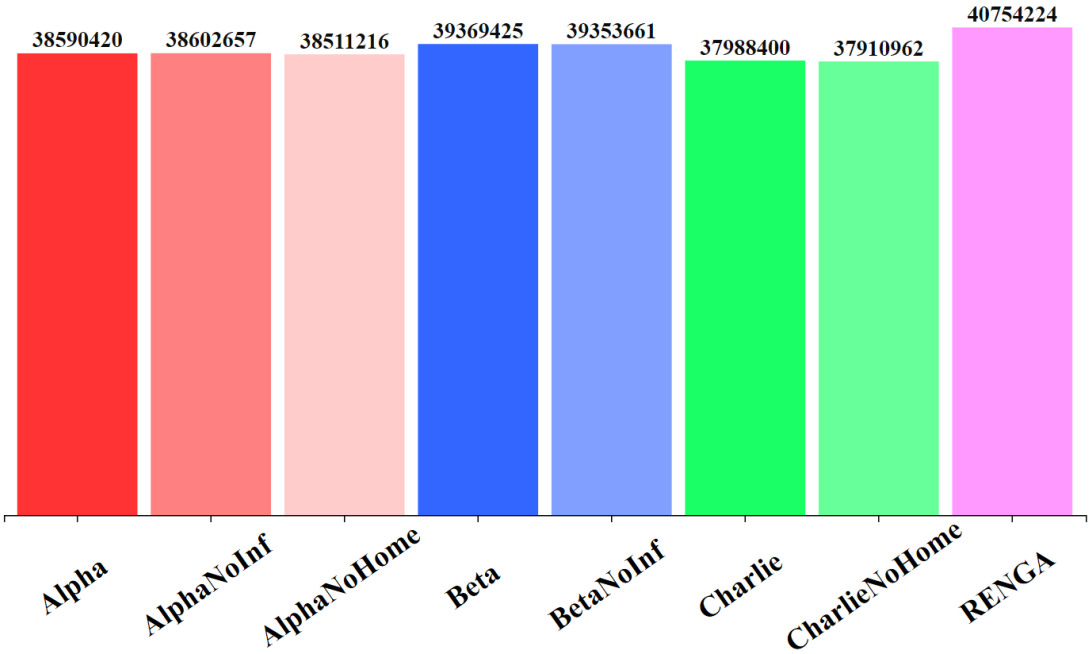
Tempo Illimitato



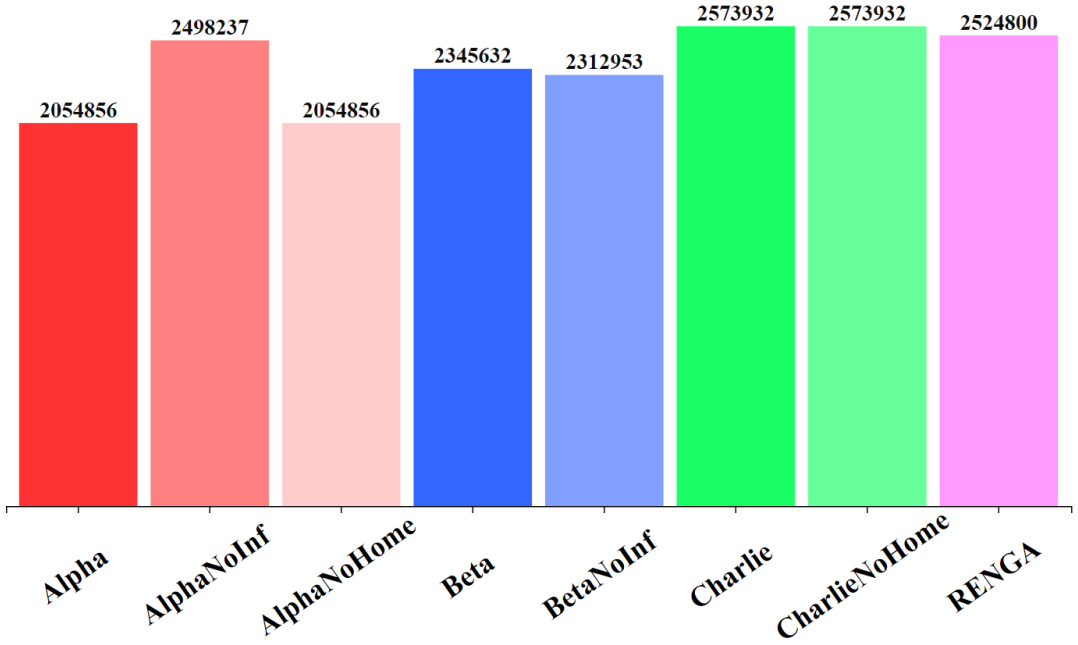
3.5 World 20x20 30d 10f 1p

Figura 3.5

Tempo Limitato (400)



Tempo Illimitato



Capitolo 4

Conclusioni

Le strategie che abbiamo implementato, in base ai riscontri che abbiamo avuto, ci sono sembrate efficaci. Tuttavia vi sono molte altre strategie che avremmo voluto implementare, ma che per questioni di tempo non siamo riusciti a realizzare, strategie da rafforzare, ed altre che ci sono state suggerite dai risultati, che potrebbe essere interessante investigare.

Tra le idee che non hanno visto realizzazione, pensavamo di guidare la scelta sulle celle da visitare assegnando ad ogni cella la somma di valori corrispondenti al tipo della sua cella e delle celle del suo vicinato, pesando gli addendi in base alla distanza dalla cella presa in esame. Insomma, una sorta di convoluzione con un filtro gaussiano, che avrebbe permesso di valutare la bontà di un obiettivo, non solo in base all'obiettivo stesso, ma anche al numero e al valore degli obiettivi nei suoi pressi.

Nell'implementazione della ricerca, il caso in cui l'agente si trova carico e desidera passare oltre una cella contenente detriti, richiederebbe una pianificazione per le operazioni di scarico prima di continuare. Questa complessità extra non è stata implementata, ed è gestita indirettamente dal fatto che all'agente non è permesso passare su celle contenenti detriti in caso sia già carico. Nel caso si verifichi una situazione del genere, anche nel caso Single Goal, l'agente passerebbe alla selezione di un nuovo obiettivo, eventualmente una unload, così da poter soddisfare in seguito l'obiettivo fallito in precedenza. Tuttavia è possibile immaginare casi molto particolari in cui, a causa di soccorritori umani, sarebbe necessaria una robustezza extra durante la pianificazione.

Abbiamo visto un caso in cui non c'è possibilità di evitare di rimanere bloccati da un soccorritore umano. In altri casi, l'agente, trovatosi bloccato da un operatore umano e senza obiettivi realizzabili, attende che esso si sposti compiendo un'azione a vuoto. In caso abbia

obiettivi raggiungibili si dirige verso di essi. Questo durante i test è risultato sufficiente per sbloccare la situazione. Tuttavia qualora non avesse obiettivi realizzabili, neppure un gate da raggiungere, effettuerebbe una done sul posto dopo aver atteso uno step. Questa situazione potrebbe essere risolta con metodi più robusti: l'agente potrebbe, se scarico, caricare un detrito senza feriti e spostarsi al suo posto, o potrebbe andare a scegliere in modo casuale delle azioni disponibili, nella speranza che il soccorritore riesca a proseguire il suo cammino.

A partire dai risultati sperimentali ottenuti, mentre la strategia senza Home Plan non pare aver avuto un impatto significativo, la strategia che evita di compiere l'inform clear realizza risultati promettenti in determinati casi. Il passo successivo potrebbe essere quello di raccogliere dati statistici sull'ambiente in modo da decidere in corso d'opera se effettuare inform o meno. Purtroppo all'agente non è dato sapere quante penalità stia accumulando, e dunque non può valutare quanto è prezioso il tempo.

In seguito all'osservazione dell'esecuzione di agenti senza l'opzione Single Goal, potrebbe essere interessante provare a far variare il numero di obiettivi considerati ed i bias. All'inizio delle operazioni l'agente potrebbe esplorare il più velocemente possibile l'ambiente andando a valutare un grande numero di obiettivi, utilizzando bias che rendano sfavorevoli azioni costose come la drill. Una volta superata una fase esplorativa iniziale, l'agente potrebbe restringere il suo campo d'azione, andando a lavorare localmente zona per zona, ottenendo il conseguimento di tale strategia diminuendo il numero di obiettivi considerati e riducendo i bias delle azioni più lunghe.

In seguito sono riportati i risultati di tutti i test che abbiamo condotto, a confronto con quelli di un altro gruppo.

	Mappe				
Strategia	**First World S**	**15x15 10D 30F 1P S**	**15x15 30D 10F 1P S**	**20x20 10D 30F 1P S**	**20x20 30D 10F 1P S**
Base	535281	12433336	22071073	25002216	40712665
Double-Plan	835291	12528951	21976353	24892170	40623786
Delib \w Scores	923442	13622144	21660740	22365666	40275441
Open Minded	523366	9413910	21561304	16764020	38260275
Single Minded	523366	9413910	21461206	20130870	38277099
No Inform Clear	2354880	12211943	22078885	18198755	40725374
Alpha	823627	9090285	21149138	10630612	38590420
AlphaNoInf	827169	5098173	21463421	5004825	38602657
AlphaNoHome	824768	9199328	21561459	8730241	38511216
Beta	822882	6685395	18743624	17259522	39369425
BetaNoInf	827588	6695207	18925588	16643837	39353661
Charlie	822723	9090285	21122968	10833766	37988400
CharlieNoHome	924042	9199328	21561459	8734917	37910962
RENGA	1023022	13484165	21136429	23633267	40754224

Strategia	**First World L**	**15x15 10D 30F 1P L**	**15x15 30D 10F 1P L**	**20x20 10D 30F 1P L**	**20x20 30D 10F 1P L**
Base	236144	196399	873339	813477	3197944
Double-Plan	236144	196399	873339	813477	3197944
Delib \w Scores	225628	228726	635970	474865	2523690
Open Minded	224409	171371	676735	410878	2178984
Single Minded	224409	171775	666213	456218	2222213
No Inform Clear	257140	182119	882364	578435	3132076
Alpha	226286	135678	461707	317826	2054856
AlphaNoInf	232523	133643	* 21463421	293274	2498237
AlphaNoHome	226286	135678	461707	317826	2054856
Beta	225281	120206	547810	378136	2345632
BetaNoInf	233266	136280	571911	455174	2312953
Charlie	225112	134943	546833	279980	2573932
CharlieNoHome	225112	134943	546833	279980	2573932
RENGA	226889	185211	649391	407055	2524800