BATTAGLIA NAVALE IN CLIPS

IALab A.A. 19/20

Amedeo Racanati 928995

Angelo Pio Sansonetti 928869

Introduzione al problema

Lo scopo del progetto è stato quello di implementare un sistema esperto che giochi ad una versione semplificata di battaglia navale, all'interno di una griglia di dimensione 10x10. L'agente deve cercare di individuare le seguenti navi:

- 1 corazzata da 4 caselle;
- 2 incrociatori da 3 caselle ciascuno;
- 3 cacciatorpedinieri da 2 caselle ciascuno;
- 4 sottomarini da 1 casella ciascuno.

La versione di battaglia navale cui l'agente dovrà giocare è semplificata poiché:

- il contenuto di alcune celle potrebbe essere noto fin dall'inizio;
- in corrispondenza di ciascuna riga e colonna l'agente conosce il numero di celle che contengono navi;
- tra due navi deve esserci almeno una cella libera (cioè con dell'acqua). Noi abbiamo fatto l'assunzione che tra due navi vi deve essere almeno una cella libera anche diagonalmente.

Il sistema esperto potrà eseguire all'interno della griglia le seguenti azioni:

- Fire [x,y]: permette di vedere il contenuto della cella;
- Guess [x,y]: il sistema esperto ipotizza che ci sia una nave in posizione [x, y];
- Unguess [x,y]: il sistema esperto ritratta l'ipotesi sviluppata con la guess (quest'azione non viene utilizzata dal nostro sistema esperto);
- Solve [x,y]: il sistema esperto ritiene di aver risolto il gioco e attiva il calcolo dello score.

Il sistema esperto ha a disposizione massimo 5 fire e massimo 20 guess. L'obiettivo è quindi marcare tutte le celle contenenti porzioni di navi come guessed o eventualmente colpirle con una azione di fire.

Quando il sistema esperto ritiene di aver concluso e richiama il calcolo dello score, questo viene calcolato con la seguente formula:

$$(10*fok+10*gok+15*sink\ )-(25*fko+15*gko+10*safe)$$

Dove:

- fok è il numero di azioni fire che sono andate a segno;
- gok è il numero di celle guessed corrette;
- sink è il numero di navi totalmente affondate;
- fko è il numero di azioni fire andate in acqua;
- gko è il numero di celle guessed errate;
- safe è il numero di celle che contengono una porzione di nave e che sono rimaste inviolate (né guessed né fired).

Il nostro sistema esperto

La prima fase della progettazione del sistema si è basata sull'ideare tutte le possibili regole di inferenza che permettono di ottenere maggiore conoscenza riguardo il contenuto delle celle.

Dopo questa fase si è passati alla definizione dei template che hanno permesso l'asserzione di nuova conoscenza. Qui di seguito sono commentati i template definiti nel sistema.

```
(deftemplate g-cell (slot x)
                    (slot y)
                    (slot probability (default -1))
                    (slot updated (default 0) (allowed-values 0 1))
                    (slot considered (default 0) (allowed-values 0 1))
                    (slot content (default nil) (allowed-
values water left right middle top bot sub nil)))
```

Anche se esiste il template non ordinato *k-cell*, abbiamo avuto l'esigenza di definirne uno similare che ha permesso la gestione di quelle celle per le quali non si conosce

l'esatto contenuto. Oltre alle coordinate x e y abbiamo introdotto una variabile indicante la probabilità per la cella di contenere un pezzo di nave; abbiamo introdotto due variabili booleane, *updated* e *considered*, ove la prima permette di aggiornare le stime di probabilità dopo ogni step, mentre la seconda permette di effettuare una singola azione di fire o guess su ciascuna cella.

Ogni volta che è rilevato un nuovo fatto non ordinato di tipo *k-cell*, vengono aggiornate le informazioni della relativa *g-cell*. Inoltre le celle contenenti acqua vengono rimosse dalla lista dei fatti.

```
(deftemplate g-per-row
    (slot row)
    (slot num)
    (multislot g-cells)
    (multislot gk-cells)
)
(deftemplate g-per-col
    (slot col)
    (slot num)
    (multislot g-cells)
    (multislot gk-cells)
)
```

Questi due template svolgono un ruolo similare a quello di *k-per-col* e *k-per-row*, ma con qualche differenza. Lo slot *num* indica quante parti di navi devono essere ancora trovate per la relativa riga/colonna, ossia indica il numero di celle che contengono una nave, ma per le quali non si conosce la posizione. Quando *num* è valorizzato a 0, significa che per quella riga/colonna sono state individuate tutte le celle contenenti una nave e quindi le rimanenti celle conterranno sicuramente acqua.

Il multislot *g-cells* contiene le celle per le quali non si sa se contengono un pezzo di nave o meno, mentre il multislot *gk-cells* contiene le celle per le quali non si conosce esattamente il contenuto, anche se la probabilità di contenere qualcosa è pari a 1 (questo multislot include per definizione le celle del multislot precedente).

```
(deftemplate g-boat
    (slot size (allowed-values 1 2 3 4))
    (slot alignment (allowed-values hor ver))
    (slot mainColRow)
    (multislot secColRow)
)
```

Questo template serve per asserire le navi trovate, che hanno una determinata grandezza ed uno specifico orientamento. Qualora l'orientamento sia orizzontale (oppure verticale), *mainColRow* indica la riga (colonna) nella quale è collocata la nave, mentre *secColRow* indica le colonne (righe) nelle quali è collocata la nave.

```
(deftemplate comb-boat
    (slot size (allowed-values 1 2 3 4))
    (slot alignment (allowed-values hor ver))
    (slot mainColRow)
    (multislot initialSecColRow)
    (multislot secColRow)
)
```

Questo template permette di asserire tute le possibili combinazioni di navi di una determinata grandezza. Nel sistema vengono generate combinazioni di navi di grandezza 3 o 4. Qualora l'orientamento sia orizzontale (oppure verticale), *mainColRow* indica la riga (colonna) nella quale è collocata la nave, mentre *initialSecColRow* indica le colonne (righe) nelle quali è collocata la nave.

Il multislot *secColRow* è similare ad *initialSecColRow*, ma contiene solo quelle celle per le quali la probabilità di contenere un pezzo di nave non è ancora certa. Ciò ha permesso di effettuare particolari tipi di inferenze: ad esempio, qualora ci sia una combinazione orizzontale sulla riga 3 per una nave di grandezza 4, e per questa combinazione 2 celle non hanno la certezza di contenere una nave; qualora sulla riga 3 è presente il fatto (g-per-row (num 1)), si può dedurre che in quella riga ci sarà soltanto un'altra cella occupata da una nave, e quindi questa combinazione può venire scartata perché inconsistente.

Regole di inferenza

Così come la regola di inferenza citata alla fine della sezione precedente, abbiamo definito altre regole che possono essere classificate in tre categorie principali:

- regole basate sul contenuto di una cella e di quelle adiacenti ad essa;
- regole basate sulle navi trovate e quelle ancora da trovare;
- regole basate sulle possibili combinazioni di navi.

Lo scopo di ciascuna di queste regole è quello di stabilire per le varie celle la probabilità di contenere una nave, e possibilmente anche il loro contenuto.

Il primo gruppo di regole si basa sul contenuto delle singole celle e permette di fare inferenze sulle celle adiacenti. Ad esempio, dato che una nave non può trovarsi vicino ad un'altra nave, qualora si sappia che una cella ha il contenuto "top", si può dedurre che le celle immediatamente sopra così come le celle laterali possiedono con certezza dell'acqua, mentre la cella che si trova in basso possiede un pezzo di nave, anche se il contenuto è sconosciuto.

Una cella "sub" invece avrà intorno a sé delle celle contenenti acqua. In sostanza queste regole hanno lo scopo di inferire la probabilità delle celle adiacenti a quelle già conosciute.

Il secondo gruppo di regole, basate sulle navi trovate o ancora da trovare, servono per effettuare inferenze anche sul contenuto delle celle. É bene citare il fatto ordinato *(found-all-boats ?num)* che viene asserito quando sono state trovate tutte le navi di una data dimensione *?num*.

Se ad esempio la nave da 4 è stata individuata e si sa che 3 celle contigue hanno con certezza un pezzo di nave, si può dedurre che le 3 celle appartengono alla stessa nave che ha necessariamente dimensione 3, e pertanto si può dedurre anche il loro contenuto.

Inoltre se sono state trovate tutte le navi di dimensione 1 e 2, quelle celle contigue tra loro che non possono contenere una nave di dimensione maggiore a 2 vengono escluse, perché contenenti acqua.

Il terzo gruppo di regole si basa sulle possibili disposizioni di navi da 3 e 4 sulla griglia, qualora queste non siano state già tutte trovate. Dopo l'esecuzione di alcune

regole che asseriscono tutte le possibili combinazioni delle navi, via via che si ha maggiore conoscenza della griglia, è possibile rimuovere le combinazioni inconsistenti. Quando sarà rimasta una sola possibile combinazione da 4 (o da 3), si potrà asserire con certezza che in quella posizione ci sarà una nave di relativa grandezza. Infine qualora una cella dovesse essere presente in tutte le possibili combinazioni per una data grandezza, si può dedurre che in quella cella ci sarà sicuramente un pezzo di nave.

Moduli dell'agente

Il modulo dell'agente è composto a sua volta da 3 sotto-moduli:

- AGENT
- AGENT_CELL_BASE_INFERENCE
- AGENT_DECISION

Il modulo AGENT ha lo scopo di guidare il processo di inferenza e di decisione.

Dopo l'asserzione dei principali fatti non ordinati (*g-cell*, *g-per-row*, ecc…), ogni volta che si attiva il focus su questo modulo a seguito di un'azione *fire*, è necessario aggiornare le proprie conoscenze, passando il focus al modulo AGENT_CELL_BASE_INFERENCE. Questo modulo provvede ad effettuare tutte le inferenze di cui abbiamo discusso nella sezione precedente.

Una volta terminato il processo di inferenza, si asserisce il fatto (*agent-updated*), che viene ritrattato in caso di nuove azioni di *fire*. In seguito si provvede a stimare la probabilità di ciascuna cella nel contenere un pezzo di nave. É bene dire che per le celle che hanno probabilità pari a 1 non viene ricalcolata nessuna probabilità. Per le altre celle, la probabilità ci ciascuna di esse è data dalla media aritmetica di due valori (un valore per riga e uno per colonna): il numero di celle che contengono una nave diviso il numero di celle candidate a contenere qualcosa. Ad esempio per la cella presente in riga 4 e colonna 3, si ottiene il valore *r1* come il numero di celle della riga 4 ancora da individuare e che contengono un pezzo di nave, fratto il numero di celle per la stessa riga che potrebbero contenere qualcosa; in maniera analoga si calcola *c1* per la colonna 3, e lo score della cella è dato da [(*r1* + *c1*) / 2].

Per mantenere i dati aggiornati, quando si inferisce che una cella (*cella1*) ha probabilità certa di contenere qualcosa, allora si aggiorna la lista delle celle candidate per la relativa colonna e la relativa riga, togliendo di fatto la cella da questa lista. Inoltre si decrementa di 1 il contatore che indica il numero di celle che contengono un pezzo di nave ma che non sono state ancora individuate (*num*), dato che *cella1* è stata individuata.

Una volta calcolate le probabilità, si passa il focus al modulo AGENT_DECISION che provvede ad effettuare un'azione di *fire* o *guess*. Abbiamo sviluppato due differenti versioni di questo modulo che si basano su ragionamenti leggermente differenti e denominati agent*_greedy* e *agent_standard*.

Entrambi gli agenti effettuano l'azione *fire* sulle celle aventi maggiore probabilità di contenere una nave, anche se *agent_standard* preferisce sparare prima sulle celle sicure per le quali il contenuto è ignoto, invece che sulle celle sicure per le quali si conosce il contenuto: in questa maniera si ottiene maggiore conoscenza sulla griglia.

Agent_greedy (che non ha nulla a che vedere con gli algoritmi di ricerca greedy), terminate tutte le azioni di *fire* effettua 20 guess sulle restanti celle, partendo da quelle con maggiore probabilità. Qualora si possa ancora effettuare una *guess* ma le restanti celle hanno probabilità 0, si decide di effettuare la "solve".

Agent_standard, oltre al differente comportamento per le *fire*, è più cauto per le *guess*. Infatti si decide di effettuare una *guess* su una cella soltanto se la probabilità è maggiore o uguale ad una certa soglia. La soglia da noi impostata è 0.4, anche se abbiamo notato che valori differenti provocano migliori score in alcune mappe ma peggiori score in altre mappe.

Mappe e risultati

Per sperimentare il comportamento dell'agente sono state create diverse mappe e in base alla conoscenza a priori sul contenuto di alcune celle, si sono definiti diversi scenari. Le mappe definite sono tre:

1. Mappa con disposizione di navi vicine tra loro;
2. Mappa con disposizione di navi lontane tra loro;
3. Mappa con disposizione non uniforme delle navi.

Per ognuna di queste mappe è stato valutato il comportamento dell'agente in base a quante celle sono conosciute a priori:

1. Nessun indizio;
2. Pochi indizi: l'agente conosce il contenuto di 4 celle della mappa;
3. Molti indizi: l'agente conosce il contenuto di 6/7 celle della mappa.

Per poter avere un riscontro immediato delle azioni dell'agente e dei risultati ottenuti, oltre a raccogliere lo score è stata realizzata una piccola interfaccia grafica che raffigura le azioni intraprese dall'agente.

Qui di seguito vengono esposti gli score secondo i due possibili modi di effettuare le azioni. Saranno poi mostrati i risultati degli scenari evidenziati in rosso e in verde, per entrambi i casi.

Agente standard

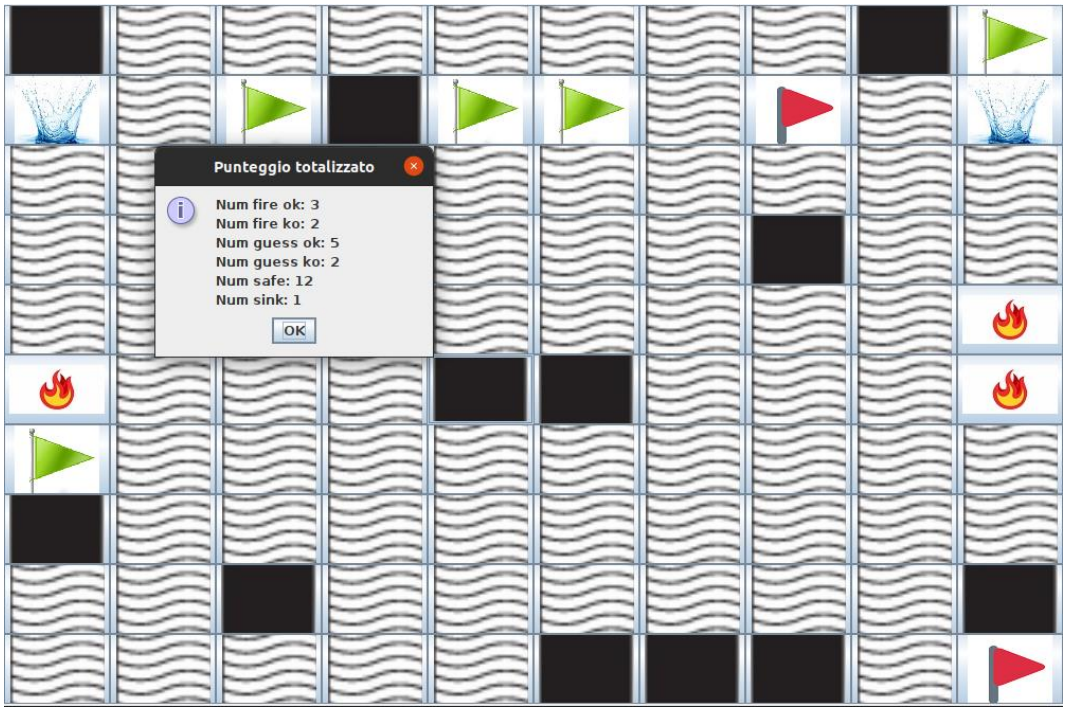
Mappa/Indizi	Nessuno	Pochi	Molti
Disp. sparsa	**-105**	105	215
Disp. vicine	160	235	**280**
Disp. mista	**-185**	195	205

Agente greedy

Mappa/Indizi	Nessuno	Pochi	Molti
Disp. sparsa	**-80**	25	205
Disp. vicine	25	205	**280**
Disp. mista	**-30**	175	185

Agente standard

Disposizione di navi sparsa e nessun indizio:

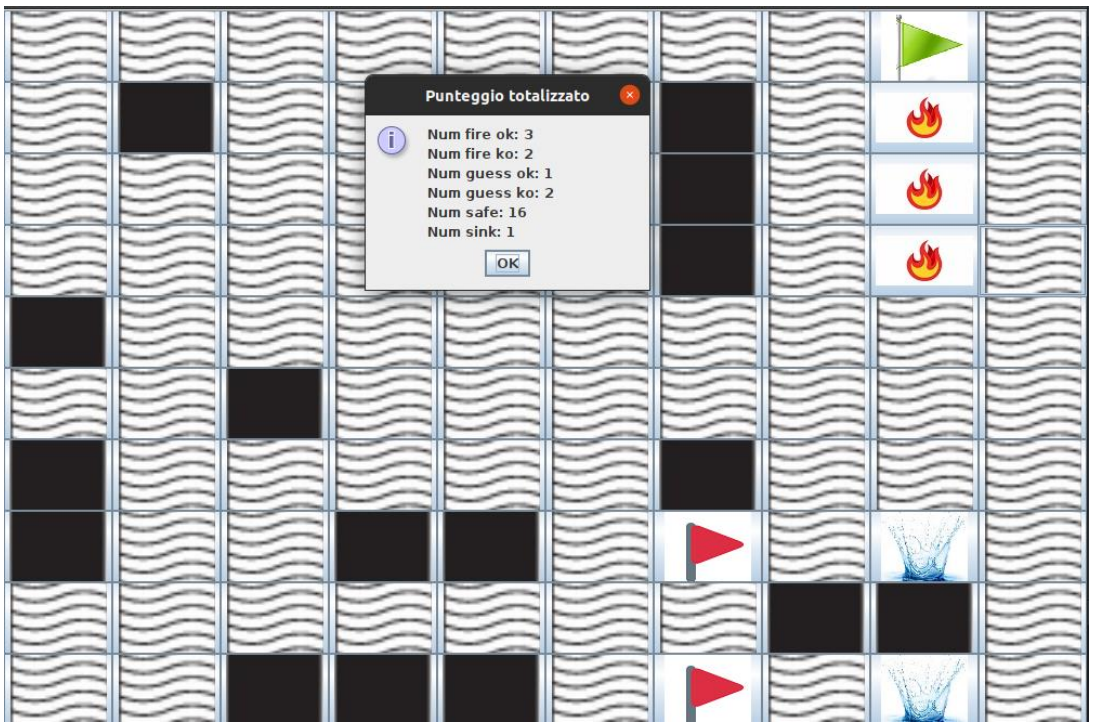


Legenda dei risultati grafici:

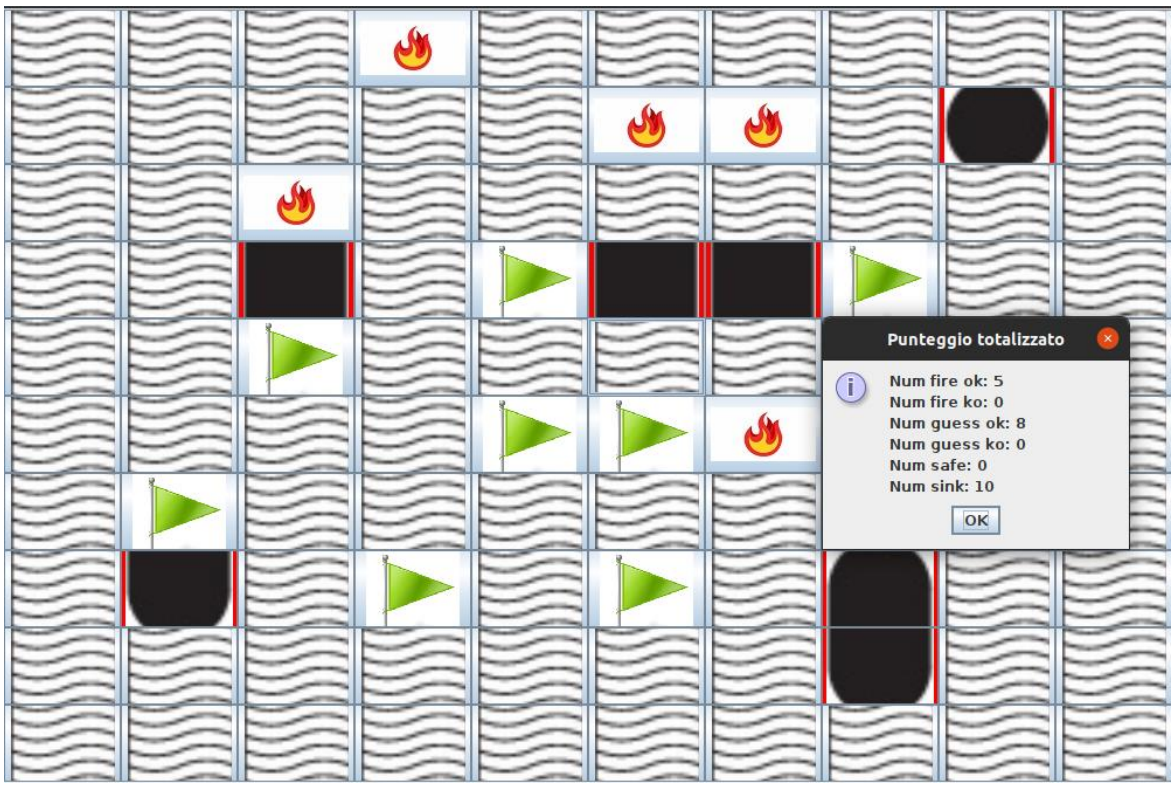
- Bandierina verde: guess corretta
- Bandierina rossa: guess non corretta
- Fuoco: fire corretta
- Acqua: fire non corretta

Inoltre quando una casella è evidenziata con dei bordi rossi, significa che l'agente era a conoscenza del suo contenuto dall'inizio della partita.

Disposizione di navi mista e nessun indizio:

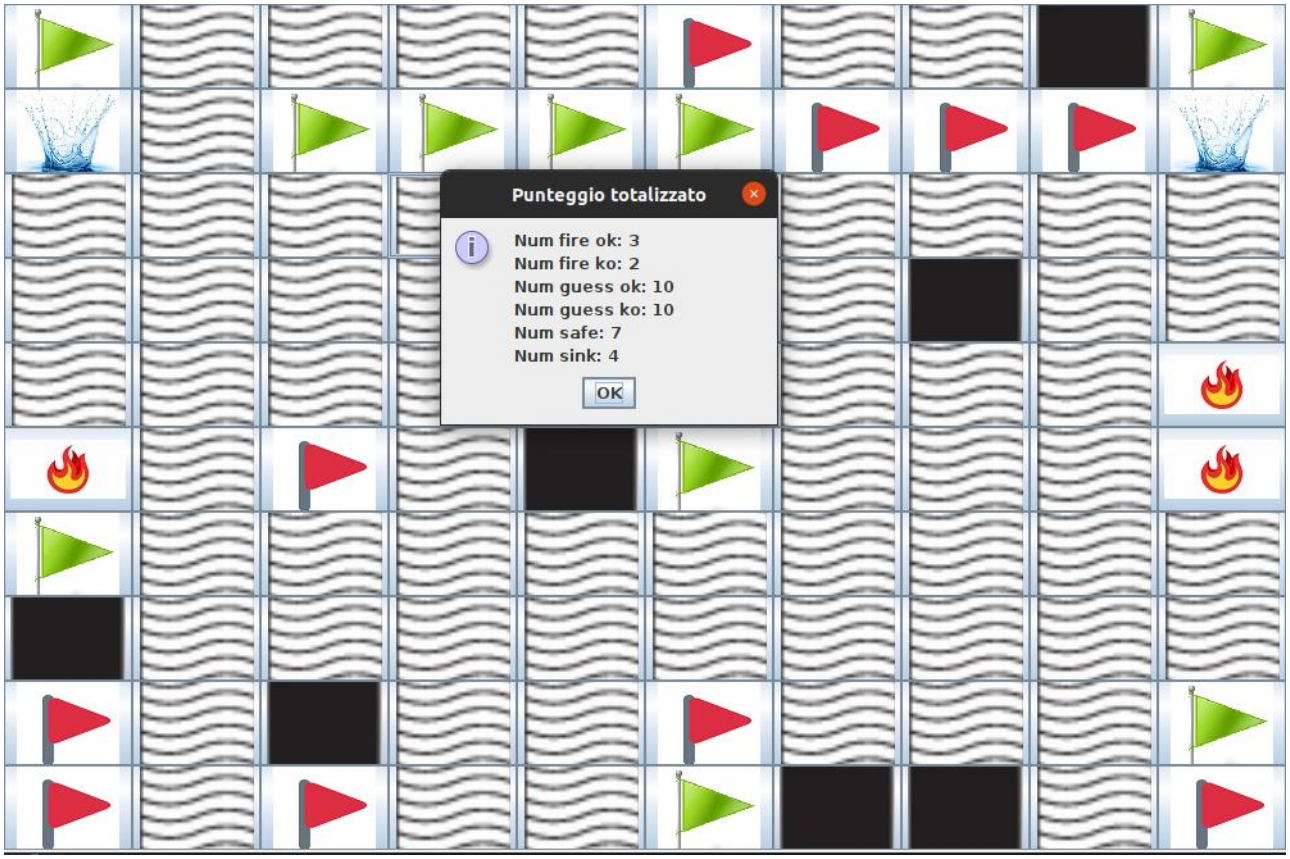


Disposizione di navi vicine e molti indizi:

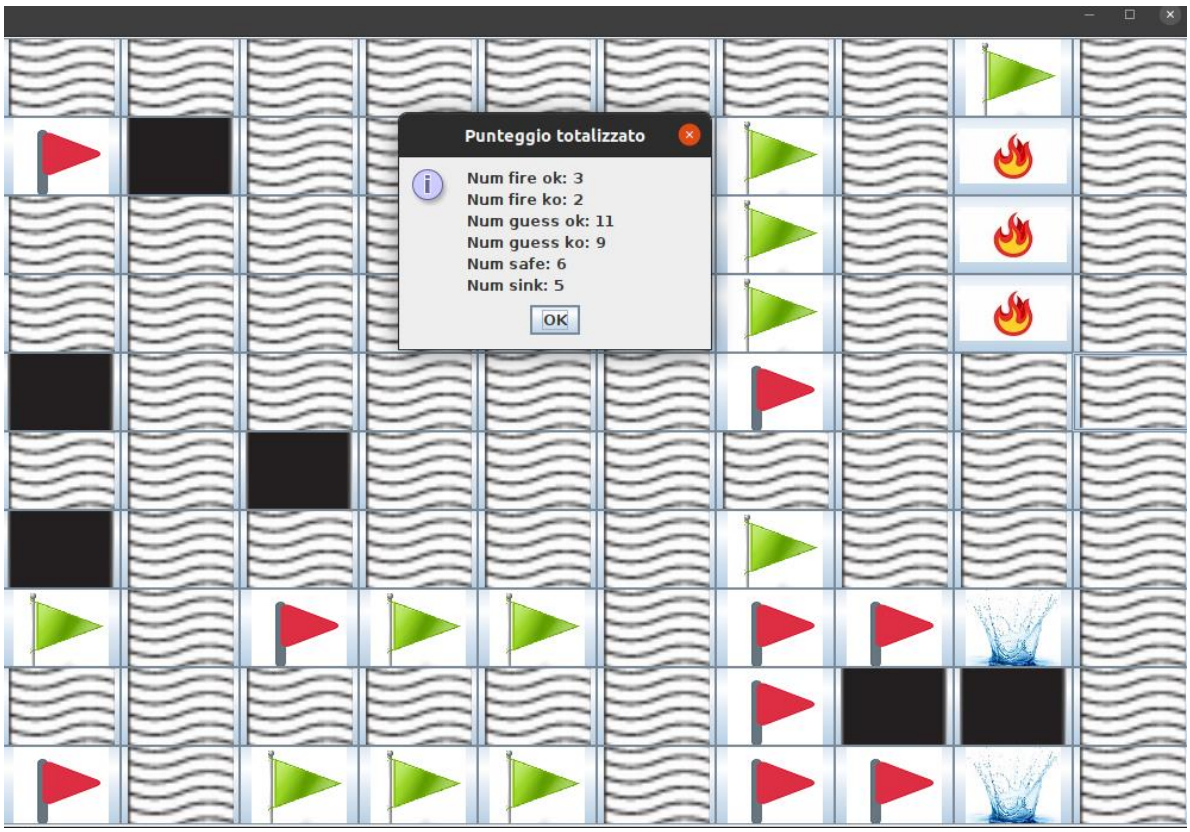


Agente greedy

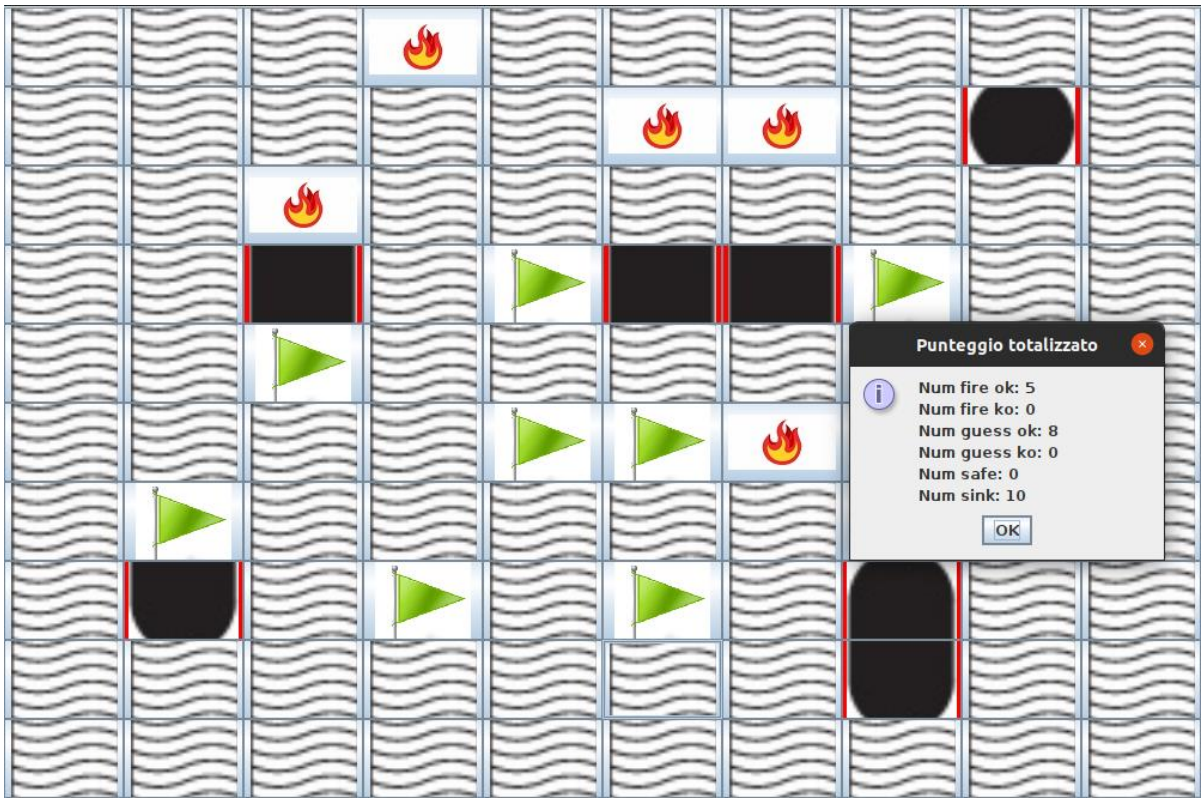
Disposizione di navi sparsa e nessun indizio:



Disposizione di navi mista e nessun indizio:



Disposizione di navi vicine e molti indizi:



Considerazioni finali

Come era plausibile in entrambi i casi si riscontrano performance migliori quando si hanno maggiori indizi a priori.

I punteggi degli scenari con navi vicine tra loro sono i più alti: questo perché conoscendo per ogni riga e colonna il numero di celle contenenti porzioni di navi, ed essendo quest'ultime concentrate, gli agenti riescono ad trovare molte celle che contengono acqua.

Confrontando i due tipi di agenti, si noti come l'agente standard abbia perfomance migliori quando possiede qualche indizio sulla mappa, dato che effettua delle guess piuttosto sicure e non rischia di perdere punti per delle guess errate; cosa che invece l'agente greedy non fa.

In corrispondenza di scenari senza indizi, l'agente greedy ottiene un punteggio più alto perché riesce a trovare qualche cella, mentre la controparte effettua pochissime guess, e trovandosi di fatto in una situazione di stallo.

Un limite dell'agente è quello di effettuare delle scelte basate sulla probabilità di singole celle, piuttosto che su probabilità di cluster (o combinazioni) di celle. Infatti l'agente effettua delle *guess* o delle *fire* su quelle celle che possiedono un'alta probabilità di avere una nave, ma in alcuni casi questa strategia si rivela inefficace. Sarebbe invece preferibile implementare un sistema più complesso che permetta di valutare possibili disposizioni per tutte le navi sulla griglia, e magari effettuare delle *fire* che permettano di ridurre di molto l'insieme delle possibili disposizioni delle navi, con la conseguenza di effettuare delle *guess* con maggior successo.