**BATTAGLIA NAVALE IN CLIPS**

*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 della battaglia navale all’interno di una griglia 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**

All’inizio della progettazione del sistema esperto, abbiamo pensato alle diverse regole di inferenza che permettono di avere una conoscenza maggiore sul contenuto delle celle.

Solo dopo aver pensato a ciò che l’agente avrebbe dovuto inferire si è passati alla definizione dei template che di fatto permettono l’asserzione di nuova conoscenza. Qui di seguito vengono elencati i template da noi definiti.

(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 esisteva già il fatto non ordinato “k-cell”, abbiamo avuto l’esigenza di definirne uno similare ma con qualche informazione in più, e che permettesse anche la trattazione di quelle celle per le quali non si conosce il contenuto delle stesse. Oltre alle coordinate x e y abbiamo introdotto una variabile indicante la probabilità per la cella di contenere un pezzo di nave; inoltre abbiamo introdotto due variabili booleane, updated e considered, ove la prima permette di aggiornare le stime di probabilità e la seconda fa sì che una volta che è stata effettuata un’azione di fire o guess per quella cella, essa non venga più inclusa in successive azioni.

Ogni volta che c’è una nuova k-cell, vengono aggiornate le informazioni della relativa g-cell. Inoltre le g-cell 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 sconosciute che contengono una nave. Quando “num” è valorizzato a 0, significa che per quella riga/colonna sono state trovate tutte le celle contenenti una nave e che quindi le rimanenti celle possono venire escluse da azioni di fire o guess, dato che conterranno sicuramente acqua.

Il multislot “g-cells” contiene le celle per le quali non si sa se contengano con certezza un pezzo di nave o meno, mentre il multislot “gk-cells” contiene sia le celle del multislot precedente, e in aggiunta contiene quelle celle per le quali non si conosce esattamente il contenuto, anche se la probabilità di contenere qualcosa è pari a 1.

(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 scoperte, che hanno una determinata grandezza ed un orientamento. Qualora l’orientamento sia orizzontale (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, anche se di fatto nel sistema vengono generate le sole combinazioni di navi di grandezza 4 o 3. Qualora l’orientamento sia orizzontale (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 al suo interno solo quelle celle (identificate da riga e colonna) per le quali la probabilità di contenere una nave non è certa. Ciò ha permesso di effettuare particolari tipi di inferenze, infatti qualora ci sia una combinazione orizzontale sulla riga 3 per una nave di grandezza 4, e per questa combinazione 2 celle di essa non hanno la certezza di contenere una nave, qualora sulla riga 3 c’è il fatto (g-per-row (num 1)), allora si può dedurre che in quella riga ci sarà soltanto un’altra cella occupata dalla nave, e quindi questa combinazione può venire scartata perché inconsistente.

**Regole di inferenza**

Così come la regola di inferenza appena citata, abbiamo definito diverse regole che possono essere classificate in tre categorie principali:

* regole basate sul contenuto di una cella più qualche cella ad essa adiacente;
* regole basate sulle nave trovate e quelle ancora da trovare;
* regole basate sulle possibili combinazioni di navi.

Lo scopo di ciascuna di queste regole è quello di stabilire la probabilità per una cella di contenere una nave, e possibilmente anche il suo contenuto.

Il primo gruppo di regole si basa sul contenuto delle celle e permette di fare inferenze sulle celle adiacenti. Ad esempio, dato che una nave non può trovarsi vicino ad un’altra nave, qualora io sapessi che una cella ha il contenuto “top”, si può dedurre che le celle immediatamente sopra così come le celle laterali possiedano con certezza dell’acqua, mentre la cella che si trova in basso possiede con certezza un pezzo di nave (la stessa di quella della cella top), anche se il contenuto della stessa è sconosciuto.

Una cella “sub” invece avrà sicuramente delle celle con dell’acqua intorno a se. In sostanza queste regole hanno lo scopo di rimuovere diverse celle grazie alla conoscenza di una singola cella .

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

Se ad esempio la nave da 4 è stata trovata, qualora si sappia che 3 celle contigue hanno sicuramente un pezzo di nave, si può dedurre che le 3 celle facciano riferimento ad una nave da 3 e quindi si può dedurre anche il loro contenuto.

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

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

**Moduli dell’agente**

Il nostro sistema esperto è composto da 3 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’inizializzazione 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, qualora sia necessario aggiornare le proprie conoscenze, si passa il focus al modulo AGENT\_CELL\_BASE\_INFERENCE che provvede ad effettuare tutte le inferenze di cui abbiamo discusso sopra. Una volta che sono state aggiornate tutte le celle si asserisce il fatto (agent-updated), che viene ritrattato in caso di nuove fire.

Una volta terminato il processo di inferenza, si ritorna al modulo AGENT che provvede a stimare la probabilità di ciascuna cella nel contenere una nave. E’ bene dire che per le celle che hanno sicuramente una nave non viene ricalcolata nessuna probabilità. Per il resto delle celle, la probabilità di una cella è data dalla media aritmetica di due fattori (uno per riga e uno per colonna): il numero di celle che contengono qualcosa fratto il numero di celle candidate a contenere qualcosa. Quando una cella ha probabilità pari a 1 di contenere qualcosa, allora per la relativa colonna e riga si rimuove la cella dalle celle candidate e si decrementa di uno il numero di celle che contengono qualcosa.

Una volta calcolate le probabilità, si passa il focus all’ultimo modulo che provvede ad effettuare una decisione. Abbiamo sviluppato due differenti versioni di questo modulo che si basano su ragionamenti leggermente differenti: agent\_greedy e agent\_standard.

Entrambi gli agenti effettuano l’azione fire sulle celle aventi maggiore probabilità di avere una nave, anche se agent\_standard se può sparare su delle celle sicure, spara prima su quelle celle per le quali il contenuto è ignoto, in maniera tale da ottenere maggiore conoscenza sulla griglia.

Agent\_greedy, una volta effettuate tutte le fire, effettua massimo 20 guess su tutte le celle, partendo da quelle con maggiore probabilità. Qualora si possa effettuare ancora qualche guess ma tutte le celle hanno probabilità 0, si decide di effettuare la “solve”.

Agent\_standard, oltre al diverso comportamento nelle 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 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 combinazione con quanto l’agente conosca a priori sul contenuto di alcune caselle, si sono definiti diversi **scenari**.

Le mappe sviluppate sono state tre:

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

Per ognuna di queste mappe è stato testato il comportamento dell’agente in base a quanti indizi sono forniti inizialmente:

1. Nessun indizio
2. Pochi indizi: l’agente conosce il contenuto di 4 caselle della mappa
3. Molti indizi: l’agente conosce il contenuto di 6/7 caselle 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 espone le azioni intraprese all’agente.

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.

Qui di seguito vengono esposti gli score per l’agente, secondo i due possibili modi di effettuare le azioni.

***Agente standard***

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

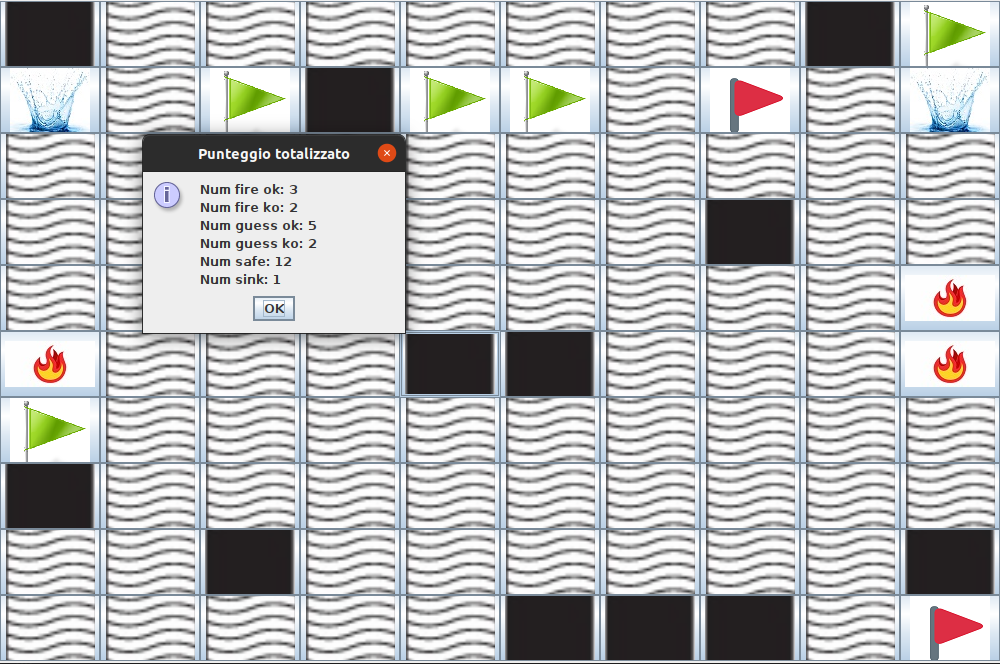
***Agente greedy***

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

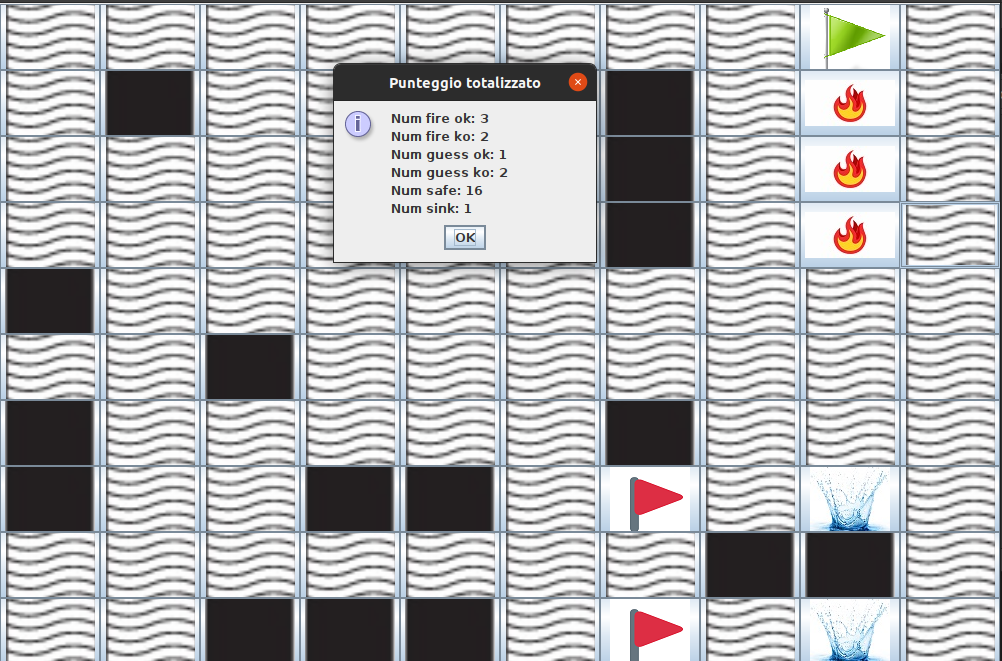
Di seguito sono mostrati graficamente i risultati degli scenari evidenziati in rosso o in verde, per entrambi i decisori.

***Agente standard***

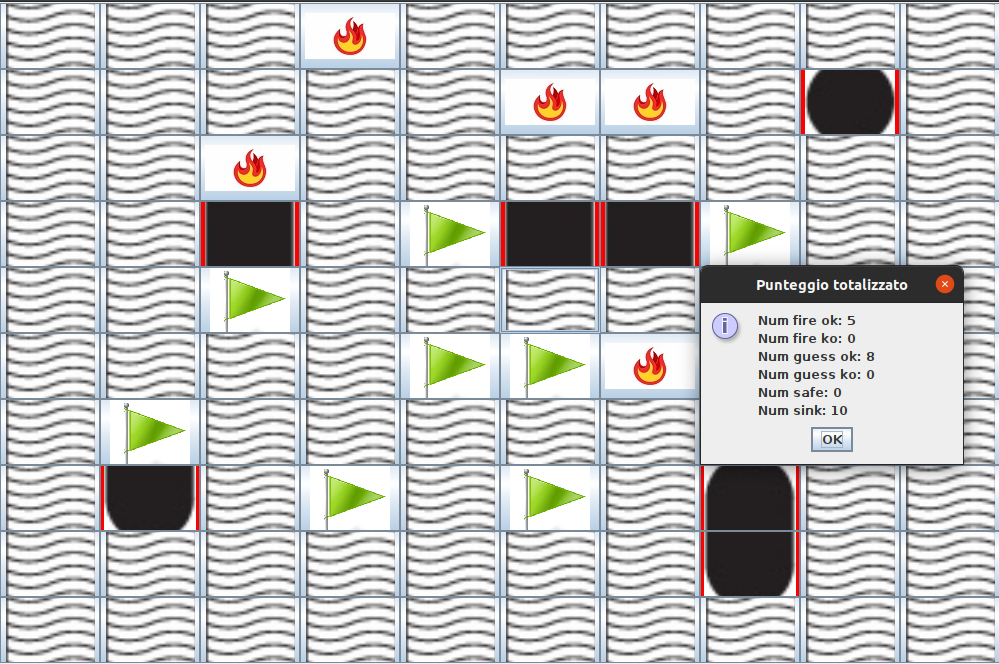
Disposizione navi sparsa e nessun indizio:



Disposizione navi normale e nessun indizio:

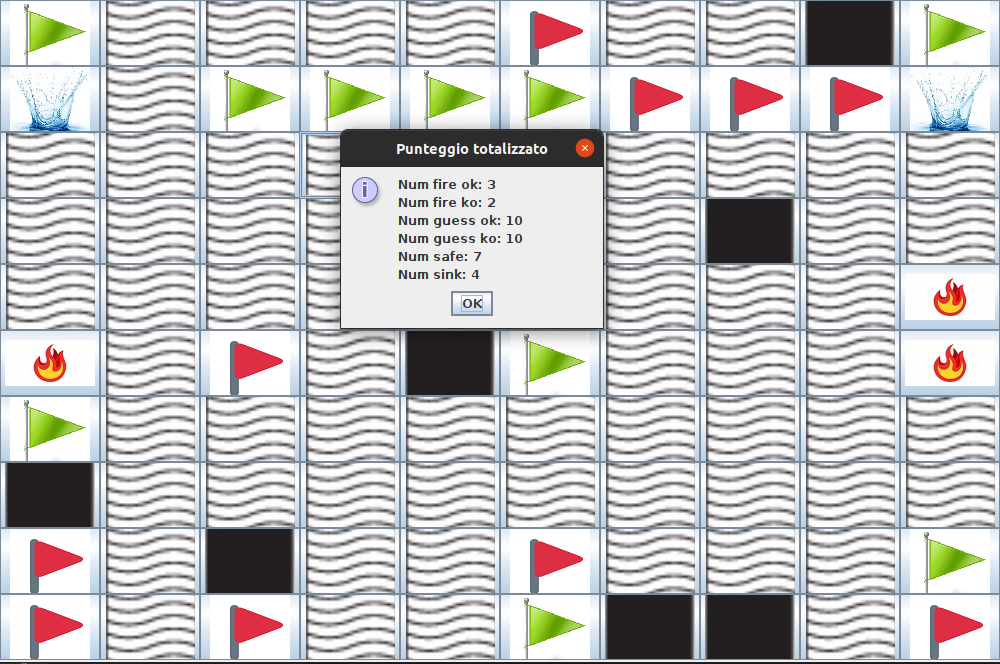


Disposizione navi vicine e molti indizi:

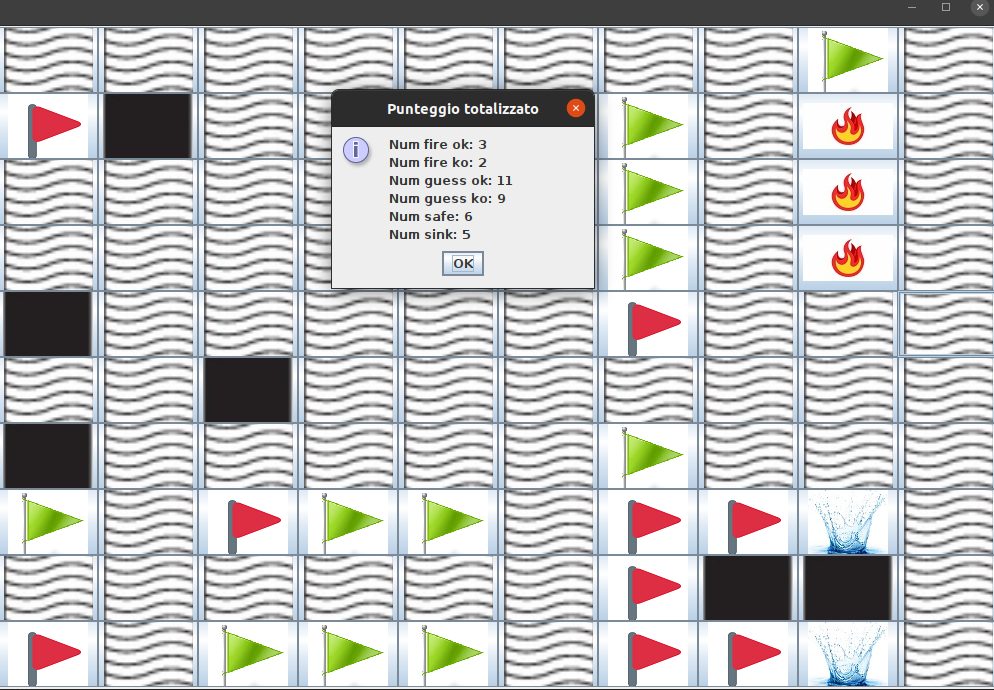


***Agente greedy***

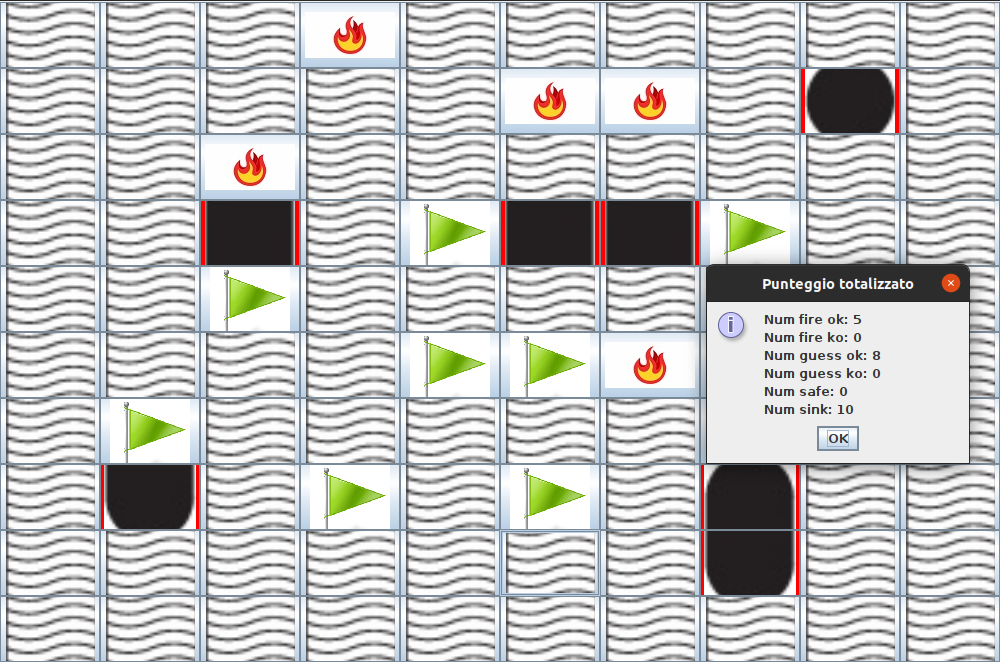
Disposizione navi sparsa e nessun indizio:



Disposizione navi normale e nessun indizio:



Disposizione navi vicine e molti indizi:

****

**Considerazioni finali**

Come era plausibile entrambi i decisori riscontrano performance migliori quando 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, essendo quest’ultime concentrate gli agenti riescono ad inferire ed escludere molte celle che contengono acqua nonostante non ricevano alcun indizio.

Confrontando i due decisori, si noti come agente standard abbia perfomance migliori quando possiede qualche indizio sulla mappa, visto che effettua delle guess piuttosto sicure e non rischia di perdere punti per delle guess errate, come avviene invece per l’altro decisore. In corrispondenza di scenari con nessun indizio, l’agente greedy ottiene un punteggio più alto perché la controparte non riesce ad effettuare nessuna guess con sicurezza, e quindi si trova in una situazione di stallo.

Un limite dei due decisori è quello relativo al fatto che effettuano delle scelte basate su singole celle, piuttosto che su possibili combinazioni di navi. Pertanto provano ad effettuare delle guess o delle fire su quelle celle che possiedono un’alta probabilità di avere una nave, ma in alcune combinazioni questa strategia si rivela inefficace. Sarebbe invece preferibile implementare un sistema più articolato che permetta di valutare possibili disposizioni di tutte le navi sulla griglia, ed effettuare delle fire o guess che permettano di ridurre di molto l’insieme delle possibili disposizioni delle navi.