AI LAB 2022-23
Progetto clips

Frigato Luca

Reuirements

Implementazione di un sistema esperto, in grado di dedurre la posizione del maggior numero possibile di navi del gioco Battaglia Navale.

Reuirements

Individuazione delle Navi: L’agente deve determinare la posizione esatta di tutte le navi nemiche.

Limiti delle Risorse: L’agente ha risorse limitate, come le azioni "fire" e "guess." Deve utilizzare queste risorse con saggezza per ottenere il massimo punteggio possibile.

Gestione delle Informazioni: L’agente deve tenere traccia delle informazioni disponibili sulla griglia, come celle già "guessed" o "fired," e utilizzare queste informazioni per prendere decisioni informate.

→ Associata ad ogni riga e colonna, vi è indicato il numero di celle che contengono barche.

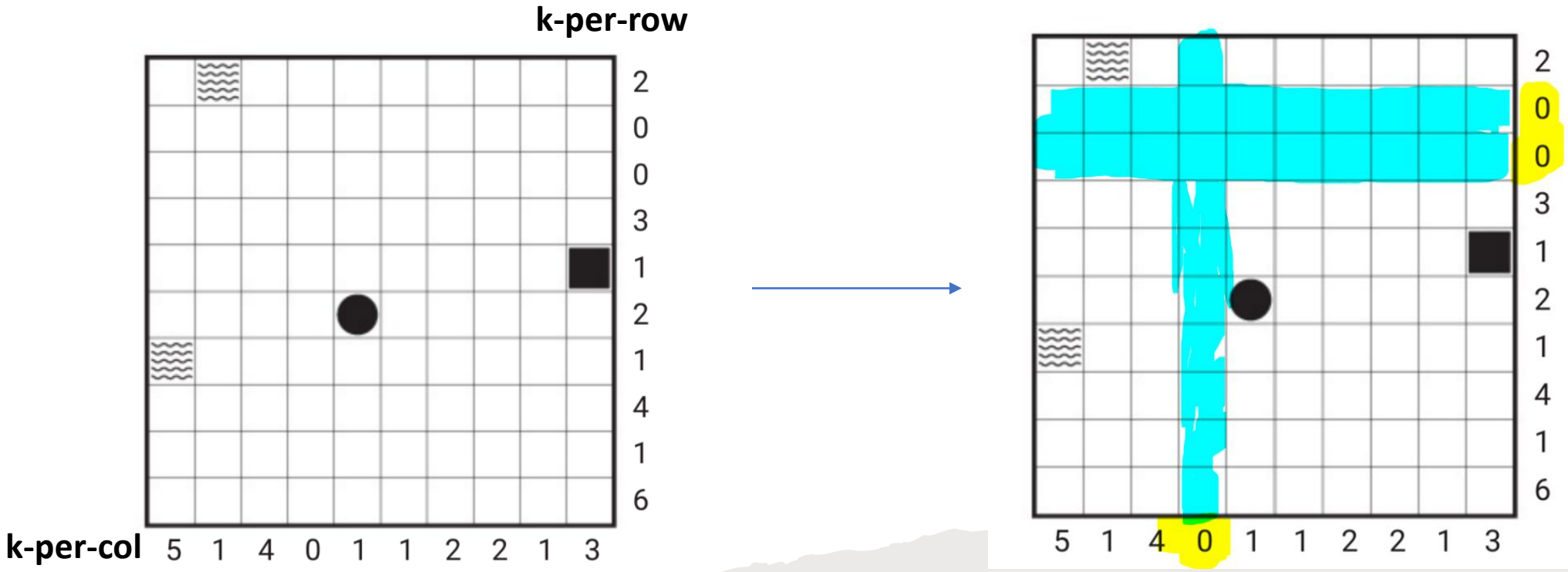
Reuirements

Il numero limitato di mosse impedisce l'adozione di una strategia "colpisci e osserva" tipica del gioco classico.
Le informazioni sulle righe e colonne richiamano giochi matematici come il Sudoku.

L'analogia con la battaglia navale è limitata alle prime mosse dove è essenziale utilizzare ogni «**fire**» per apprendere info sull'ambiente.

Primo Agente: Strategia e Loop

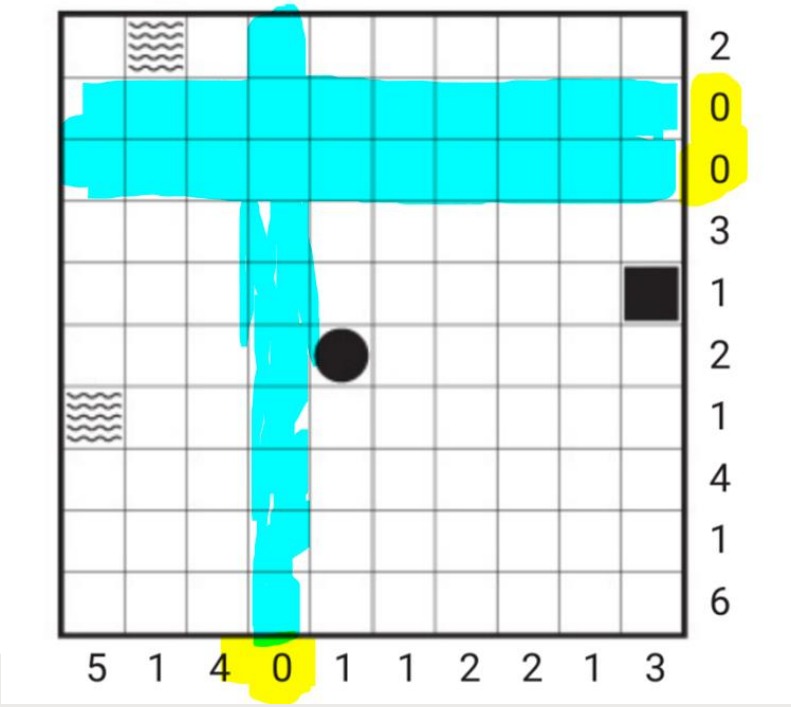
L'idea principale è quella di inizializzare gran parte della griglia con celle contenenti ACQUA.
Per farlo viene sfruttata l'informazione accessibile tramite i facts **k-per-row** e **k-per-col:**



Primo Agente: Strategia e Loop

L'idea principale è quella di inizializzare gran parte della griglia con celle contenenti ACQUA.
Per farlo viene sfruttata l'informazione accessibile tramite i fatti **k-per-row** e **k-per-col:**

Nello specifico, questo permette all'agente di asserire ACQUA in intere righe e colonne.

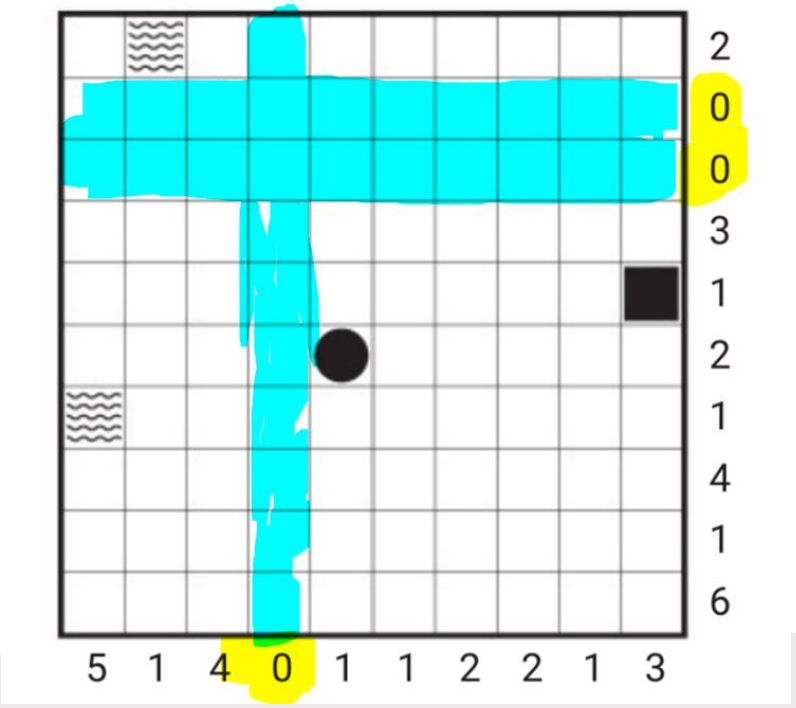


Primo Agente: Strategia e Loop

L'idea principale è quella di inizializzare gran parte della griglia con celle contenenti ACQUA.
Per farlo viene sfruttata l'informazione accessibile tramite i fatti **k-per-row** e **k-per-col:**

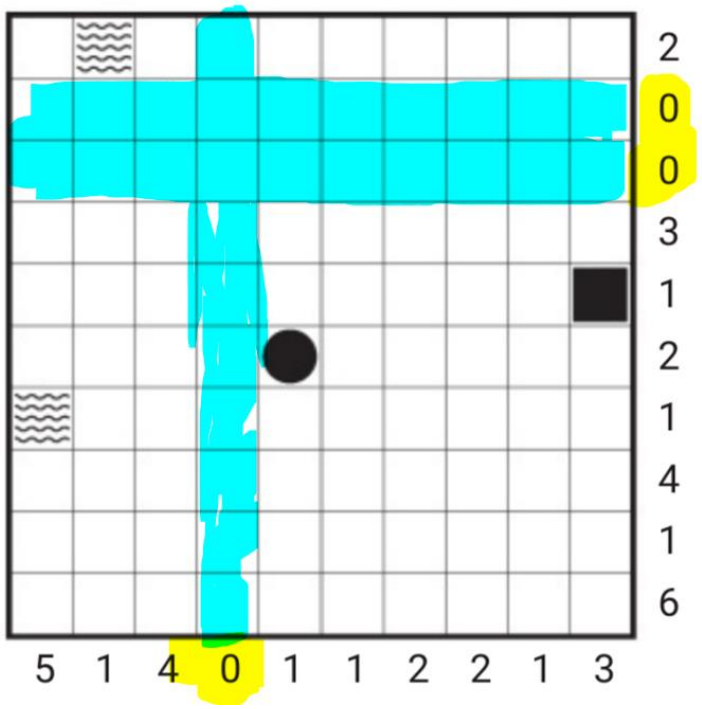
Nello specifico, questo permette all'agente di asserire ACQUA in intere righe e colonne.

Questa info può essere utilizzata successivamente dall'agente per compiere decisioni.



Primo Agente: Strategia e Loop

Viene così definito un **battle-field** complementare a quello pre-generato tramite l'editor di mappe




```
(deffacts agent-battle-field
    (agent-cell (x 0) (y 0) (content unknown) )
    (agent-cell (x 0) (y 1) (content unknown) )
    (agent-cell (x 0) (y 2) (content unknown) )
    (agent-cell (x 0) (y 3) (content unknown) )
    (agent-cell (x 0) (y 4) (content unknown) )
    (agent-cell (x 0) (y 5) (content unknown) )
    (agent-cell (x 0) (y 6) (content unknown) )
    (agent-cell (x 0) (y 7) (content unknown) )
    (agent-cell (x 0) (y 8) (content unknown) )
    (agent-cell (x 0) (y 9) (content unknown) )
    (agent-cell (x 1) (y 0) (content unknown) )
    (agent-cell (x 1) (y 1) (content unknown) )
    (agent-cell (x 1) (y 2) (content unknown) )
    (agent-cell (x 1) (y 3) (content unknown) )
    (agent-cell (x 1) (y 4) (content unknown) )
    (agent-cell (x 1) (y 5) (content unknown) )
```

Primo Agente: Strategia e Loop

Viene così definito un **battle-field** complementare a quello pre-generato tramite l'editor di mappe

```
(deffacts agent-battle-field
    (agent-cell (x 0) (y 0) (content unknown) )
    (agent-cell (x 0) (y 1) (content unknown) )
    (agent-cell (x 0) (y 2) (content unknown) )
    (agent-cell (x 0) (y 3) (content unknown) )
    (agent-cell (x 0) (y 4) (content unknown) )
    (agent-cell (x 0) (y 5) (content unknown) )
    (agent-cell (x 0) (y 6) (content unknown) )
    (agent-cell (x 0) (y 7) (content unknown) )
    (agent-cell (x 0) (y 8) (content unknown) )
    (agent-cell (x 0) (y 9) (content unknown) )
    (agent-cell (x 1) (y 0) (content unknown) )
    (agent-cell (x 1) (y 1) (content unknown) )
    (agent-cell (x 1) (y 2) (content unknown) )
    (agent-cell (x 1) (y 3) (content unknown) )
    (agent-cell (x 1) (y 4) (content unknown) )
    (agent-cell (x 1) (y 5) (content unknown) )
```

Oltre a questo, torna utile all'agente manipolare e aggiornare anche i valori degli indici **k-per-row** e **k-per-col,** che quindi sono stati clonati e rinominati

agent-per-col e **agent-per-row.**

```
(deftemplate agent-per-row
    (slot row)
    (slot num)
)

(deftemplate agent-per-col
    (slot col)
    (slot num)
)
```

Primo Agente: Strategia e Loop

1. Inizializzazione e memorizzazione delle info tabellari e flooding della «scacchiera» andando ad editare la griglia doppione puntalmente.

```
(defrule flood-rows (declare (salience 30))
    (status (step 0)(currently running))
    (agent-per-row (row ?r) (num 0))
    ?cell <- (agent-cell (x ?r) (y ?c) (content unknown) )
=>
    ; (printout t "WATER on ROW " ?r crlf)
    (modify ?cell (content water))
)
```

Strategy ->depth

Salience a 30 **->** evitare che si attivi una fire prima di avere la **agent-battle-field** correttamente annotata.

(step 0) -> la eseguo solo al primo «ciclo»

Primo Agente: Strategia e Loop

1. Inizializzazione e memorizzazione delle info tabellari e flooding della «scacchiera» andando ad editare la griglia doppione puntalmente.

```
(defrule clone-k-per-rows
    (declare (salience 31))
    (status (step 0))
    (k-per-col (col ?c) (num ?n))
    (not (agent-per-col (col ?c) (num ?nc)))

=>
    (assert (agent-per-col (col ?c) (num ?n)))
)
```

Salience a 31 **->** priorità alla memorizzazione

(step 0) -> la eseguo solo al primo «ciclo»

Primo Agente: Strategia e Loop

2. Asserzione delle celle note a priori: viene estratta dalle **k-cell**

```
(defrule observed-data-no-water (declare (salience 20))
    ?ag_cell <- (agent-cell (x ?x)(y ?y)(content unknown))
    ?k_cell <- (k-cell (x ?x) (y ?y) (content ?c&:(neq ?c water)))
    ?nrow <- (agent-per-row (row ?x) (num ?nr))
    ?ncol <- (agent-per-col (col ?y) (num ?nc))
=>
    ; (printout t "pre ship per row/col [" ?x ", " ?y "]  " ?nr "|"  ?nc crlf)

    (modify ?ag_cell (content ?c))
    (bind ?nr (- ?nr 1))
    (modify ?nrow (num ?nr))
    (bind ?nc (- ?nc 1))
    (modify ?ncol (num ?nc))
    (printout t "nrow-observed in pos [
    ; (printout t "pre ship row/col ["

)
```

```
(defrule observed-data-water  (declare (salience 20))
    ?ag_cell <- (agent-cell (x ?x)(y ?y)(content unknown))
    ?k_cell <- (k-cell (x ?x) (y ?y) (content water))
=>

    (modify ?ag_cell (content water))
    (printout t "last shot fired was on water"  crlf)

)
```

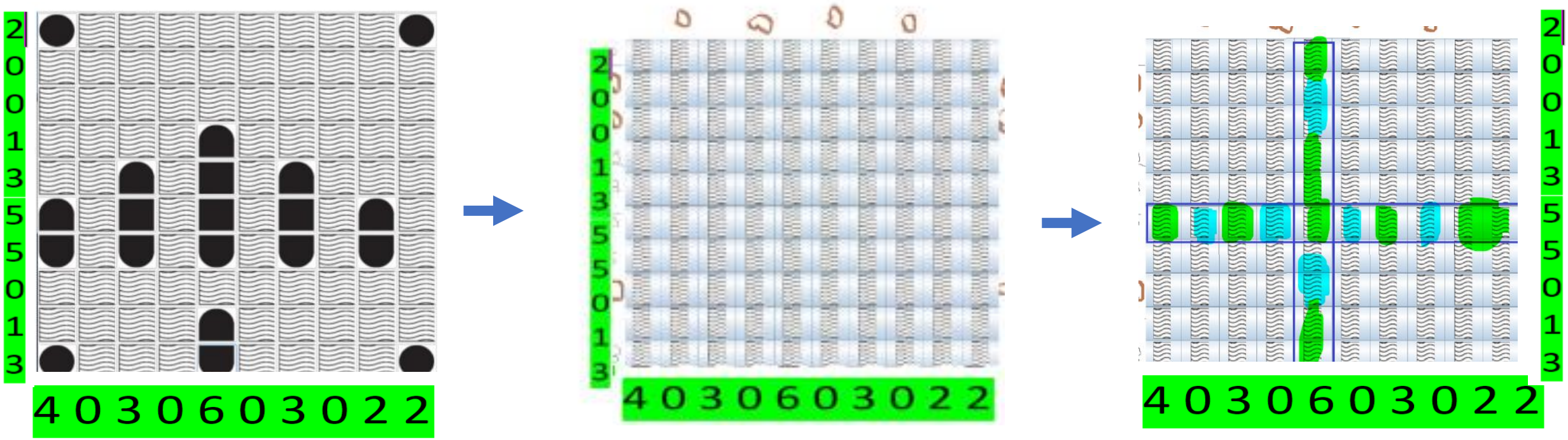
- update della griglia interna, dove viene indicato il contenuto reale della cella

Primo Agente: Strategia e Loop

3. Fire e Guess: si possono da questo momento in poi interfogliare FIRE a GUESS.

L'agente non ha però una vera concezione dell'ambiente e sia FIRE sia GUESS lavorano su una semplice idea:

le celle migliori da selezionare sono quelle che si trovano a coordiate dove **k-per-col** e **k-per-row** sono alti.
Generalmente mi sarebbe piaciuto anche imporre

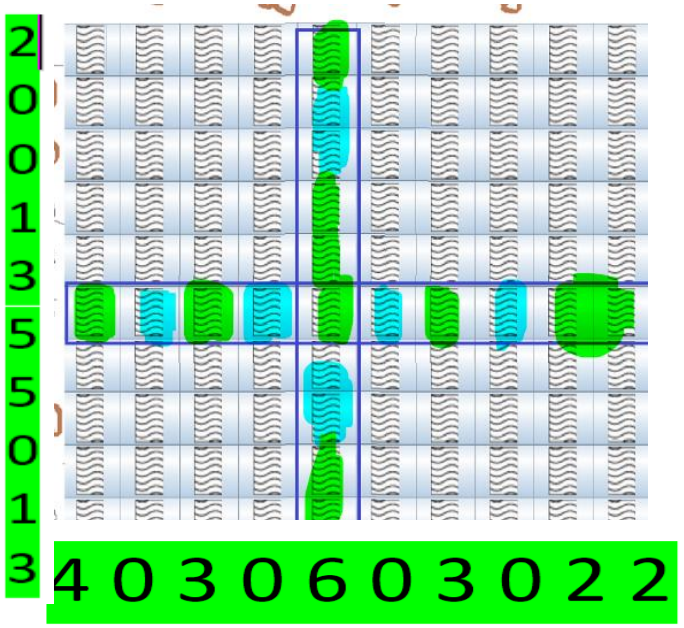


Primo Agente: Strategia e Loop

3. Fire e Guess: si possono da questo momento in poi interfogliare FIRE a GUESS.

L'agente non ha però una vera concezione dell'ambiente e sia FIRE sia GUESS lavorano su una semplice idea:

le celle migliori da selezionare sono quelle che si trovano a coordiate dove **k-per-col** e **k-per-row** sono alti. Generalmente mi sarebbe piaciuto anche imporre



Max k-per-col e max-k-per-row sono agli indici **[5,4]**

Immaginando di aver inondato le celle su righe o colonne a 0, possiamo dire che solo le zone rimaste verdi all'interno di questa croce sono appetibili per una FIRE o Guess.

Sarebbe interessante anche poter ordinare questi elementi in base ad un criterio, come potrebbe essere proprio la

Somma **k-per-col + max-k-per-row**

NOTA: questo non è stato esplorato

Primo Agente: Strategia e Loop

3. Fire e Guess: si possono da questo momento in poi interfogliare FIRE a GUESS.

L'agente non ha però una vera concezione dell'ambiente e sia FIRE sia GUESS lavorano su una semplice idea:

le celle migliori da selezionare sono quelle che si trovano a coordiate dove **k-per-col** e **k-per-row** sono alti.
Generalmente mi sarebbe piaciuto anche imporre

In base a questa assunzione, sono state definiti RULES per fare FIRE e GUESS proprio su questi elementi.

Generalmente, cerco per prime le celle dove la coppia ROW e COL è più alta.

Qui è dove verrà eseguita la FIRE o GUESS

```
(defrule guess-most-probable-cell (declare (salience 2))
    (moves  (guesses ?ng&:(> ?ng 0)))
    (status (step ?s)(currently running))

    ?nrow <- (agent-per-row (row ?r) (num ?nr))
    ?ncol <- (agent-per-col (col ?c) (num ?nc))
    ; non esiste un altro massimo
    (not (agent-per-row (row ?r2&:(neq ?r2 ?r)) (num ?n2 &:(> ?n2 ?nr))))
    (not (agent-per-col (col ?c2&:(neq ?c2 ?c)) (num ?m2 &:(> ?m2 ?nc))))
    (agent-cell (x ?r) (y ?c)(content ~water))
    (not (exec (action guess) (x ?r) (y ?c)))

=>
    (assert (exec (step ?s) (action guess) (x ?r) (y ?c)))
    (bind ?nr (- ?nr 1))
    (modify ?nrow (num ?nr))
    (bind ?nc (- ?nc 1))
    (modify ?ncol (num ?nc))
    ; (printout t "Guess intera alla  posizione [" ?r ", " ?c "] at step " ?s crlf)
    (pop-focus)
)
```

Primo Agente: Strategia e Loop

3. Fire e Guess: si possono da questo momento in poi interfogliare FIRE a GUESS.

RULE	LHS	RHS
fire-most-probable-cell-row-col	- Ancora almeno 1 mossa FIRE disponibile - La cella rispetta il criterio visto prima (elevata prossimità a navi). - Ha sia max Row sia COL - Non sono state eseguite action su questa cella	Asserita la EXEC di una FIRE su questa cella. Successivamente l'ENV si preoccuperà di generare una k-cell con questa informazione.
fire-most-likelihood-cell-row	- Come la precedente ma la condizione settata si limita a matchare celle con il valore piu alto di prossimità per RIGA	Esegue la fire..

Primo Agente: Strategia e Loop

3. Fire e Guess: si possono da questo momento in poi interfogliare FIRE a GUESS.

RULE	LHS	RHS
fire-most-probable-cell-row-col-only	- Come la precedente ma la condizione settata si limita a matchare celle con il valore piu alto di prossimità per COL	Asserita la EXEC di una FIRE su questa cella.
guess-most-probable-cell-row-col	La guess in questo caso si preoccupa delle sole celle in memoria, agent-per-col, che non siano state settate a Water in precedenza. Come per la **fire-cell-row**, identifica la piu alta combo.	Asserita la EXEC di una GUESS su questa cella Aggiornati i riferimenti della griglia (siccome non si può sapere se sia stata effettivamente affondata una nave)

Primo Agente: Strategia e Loop

3. Fire e Guess: si possono da questo momento in poi interfogliare FIRE a GUESS.

RULE	LHS	RHS
Guess-tail	- E’ tipo un «default» che non considera la combo migliore ROW e COL ma - Matcha tutte le celle che almeno hanno una nave vicina (o sulla riga o sulla colonna)	Asserita la EXEC di Guess su questa cella.

Primo Agente: Strategia e Loop

4. Condizione di uscita: quando non è più applicabile alcuna regola, è richiamata la **SOLVE**.

RULE	LHS	RHS
nothing-more-to-do	- …	Asserita la EXEC su Solve e termina l'esecuzione del tutto.

Primo Agente: Limiti

Implementazione semplice che non tiene conto della tipologia/morfologia delle navi

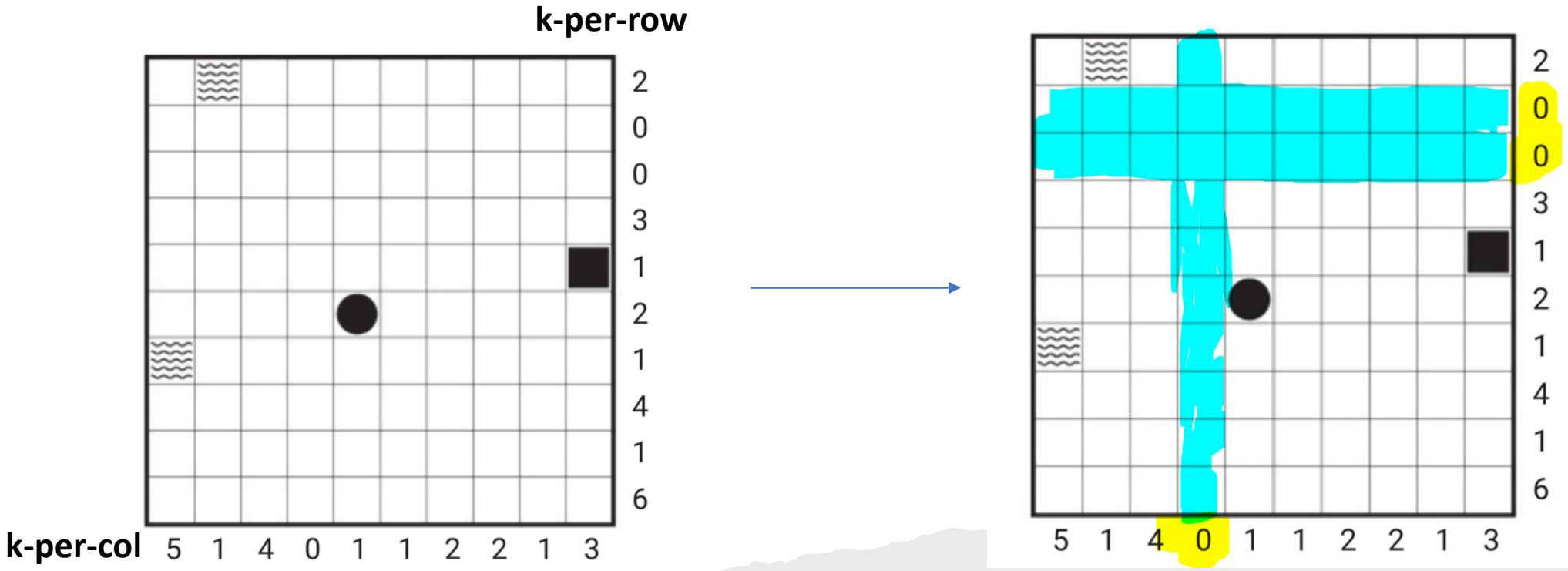
- nel secondo agente sono state introdotte **regole più specifiche** e legate in modo particolare «**content**» della **k-cell asserita** (dal sistema) o dai legami di prossimità che si sono creati tra gli elementi della griglia «interna».

- in un primo momento si è anche cercato di modularizzare l'agente ma con scarsi risultati, in particolare, non sembravano essere triggherare le rules all'interno di un sottomodulo (FIRE Module). Perciò si è tornati ad un approccio a singolo modulo per semplicità.

- non è stato utilizzato alcun meccanismo di backtracking per rivedere le scelte quindi non si è previsto l'utilizzo dell'azione di unguess

Secondo Agente: Strategia e Loop

Anche in questo caso ho riutilizzato la stessa impostazione e lo stesso principio visto con il primo Agente
I facts **k-per-row** e **k-per-col etc…**



Secondo Agente: Strategia e Loop

Anche in questo caso ho riutilizzato la stessa impostazione e lo stesso principio visto con il primo Agente
I facts **k-per-row** e **k-per-col etc…**

1. Inizializzazione e memorizzazione delle info tabellari e flooding della «scacchiera» andando ad editare la griglia doppione puntalmente.

2. Asserzione delle celle note a priori: viene estratta dalle **k-cell**

Secondo Agente: Strategia e Loop

3. Fire e Guess: oltre alle stesse regole viste sul primo agente, sono state aggiunte rule che vanno in trigger principalmente in seguito alla scoperta di nuove barche, più nello specifico delle PARTI di Barca.

Ricordo infatti che **k-cell** specifica nel contenuto a quale tipo di imbarcazione si riferisce.

- un altro requisito è il seguente: almeno una cella di separazione tra ogni barca.
- possiamo di conseguenza aggiungere altre celle «WATER» alla griglia interna dell'agente

```
(defrule update-observed-sub-left (declare (salience 20))
    ?ag_cell <- (agent-cell (x ?x)(y ?y)(content unknown))
    ?k_cell <- (k-cell (x ?x) (y ?y) (content sub))

    ?ag_cell_left <- (agent-cell (x ?x) (y ?y_left & :(- ?y
=>
    (modify ?ag_cell_left (content water))

    (printout t "shot a sub"  crlf)
)
```

```
(defrule update-observed-sub-right (declare (salience 20))
    ?ag_cell <- (agent-cell (x ?x)(y ?y)(content unknown))
    ?k_cell <- (k-cell (x ?x) (y ?y) (content sub))

    ?ag_cell_right <- (agent-cell (x ?x) (y ?y_right & :(+ ?y_right 1))(content unknown))
=>
    (modify ?ag_cell_right (content water))
    (modify ?ag_cell (content water))
    (printout t "shot a sub"  crlf)
```

Secondo Agente: Strategia e Loop

3. Fire e Guess: oltre alle stesse regole viste sul primo agente, sono state aggiunte rule che vanno in trigger principalmente in seguito alla scoperta di nuove barche, più nello specifico delle PARTI di Barca.

Ricordo infatti che **k-cell** specifica nel contenuto a quale tipo di imbarcazione si riferisce.

RULE	LHS	RHS
guess-part-of-boat-top-given-left	- Rule mirata alle k-cell che identificano la parte «bot» di una barca - Ricerca una cella adiacente, non ancora «guessed» e con contenuto unknown da indicare come «altro pezzo di barca»	Asserita la EXEC su Guess Decrementato il counter sulla griglia rispetto alla posizione della cella che contiene «bot»
guess-part-of-boat-bottom-given-right		
guess-part-of-boat-down-given-middle		

Secondo Agente: Strategia e Loop

3. Fire e Guess: oltre alle stesse regole viste sul primo agente, sono state aggiunte rule che vanno in trigger principalmente in seguito alla scoperta di nuove barche, più nello specifico delle PARTI di Barca.

Ricordo infatti che **k-cell** specifica nel contenuto a quale tipo di imbarcazione si riferisce.

RULE	LHS	RHS
guess-part-of-boat-left-given-middle	….	Asserita la EXEC su Guess Decrementato il counter sulla griglia rispetto alla posizione della cella che contiene «bot»
guess-part-of-boat-right-given-middle	…	----
guess-part-of-boat-down-given-middle	….	…..

Secondo Agente: Strategia e Loop

3. Fire e Guess: oltre alle stesse regole viste sul primo agente, sono state aggiunte rule che vanno in trigger principalmente in seguito alla scoperta di nuove barche, più nello specifico delle PARTI di Barca.

Ricordo infatti che **k-cell** specifica nel contenuto a quale tipo di imbarcazione si riferisce.

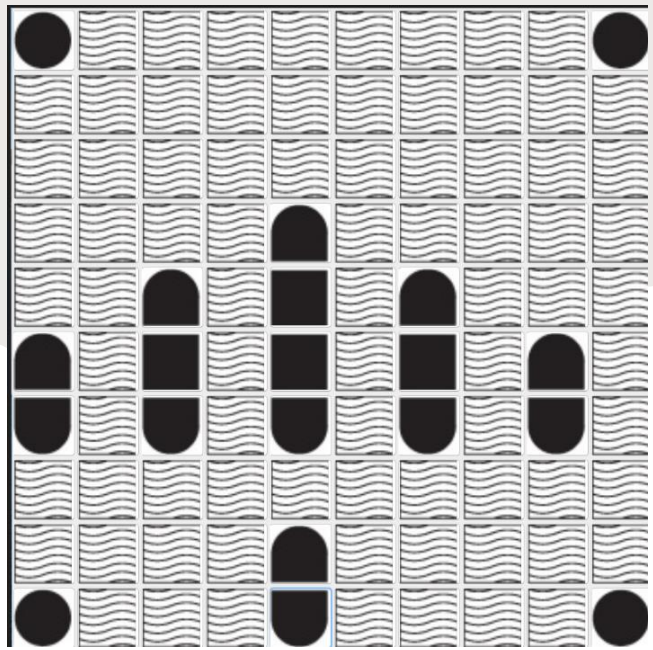
RULE	LHS	RHS
guess-part-of-boat-left-given-middle	….	Asserita la EXEC su Guess Decrementato il counter sulla griglia rispetto alla posizione della cella che contiene «bot»
guess-part-of-boat-right-given-middle	…	----
guess-part-of-boat-down-given-middle	….	…..

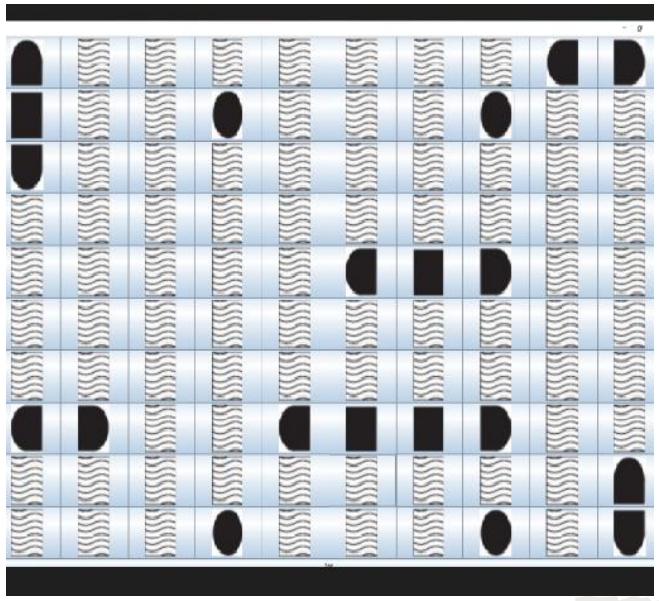
Primo Agente: Strategia e Loop

4. Condizione di uscita: quando non è più applicabile alcuna regola, è richiamata la **SOLVE**.

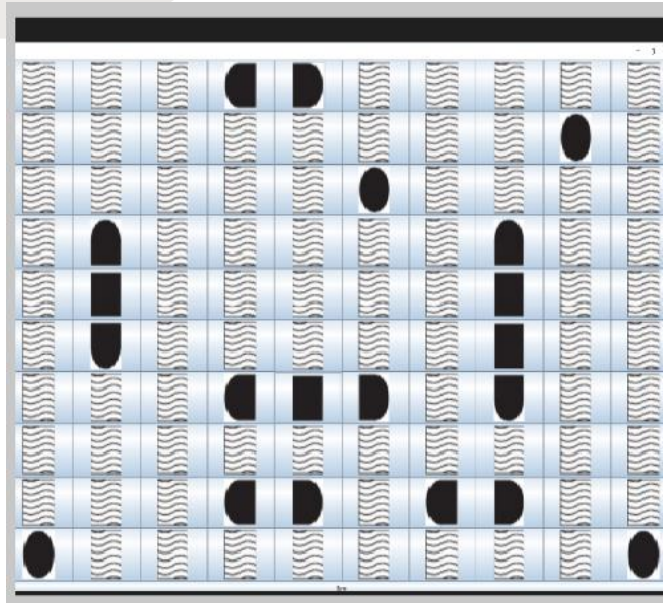
RULE	LHS	RHS
nothing-more-to-do	- …	Asserita la EXEC su Solve e termina l'esecuzione del tutto.

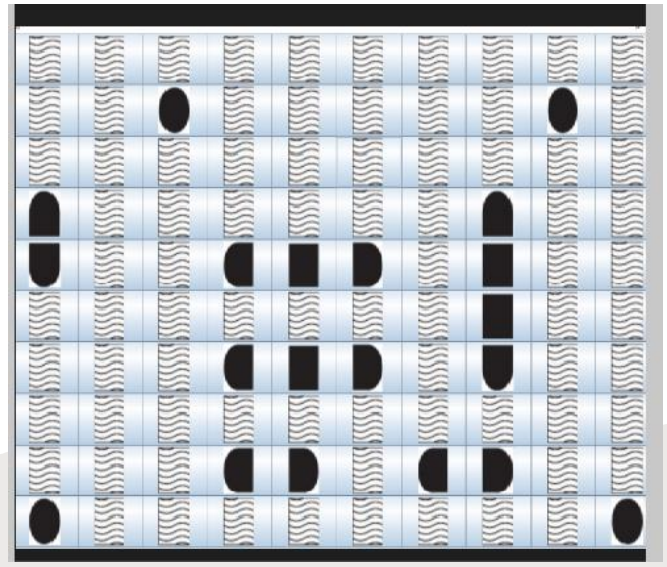
Risultati e test

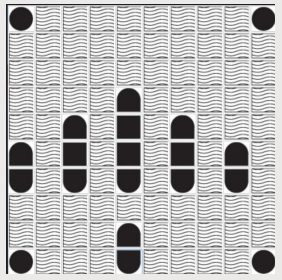




Verifichiamo i risultati degli esperimenti
di confronto dei due agenti e verifichiamo
quale impatto ha avuto aggiungere
regole legate al contesto nonché esperimenti
In cui sono state aumentate o ridotte le info
disponibili ad inizio partita.







Risultati e test

Mappa 0

	conoscenza iniziale	AGENTE 1	AGENTE 2
MAPPA 0	0	155	265
	2	140	250
	4	125	235
	8	95	205
	12	165	220

Il secondo agente sembra avere avuto performance generalmente migliori rispetto al primo. Il degrado in performance è invece da indagare.

Risultati e test

Mappa 0 – Agente 2

Il secondo agente sembra avere avuto performance generalmente migliori rispetto al primo. Il degrado in performance è invece da indagare.

Statistics Conoscenza Di base	num_fire_ok	num_fire_ko	num_guess_ok	num_guess_ko	num_safe	num_sink
0	5	0	13	7	28)	
12 celle	5	0	8	7	19)	

Ci sono stati 5 errori di troppo nella fase di GUESS quando i dati in input erano molti di più..

Risultati e test

Il resto dei test

MAPPA 3	0	70	30
	2	55	120
	4	40	120
	8	40	195
	12	55	180

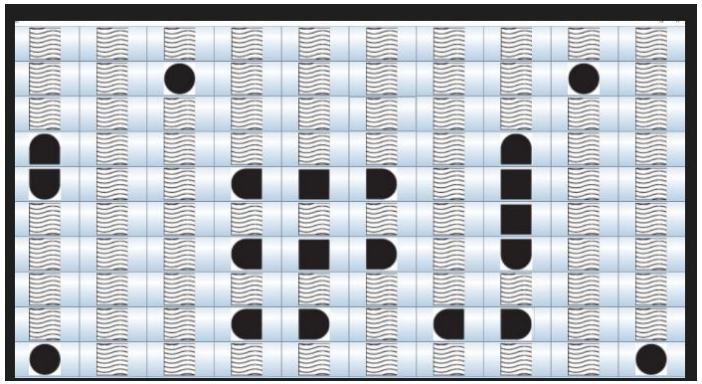
MAPPA 4	0	0	0
	2	0	85
	4	40	95
	8	25	135
	12	25	190

MAPPA 1	0	-70	45
	5	-60	110
	15	65	200

MAPPA 2	0	80	135
	5	105	-70
	15	180	-45

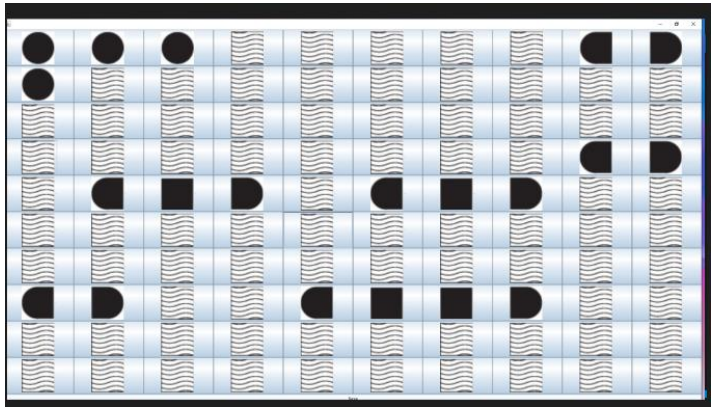
Risultati e test

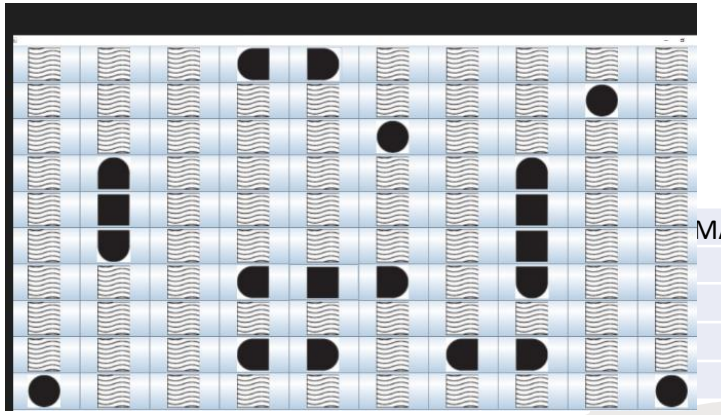
Il resto dei test



MAPPA 3	0	70	30
	2	55	120
	4	40	120
	8	40	195
	12	55	180

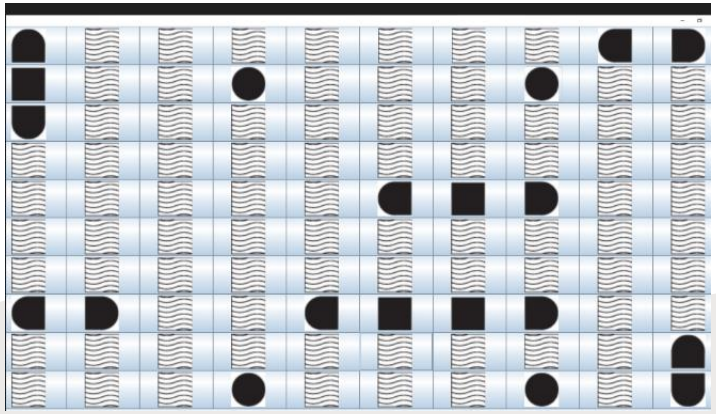
MAPPA 1	0	-70	45
	5	-60	110
	15	65	200





MAPPA 4	0	0	0
	2	0	85
	4	40	95
	8	25	135
	12	25	190

MAPPA 2	0	80	135
	5	105	-70
	15	180	-45



Risultati e test

Generalmente il secondo agente si comporta meglio del primo. Per quanto riguarda mappa 2 e mappa 3, in realtà ho commesso un errore durante l'editing e infattinon persiste il requisito di distanziamento di almeno 1 casella tra le varie celle. Ho voluto comuque riutilizzare la mappa per verificare come gli agenti si comportano anche se fuori dall'ambiente su cui è stato modellato il loro comportamento.

Possibili evolutive

- Si potrebbe introdurre i Certainty factors in quanto questi sono utili in molte situazioni in cui un agente deve prendere decisioni basate su dati incerti o parziali.
- potrebbero essere usati per indicare che appunto una barca sia o meno in una cella, e dunque la <probabilità> che lo sia
- quindi risulterebbero molto utili per «pesare» le diverse opzioni disponibili