“Colpito e affondato...?”

Scopo del gioco

L’obiettivo del progetto è quello di sviluppare un sistema esperto che giochi ad una versione semplificata della famigerata Battaglia Navale. Il gioco è nella versione “in-solitario”, per cui c’è un solo giocatore che deve indovinare la posizione di una flotta di navi distribuite su una griglia 10x10. L’obiettivo del gioco è di annotare come “guessed” le venti caselle sotto cui si trovano le navi.

Griglia di gioco

La griglia di gioco ospita, come di consueto, 10 navi, posizionate in verticale o in orizzontale, individuate fra i seguenti tipi:

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

La griglia dispone sulle righe e sulle colonne il numero di celle che contengono pezzi di nave. L’indicazione è disponibile al giocatore.

Vincoli di gioco

- deve esserci almeno una cella libera tra due navi.
- il giocatore può eseguire 4 tipi di mosse: fire, guess, unguess, solve
- il giocatore ha un numero limitato di mosse: 5 fire, 20 guess, 1 solve
- in presenza di celle contenenti pezzi di nave note sin dall’inizio, il numero di guess va diminuito per il numero di tali celle.
- Al termine del gioco, l’agente deve aver usato tutte le 20 guess.

Varianti

Sono richieste almeno due varianti di giocatore che seguano due strategie diverse, da mettere poi a confronto su mappe con scenari di gioco differenti.

Introduzione

Il nostro progetto “colpito o affondato” ha visto la definizione di 3 sistemi esperti che, sfruttando strategie ed inferenze differenti, simulano il comportamento di un giocatore umano provando così a giocare e a vincere al gioco della battaglia navale in solitario.
I sistemi esperti che abbiamo modellato, di seguito indicati come “Agente1, Agente2, Agente3, seguono un set di inferenze di base che, lungo le diverse versioni, va via via crescendo producendo come risultato finale score differenti. Infatti, a partire dalle mancanze e dagli errori dell’Agente1, i successivi provano a raffinare le osservazioni, delineando così 2 caratteri emergenti:

- ❖ Agente impulsivo
- ❖ Agente riflessivo

La distinzione verte sia sul modo in cui i vari agenti sfruttano le informazioni iniziali, ricavabili dal terreno di gioco, sia per quanto riguarda il modo con cui essi sfruttano il numero limitato di mosse messe a disposizione.

Modellazione della conoscenza

Template agent_cell

Tutti i sistemi esperti che abbiamo modellato presentano un fatto non ordinato in comune: **"agent cell"**, la cui utilità è quella di modellare la mappa di gioco dal punto di vista dell'agente. L'agente riconosce una cella del terreno di gioco dalle sue coordinate “x, y”, da uno stato e da un certainty factor: "**guess_CF**". Questa informazione riassume la credenza dell'agente circa la presenza o meno di un pezzo di nave in una determinata cella. Maggiore è il valore di *guess_CF*, più alta sarà la sua convinzione sulla presenza di un pezzo di nave. I valori di *guess_CF* variano in un intervallo da 0 a 100. Lo stato di ogni *agent_cell* può essere di 3 tipi:

- **none**: se non è stata ne osservata con una fire, ne con una guess
- **guessed**: se è stata lanciata una guess
- **fired**: se è stata osservata con una fire

Lo stato viene usato per sapere quali celle sono ancora sconosciute all’agente, in modo da distinguerle rispetto a quelle che sono state osservate con una fire o segnate da una guess. Così facendo, nel momento in cui bisogna eseguire una certa azione, come ad esempio una guess, l’agente non va a segnare nuovamente una casella già osservata in precedenza, a meno di fare unguess sotto certe condizioni.

Template agent_status

Un secondo template **"agent_status"**, definito esclusivamente per Agente2 e Agente3 modella lo stato corrente del sistema esperto. In presenza di un'organizzazione in moduli e sotto-moduli, come appunto avviene per questi 2 agenti, il template *agent_status* permette di alternare l'esecuzione dei vari moduli tenendo traccia del modulo corrente e delle operazioni eseguibili.

Template boat_number_type

Questo template serve per rappresentare il numero di navi che l'agente conosce inizialmente. Solo l'agente3 ha a disposizione questo template. In combinazione con il template "boat_position" serve per tenere traccia di quali navi l'agente crede di aver individuato e colpito.

Template boats_position

Questo template definisce le posizioni delle navi che l'agente prova ad inferire durante il gioco. Viene definito solo per l'agente3. Durante il gioco, questo sistema esperto lanciando alcune fire, o in base a certe evidenze, come ad esempio la presenza di pezzi di nave vicino ai bordi del terreno di gioco, prova a dedurre le posizioni e l'orientamento delle varie navi da eliminare. Queste informazioni rappresentano solo un'evidenza in più con cui l'agente tenta di risolvere il gioco.

```
(deftemplate agent_cell
    (slot x)
    (slot y)
    (slot status (allowed-values none guessed fired) (default none))
    (slot guess_CF (default -1))
)

(deftemplate agent_status
    (slot currently (allowed-values main plan execute finish))
)
(deftemplate boat_number_type
    (slot type (allowed-values boat_1 boat_2 boat_3 boat_4))
    (slot number)
)

(deftemplate boats_position
    (multislot xs)
    (multislot ys)
    (slot type (allowed-values boat_1 boat_2 boat_3 boat_4))
    (slot confident (allowed-values YES NO) (default NO))
)
```

Organizzazione della conoscenza

Come accennato in precedenza, i 3 sistemi esperti rispondono a 3 organizzazioni di conoscenza differenti.

Agente1. Partendo dal primo, visibile nel file *agent_version1/3_Agent.clp*, l'Agente1 importa dai moduli **MAIN** e **ENV** alcuni fatti fra cui: *status, statistics, k-cell, k-per-row, k-per-col.* Inoltre, dichiara come fatti iniziali la rappresentazione **agent_cell** della griglia di gioco (visibile nel file **4_Agent_cells.clp**) e un fatto "**init**" il cui scopo è semplicemente quello di far eseguire alcune regole di controllo solo all'atto dell'inizializzazione. Riguardo a quest'ultimo aspetto, l'Agente1 non ha una buona separazione delle regole di expertise da quelle di controllo, infatti tutta la conoscenza dell'agente è immagazzinata nello stesso file (*3_Agent.clp)*. Come si può vedere dalla regola *update_guess_CF* (riga 61), il cui scopo è quello di aggiornare la conoscenza iniziale dell'agente rispetto alle evidenze date dal certainty factor "guess_CF", la conoscenza dell'agente viene mescolata con fatti di controllo, così come accade anche per altre regole.

Agente2. Il secondo agente, i cui file sono visibili della directory *agent_version_2*, ha un'organizzazione della conoscenza molto differente rispetto al precedente. In questa versione cominciano a delinearsi alcuni dei consueti moduli di rappresentazione di un sistema artificiale che identificano alcune delle macro azioni che esso compie in un certo ambiente. Sono stati definiti i moduli: *AGENT_VIEW_PLAN* e *AGENT_EXECUTE*. Assieme al modulo *AGENT* i due precedenti separano logicamente le azioni di **percezione, pianificazione** e **deliberazione** dell'agente.

- Il modulo AGENT ospita le definizioni dei template *agent_cell* e *agent_status*, i fatti iniziali init e agent_status, alcune regole di controllo per stampare a video la conoscenza iniziale, e le regole che modellano l'alternarsi delle azioni tra i vari moduli. Infatti, come per il MAIN il modulo AGENT si comporta come un intermediario, alternando azioni di pianificazione e visione a quelle di deliberazione. Terminata la fase di inizializzazione, il controllo passa al modulo di pianificazione.
- il modulo AGENT_VIEW_PLAN mescola le azioni di osservazione e quelle di pianificazione. Dopo aver osservato la griglia di gioco e aver tratto tutte le informazioni possibili a seguito di una certa azione precedentemente eseguita, l'agente pianifica l'azione migliore da compiere. In seguito passa il controllo all'ultimo modulo.
- Il modulo EXECUTE non fa altro che eseguire l'azione pianificata. Eseguita l'azione, il controllo passa nuovamente al modulo AGENT.

Agente3. L'ultimo agente ha una struttura e un comportamento leggermente diverso rispetto al precedente. In questi è visibile una più netta separazione tra le regole che gestiscono il flusso di controllo e quelle che manipolano la conoscenza dell'agente. Sono presenti 4 moduli: *AGENT, AGENT_OBSERVE, AGENT_PLAN* e *AGENT_EXECUTE*.

- **AGENT** si comporta solo come intermediario dei vari moduli. Dichiara il template agent_status e imposta l'esecuzione del modulo AGENT_OBSERVE. Non sono presenti regole di expertise.
- **AGENT_OBSERVE** definisce tutte le regole che consentono di ricavare delle informazioni dal terreno di gioco. Inoltre dichiara i template *agent_cell*, *boat_number_type* e *boat_position* e inizializza la griglia di gioco.
- **AGENT_PLAN** sulla base delle informazioni ricavate dal modulo precedente, sceglie l'azione da compiere che ritiene essere la più adeguata in un dato istante, quindi passa il controllo al modulo EXECUTE.
- **EXECUTE**, come per l'Agente2, non fa altro che eseguire l'azione pianificata del modulo PLAN. Eseguita l'azione, il controllo passa nuovamente al modulo AGENT.

Regole di expertise dei vari agenti

In questa tabella sono visibili le regole di expertise dei vari agenti che abbiamo ritenuto essere le più importanti e che delineano il comportamento del tipo di sistema esperto.

Regole di expertise				
Nome regole	Descrizione	Presente in		
		Agent1	Agent2	Agent3
update_guess_CF	Inizializza i valori di guess_CF sulle celle ignote	SI	SI	SI
update_guess_CF_on_k_cell	Inizializza i valori di guess_CF sulle celle ignote	SI	SI	SI
guess_left, (right, up bottom)	Dispone una guess sul lato opposto se riconosce di un k-cell a sinistra, destra, in alto o in basso.	SI	NO	NO
update_guess_CF_if_k_cell_left (right, up, bot)	Aggiorna i valori di guess CF sulle celle vicino ad un k-cell left, right, top, bot.	NO	SI	SI
update_guess_CF_if_k_cell_middle_border_X (Y)	Aggiorna i valori di guess CF sulle celle lungo l'asse X o l'asse Y a partire dalla cella in corrispondenza di un pezzo middle.	NO	SI	SI
update_guess_CF_if_water_middle_hor (ver)	Inferisce l'orientamento di una nave da 3 o da 4 a partire da un middle e da celle water nelle 4 direzioni del middle.	NO	SI	SI
update_guess_CF_if_k_cell_middle_hor (col)	Aumenta le evidenze nell'orientamento di una nave da 3 o da 4 a partire dai numeri k-per-col e k-per-row	NO	SI	SI
update_guess_CF_if_k_cell_middle	Aggiorna i valori di gluess_CF nelle celle a nord, sud, est, ovest rispetto ad una cella middle.	SI	SI	SI

update_position_near_boats_left (sub, right, top, bottom, middle)	Aggiorna le evidenze guess_CF attorno ad un blocco left, sub, right, top, bot, middle. Inferisce la posizione dei blocchi water.	NO	SI	SI
update_col_row	Modifica i numeri di k-per-row e k-per-col dato un k-cell non water. Ricalcola i guess_CF delle celle lungo la riga e la colonna rispetto al k-cell	NO	SI	SI
discover_boat_sub	Aggiorna la conoscenza rispetto al numero di sottomarini individuati.	NO	NO	SI
discover_boat_2_left (right, up, down)	Aggiorna la conoscenza rispetto al numero di cacciatorpedinieri individuati.	NO	NO	SI
discover_boat_3_if_left_right (right_left, up_down, down_up)	Aggiorna la conoscenza rispetto al numero di incrociatori individuati.	NO	NO	SI
discover_boat_4_ver_type1(ver_type2, hor_type1-2)	Aggiorna il numero di corazzate individuate.	NO	NO	SI
find_cell_to_guess	Individua la cella con valore guess_CF più alto a cui fare guess	NO	SI	SI
find_cell_to_fire	Individua la cella con valore guess_CF più alto a cui fare fire	NO	SI	SI
unguess_on_certain_cells	Individua la cella a cui fare unguess	NO	NO	SI
find_cell_to_fire_if_agent_reconsider_move	Individua la cella guessed con valore guess_CF più alto a cui fare fire per riconsiderare alcune mosse.	NO	NO	SI
do_unguess	Esegue un’azione di unguess	NO	NO	SI
do_guess	Esegue un’azione di guess	SI	SI	SI
do_fire	Esegue un’azione di fire	NO	SI	SI
do_solve	Termina il gioco ed esegue un’azione di solve	Si	SI	SI

Logica e funzionamento degli Agenti

In questa sezione, in breve descriviamo la logica di funzionamento dei vari agenti, nonché il comportamento dell'esperto umano che cerchiamo di emulare. Riguardo a quest'ultimo aspetto, come accennato in precedenza, sono 2 i comportamenti umani che vengono modellati: giocatore impulsivo e giocatore riflessivo.

Agente1 rispecchia il comportamento di un **giocatore impulsivo e sicuro di sé**. Infatti, dopo aver sfruttato eventuali informazioni iniziali, come la presenza di certi pezzi di nave in certe posizioni, questo sistema pianifica, di getto, l'intera sequenza di azioni guess da disporre sulle varie celle, una dopo l'altra. La sua sicurezza è data dal certainty factor guess_CF che viene calcolato per ogni cella alla partenza del gioco. L'agente lancia le guess premiando le celle con guess_CF più alto e penalizzando le rimanenti altre. In presenza di pezzi "middle", questo agente non prova a capire l'orientamento della nave, quindi dispone 4 guess in corrispondenza delle celle accanto nelle 4 direzioni. Da notare che questo agente **non si avvale delle fire**. A differenza degli altri agenti che usando le fire si pongono in un'ottica di continual planning, questo agente pianifica tutto a monte senza mantenere una costante interazione con l'ambiente circostante.

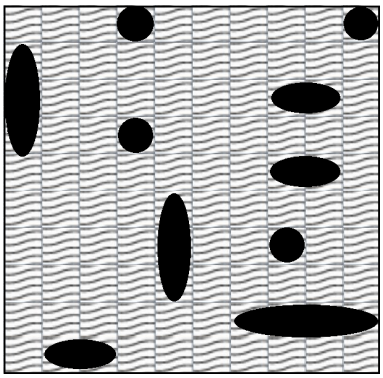
Agente2 non è altro che un **raffinamento** della **strategia semplicistica** adottata dall'Agente1. L'enorme differenza con il precedente sta sia nell'uso delle fire, sia nel modo con cui sfrutta le informazioni della griglia di gioco. Prima di disporre tutte le guess, questo agente esegue le fire, una dopo l'altra e verificando di volta in volta i cambiamenti riportati dall'ambiente. inoltre, ogni qualvolta viene a sapere di un k-cell che non sia water, questo agente ricalcola le evidenze guess_cf su tutte le celle della riga e della colonna in cui si trova quel pezzetto di nave, diminuendo i numeri k-per-row e k-per-col. Inoltre, raffina il comportamento in presenza di un pezzo "middle". Se la cella che ospita quel middle ha un numero k-per-row maggiore di un k-per-col, allora dispone le guess sui 2 lati orizzontali, viceversa su quelli verticali. A parità di valore si usa il criterio dell'agente1, quindi dispone 4 guess nelle 4 direzioni. Infine, questo agente non ripensa mai alle azioni compiute.

Agente3 modella il comportamento di un **giocatore più riflessivo.** Questo è dato dal fatto che in presenza di un "middle", a meno di ricavare per inferenza la posizione e l'orientamento della nave semplicemente perché il pezzo si trova sui bordi della scacchiera, l'agente prova a indovinare la disposizione della nave ripensando alle scelte compiute. Infatti dalle osservazioni k-per-cell e k-per-row ricava inizialmente una certa posizione e dispone le guess seguendo lo stesso criterio di *Agente2*. In seguito, essendo diffidente e riflessivo, se ha *fire* a disposizione ne lancia una in corrispondenza di una delle guess che ha disposto. Se riconosce un pezzo water, allora fa **unguess** sulle precedenti guess e aggiusta il tiro disponendole nel verso opposto. Infine, anche questo agente tende a lanciare prima tutte le fire e successivamente fare guess sulle celle restanti. La differenza è che ogni volta che fa fire

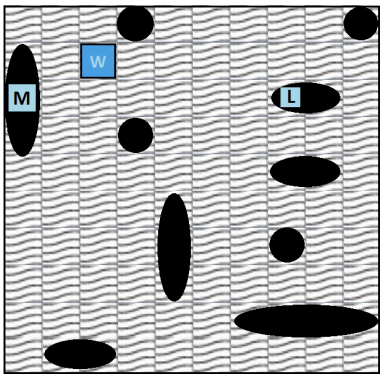
aggiorna le informazioni sulle evidenze guess_CF e fa immediatamente guess laddove ritiene che per certo si trovi un pezzo di nave.

Risultati

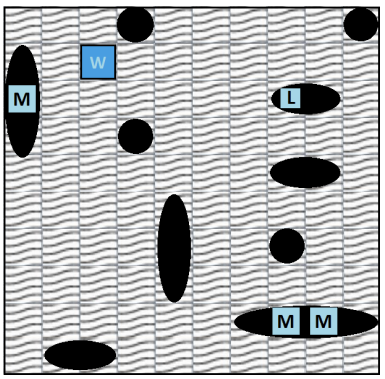
In questa sezione mostriamo i vari test che abbiamo condotto per confrontare il comportamento e le performance dei vari sistemi esperti che abbiamo modellato. Tutti i test si riferiscono alla stessa mappa di gioco che progressivamente da completamente nascosta viene rigiocata avendo in input alcune informazioni parziali. Nello specifico sono state definite **3 varianti** di mappe gioco. Le lettere “M” e “L” indicano la presenza rispettivamente di pezzi “middle” e “left”. Nel seguito, in forma tabellare riportiamo i risultati dei vari test.



Mappa 1 - completamente nascosta



Mappa 2 - 3 celle visibili



Mappa 3 - 5 celle visibili

Risultati Mappa 1 - Completamente nascosta				
Agente	**N° Fire OK**	**N° Guess OK**	**N° Boat Destroyed**	**Score**
Agente 1	0	8	2	-180
Agente 2	2	9	3	-5
Agente 3	2	10	5	80

Risultati Mappa 2 - 3 celle note				
Agente	**N° Fire OK**	**N° Guess OK**	**N° Boat Destroyed**	**Score**
Agente 1	0	6	2	-230
Agente 2	1	10	5	80
Agente 3	2	12	7	210

Risultati Mappa 3 - 5 celle note				
Agente	**N° Fire OK**	**N° Guess OK**	**N° Boat Destroyed**	**Score**
Agente 1	0	5	3	-225
Agente 2	2	9	7	115
Agente 3	3	13	10	325

Conclusioni

In tutti i test l'Agente1 risulta quello meno performante. Si può notare come l'osservabilità iniziale per questo agente, **incide negativamente** e sembra non avere distinzioni quando essa viene variata aumentando o diminuendo il numero di pezzi di nave noti a priori. Ragionamento opposto per quanto riguarda i successivi 2 Agenti, che invece in presenza di informazioni iniziali, progressivamente all'aumentare del numero di pezzi noti lo score finale subisce una variazione in positivo.
La presenza **di 2 pezzi middle** noti a priori e consecutivi non è stata una scelta a caso. La nave è chiaramente di dimensione 4 ed è disposta in orizzontale, ma mentre l'Agente2 sembra non aver chiaro come sfruttare questa informazione, l'Agente3 riconosce il tipo di nave. Infatti, dall'ultima tabella si può notare il salto di performance tra l'Agente2 e l'Agente3, garantendo per quest'ultimo la possibilità di applicare il set completo di regole di inferenze che ha a disposizione.

Bontà

Complessivamente i sistemi esperti da noi modellati rispondono a criteri e strategie di risoluzione del gioco che un giocatore umano potrebbe adottare. Gli errori commessi dai vari agenti diminuiscono in proporzione a quanto essi sono in grado di sfruttare le informazioni provenienti dalla griglia di gioco.

Limiti

Uno dei limiti della nostra implementazione è stato quello di calibrare le scelte dei giocatori solo sulla base del certainty factor guess_CF. Ciò comporta un leggero distacco dalla modellazione del comportamento umano e una migrazione, con l'agente3, verso qualcosa di più algoritmico. Una possibile soluzione è quella di combinare questo fattore con un secondo che modelli lo stato mentale dell'agente, ovvero la sua insicurezza e la sua tendenza a commettere più errori in presenza di condizioni che rendono meno divertente giocare.