Note sul sistema

Battaglia navale tra Agenti software in ambiente n-dimensionale

Nicola Zaghini

1 Idea del sistema

Si intende realizzare un sistema software abilitante al gioco della battaglia navale per giocatori umani e virtuali. Si intende realizzare anche giocatori virtuali che possano partecipare al gioco battaglia navale. Questi giocatori virtuali verranno realizzati con differenti capacitá di gioco, intendibile come differente intelligenza per verificare come un agente che incorpora tecniche migliori, ed in numero maggiore, di gioco in generale é avvantaggiato nella vittoria finale. I giocatori virtuali sarano realizzati in maniera incrementale, dal piú semplice e quindi meno abile al gioco, al piú complicato e qundi in grado di giocare alla pari con un giocatore umano.

2 Analisi del gioco Battaglia navale

La battaglia navale é un gioco di societá a cui possono partecipare da un minimo di 2 di due giocatori ad un massimo non limitato. Ogni giocatore é in possesso di un tabellone n-dimensionale in cui posizionare le proprie navi. Il tabellone rappresenta un ipercubo n-dimensionale (es: con due dimensioni si ha il classico tabellone da gioco, mentre con 3 dimensioni il tabellone diventa un cubo, con 4 un tesseratto e cos via) in cui ogni punto di interesse, che chiameremo *casella*, é identificato dalle coordinate (x1,x2,x3, ..., xn) con xi $\in$ N+. Ogni giocatore decide come posizionare le proprie navi, in maniera riservata rispetto agli altri concorrenti, in accordo alle seguenti regole:

- le navi non possono essere sovrapposte nemmeno parzialmente;
- le navi non possono essere disposte affiancate in senso verticale od orizzontale.

Finita la disposizione delle navi da parte di tutti i concorrenti il gioco inizia. Lo *scopo* di ogni giocatore é quello di colpire le navi avversarie, vince

l'ultimo giocatore che rimane con almeno una barca, o pezzo di barca, sullo scenario. Non é ovviamente possibile pensare che un nuovo giocatore possa entrare in una partita iniziata in quanto sarebbe avvantaggiato rispetto agli altri.

2.1 Partecipare al gioco

Ogni giocatore decide liberamente se partecipare al gioco, basandosi sulle informazioni legate ad esso come ad esempio le dimensioni e la dimensione del tabellone di gioco ed il numero di navi da disporre. Ottenute queste informazioni per partecipare é necessario che ci sia disponibilitá di posti nel gioco per potervi partecipare.

2.2 Tecnica di gioco

Il gioco inizia scegliendo tra i partecipanti il primo a muovere, ad esempio facendo lanciare un dado ad ogni partecipante considerando vincitore chi ottiene la faccia con il valore piú alto o piú basso. Chi lancia l'attacco deve definire una coordinata n-dimensionale corrispondente alla zona del tabellone dell'avversario scelto in cui ritiene possa essere situata una nave della flotta. Il giocatore target dell'attacco é scelto dall'attaccante stesso in maniera non fissata a priori. Passato il turno il successivo giocatore si sceglie secondo una politica che puó essere determinata dalla posizione dello stesso (destra o sinistra ipotizzando i concorrenti seduti ad un tavolo), oppure basandosi sul valore ottenuto dal lancio del dado, ad esempio dal valore maggiore a quello minore. Nel caso in cui una nave sia colpita il giocatore attaccato deve comunicare quanto accaduto esplicitando se la nave é stata completamente affondata oppure no. Infatti una nave per essere affondata deve essere colpita nella sua totalitá, ad esempio una nave lunga n caselle avrá bisogno di n colpi per essere affondata (ovviamente gli n colpi devono coprire esattamente la superficie occupata dalla nave).

2.3 Intelligenza e risvolti sociali

Come in ogni gioco il giocatore che possiede una maggiore **intelligenza**, estrinsecata in tecniche di gioco che portano a mosse non casuali ed in generale vincenti, é solitamente avvantaggiato nello scopo di vincere la partita.

Considerando come strategia di gioco la *truffa*, é possibile che due giocatori si *coalizzino* segretamente in modo da non danneggiarsi vicendevolmente oppure con lo scopo di sconfiggere un nemico ritenuto particolarmente abile. Partendo da un esempio di comportamento sleale si puó arrivare a definire un nuovo tipo di comportamento nel gioco: il *gioco a squadre*, dove il vincente non é piú un giocatore ma una squadra in cui ogni giocatore mantiene la sua indipendenza avendo peró come scopo la vittoria della propria fazione. Viene quindi chiesta ai giocatori una capacitá nuova che si puó definire come **social ability**, ovvero l'abilitá a perseguire uno scopo (goal) comune.

3 Progetto del sistema

Il sistema é costituito fondamentalmente da un insieme di giocatori virtuali ed un artefatto di coordinazione al quale tutti afferiscono come mostrato nella figura 1.

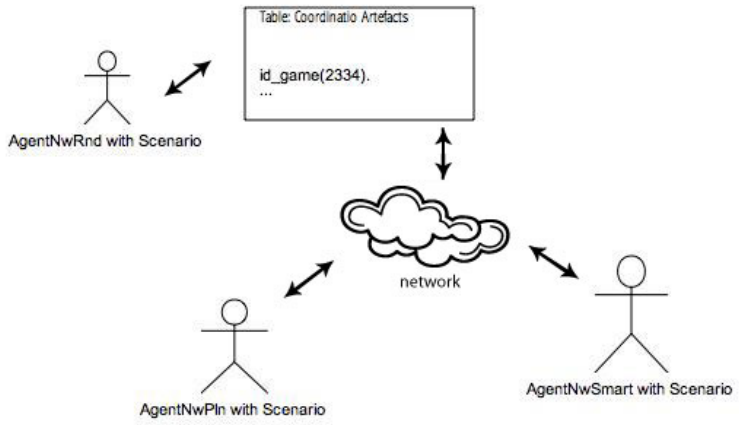

Figura 1: Architettura logica

Il centro del sistema rappresenta lo strumento di coordinazione ed abilitazione al gioco, infatti ogni giocatore deve interagire con questo per potere iscriversi ad un gioco e successivamente giocarvi. Come si nota il sistema é distribuito sull'infrastruttura abilitante internet, caratteristica ottenuta grazie all'utilizzo di TuCSoN.

3.1 Artefatto di coordinazione

L'artefatto di coordinazione fornisce, per antonomasia, i concetti di coordinazione necessari per potere regolare un gioco in cui i giocatori non possono sedersi ad un tavolo reale e quindi interagire con parole e visone diretta. Si vuole abilitare la presenza simultanea di differenti giochi su uno stesso tavolo, distinguendoli da un identificatore univoco gestito dal tavolo stesso. Il tavolo virtuale di cui si necessitá per lo svolgimento del gioco deve:

- rendere disponibili le informazioni necessarie per lo svolgimento del gioco quali:
  - identificatore univoco del gioco;
  - dimensioni del tabellone ed ampiezza;
  - numero e dimensione delle navi da disporre;
  - numero di giocatori massimo.
- fornire un meccanismo per decidere quando é possibile iniziare il gioco, ovvero nessun altro giocatore deve arrivare;
- fornire un meccanismo di scelta del giocatore che inizierá la partita;
- fornire un meccanismo di scelta del prossimo giocatore che puó muovere;
- tenere traccia dei giocatori che stanno partecipando, ovvero che non hanno abbandonato la partita oppure non hanno piú navi;
- consentire molteplici partite sullo stesso centro di coordinazione anche contemporanee;
- ripulire lo scenario a fine gioco.

Considerato il fatto che le informazioni necessarie sono facilmente esprimibili in maniera simbolica risulta comoda la manipolazione dei predicati logici con il linguaggio logico *Prolog* e si decide di utilizzare *TuCSoN* come artefatto di coordinazione programmabile in quanto ritenuto sufficientemente espressivo per rappresentare il tavolo da gioco virtuale ed in quanto abilitante l'utilizzo di componenti software, distribuiti nel network, chiamati *Agenti*

quali giocatori virtuali.
Il centro di coordinazione unito alla specifica di comportamento rappresenta l' *infrastruttura abilitante* al gioco. Si realizza una specifica in linguaggio Respect che chi intende lanciare il gioco deve imporre al centro di tuple.

3.2 Giocatori Virtuali

Adottando un approccio top-down si sceglie come componente software rappresentante dei giocatori virtuali l' *Agente*. La definizione di agente é infatti calzante con il ruolo di giocatore (di un qualsiasi gioco) virtuale, riportandone una [1]: un agente é un sistema informatico capace di azioni autonome, all'interno dell'ambiente di riferimento, atte al perseguire il proprio scopo/i. Le proprietá che si possono associare ad un Agente eletto giocatore virtuale sono le seguenti:

1. pro-activeness vs reactivenes: un agente pro-attivo é in grado di seguire il proprio goal anche quando non é il suo turno, guardando e studiando le mosse degli avversari; un agente puramente reattivo é in grado di deliberare solo nel momento in cui é il suo turno nel gioco;

2. social ability: come introdotto in analisi Agenti potrebbero coalizzarsi a squadre avendo come goal la vittoria della squadra unitamente alla propria sopravvivenza;

3. veracity: un agente, come un giocatore umano, ha un livello di incapsulazione tale da non consentirne una ispezione interna da parte di altri agenti, é quindi necessario che non dichiari il falso durante lo svolgimento del gioco;

4. rationality: un agente fa mosse coerenti con il suo goal;

5. learninig: un agente studia le caratteristiche degli avversari per potere deliberare con maggiore informazioni.

Non tutte queste peculiaritá sono necessarie perché un agente sia definito tale e possa partecipare al gioco, potremmo dire che l'*intelligenza* dell'agente é proporzionale al numero di caratteristiche che esso incorpora positivamente.

3.3 Architettura

Si sceglie di realizzare gli agenti utilizzando la combinazione di Java e Prolog in quanto il primo linguaggio é preferibile per tassonomizzare gli agenti e per le procedure in cui é richiesta una spinta capacitá computazionale, mentre il secondo risulta migliore e maggiormente immediato per la descrizione della logica di gioco. Come implementazione del linguaggio logico Prolog si decide di utilizzare *tuProlog* per la sua ottima interazione ed integrazione con Java. L'architettura messa in gioco per la realizzazione degli agenti mostrata nella figura 2.

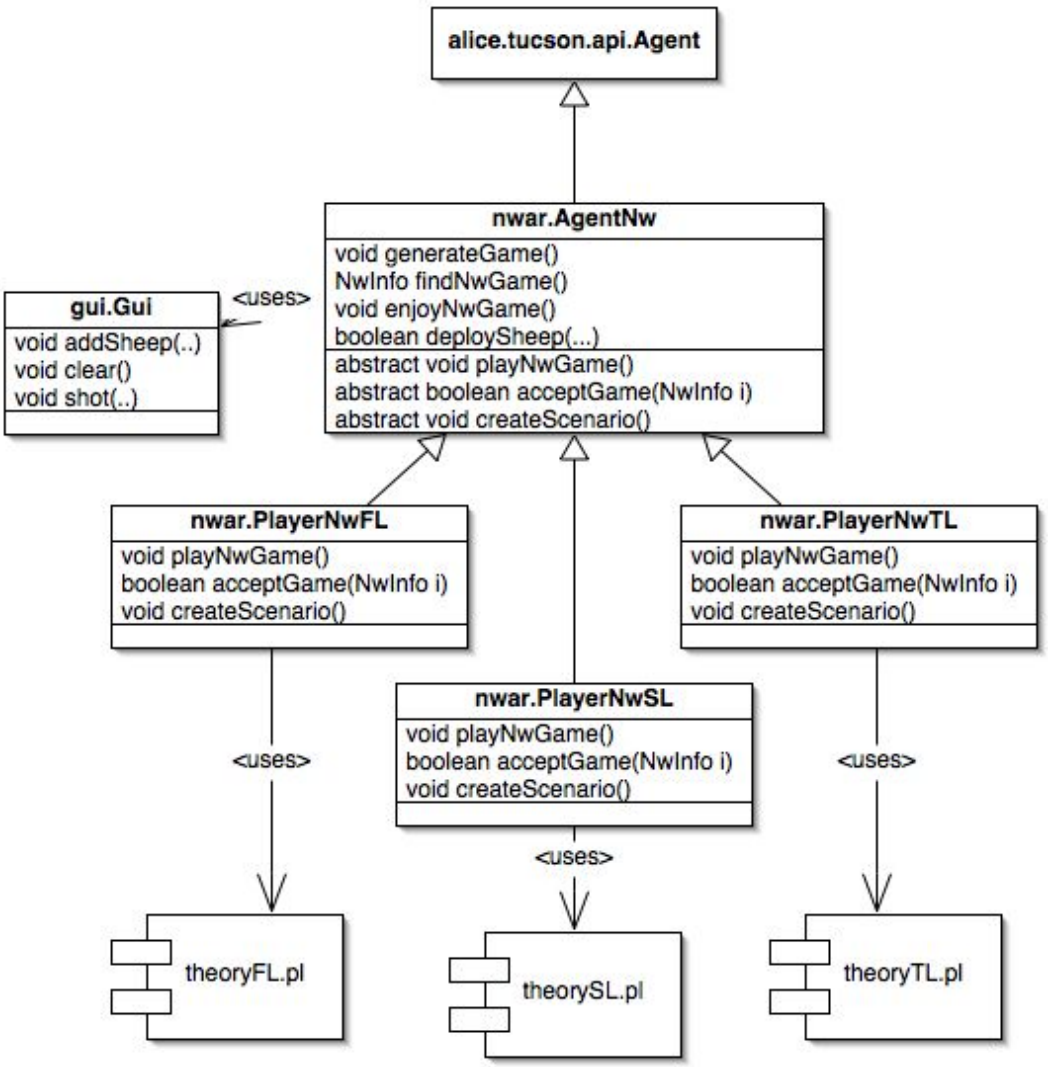

Figura 2: Architettura formale

La classe astratta AgentNw estende la classe alice.tucson.api.Agent e

mette a disposizione i metodi per:

- creare un nuovo gioco;
- trovare un gioco valido all'interno di un nodo **TuCSoN** ;
- partecipare al gioco trovato;
- disporre le navi in maniera coerente partendo da un punto dato.

Di rilevante importanza risulta il metodo che consente di disporre le navi, infatti questo metodo costruisce una teoria rappresentante le navi disposte nel tabellone privato dell'agente verificando che la nave che si intende disporre sia sempre distanziata di almeno una casella dalle altre gia disposte. In oltre evita i casi di dead-lock per impossibilitá di disposizione delle navi, facilmente verificabili con tabelloni piccoli in relazioni alle navi che vi devono essere disposte, ripartendo nella disposizione delle navi dopo un numero di tentativi falliti.
Il metodo messo a disposizione per la creazione di un nuovo gioco setta la specifica Respect che abilita alla partita in un nodo ed un centro dato ed emette una tupla, che verrá successivamente descritta, in grado di fare iniziare il gioco.

I metodi astratti sono definiti tali in quanto per ogni agente, a senconda delle capacitá che deve incorporare, si devono scegliere politiche per:

- partecipare al gioco trovato discriminando sulle informazioni legate allo stesso;
- creare lo scenario di navi;
- giocare, ovvero la strategia di gioco.

Le classi che estendono la classe AgentNw implementano in maniera differente questi metodi astratti, variano tra loro in relazione a quello che riguarda l'attivitá deliberativa e quella di means-end. In realtá tutte le implementazioni utilizzano una stessa identica tecnica di means-end che consiste nell'inserire tuple di una certa forma all'interno del centro di tuple, quindi la diffrenza sostanziale é data dalla fase di deliberazione in relazione alla fase piú importante del gioco ovvero l'attacco. La caratteristica che li accomuna

é che tutti gli agenti progettati sono di tipo reattivo, ovvero elaborano il loro piano solo nel momento in cui sono chiamati a farlo, ovvero quando sono attaccati o possono attaccare. La strategia di gioco é implementata come teoria logica Prolog caricata dinamicamente all'interno del metodo

```
abstract void playNwGame()
```

di conseguenza la differenza nel gioco tra agente e agente si trova principalmente nelle teorie caricate ed in particolare nella fase di attacco di queste; la fase di risposta ad un attacco simile per tutti infatti ci si deve limitare a dire l'effetto del colpo.

AgentNwFL La classe AgentNwFL rappresenta gli agenti di primo livello (First Level) intendendo come livello la capacitá deliberativa per quello che riguarda l'attacco. In questo agente si puó considerare ridotta ai minimi termini questa attivitá infatti l'attacco viene eseguito in maniera random, senza nemmeno verificare che una coordinata sia stata colpita oppure no in precedenza. Il piano di azione é quindi molto semplice e limitato: giungere al goal (abbattere tutte le navi avversarie) sparando in maniera casuale ad un avversario scelto anch'esso casualmente. Delibera peró inizialmente sulla sua possibilitá di vittoria nel gioco accettando di partecipare a giochi con un numero di dimnensioni e di dimensione del tabolellone limitato. In questo senso l'agente crede di non potere vincere in scenari troppo ampi. Il fatto che l'agente creda ció non puó essere altro che legato al fatto che ha una conoscenza innata dello scenario dei possibili giocatori e che lui non é abile al gioco quanto altri; a livello di sitema intenzionale puó essere descritto come segue: 'se uno scenario di gioco é molto grande, quindi molte caselle, é inutile che vi partecipi sapedo che potrei confrontarmi con giocatori tanto piú forti di me da non lasciarmi scampo'.

AgentNwSL La classe AgentNwSL rappresenta gli agenti di secondo livello (Second Level). La capacitá deliberativa in attacco di questo agente é migliore rispetto al precedente, infatti prima di sparare (pur sempre in maniera random) verifica che la coordinata di quell'agente non sia gia stata colpita da lui stesso. Il piano di azione gli consente di giungere al goal in un numero di mosse inferiore, statisticamente parlando, rispetto all'agente di primo livello sopra descritto. Questo agente delibera inizialmente sulla

possibilitá di vittoria basandosi sul numero di caselle complessive, ed accetta giochi 'piccoli' pur sempre piú grandi rispetto all'agente di primo livello.

AgentNwTL La classe AgentNwTL rappresenta gli agenti di terzo livello (Third Level). La capacitá deliberativa di questo agente gli consente di acquisire la denominazione di agente 'intelligente' intendendo il fatto che opera praticamente come un agente umano di fronte al gioco della battaglia navale. Infatti ha la *capacitá di imparare* , cioé é in grado di tenere traccia delle caselle avversarie colpite, quindi di muovere i successivi colpi nell'intorno del punto colpito per affondare la nave completamente. Se si dovesse descrivere il comportamento di questo agente in termini di sistema intenzionale si direbbe:

- non conoscendo lo scenario dell'avversario non posso fare altro che credere che lanciando casualmente prima o poi colpiró un pezzo di nave (evitando di colpire caselle gia colpite), a quel punto ovvio che cercare nell'intorno del punto colpito sia la soluzione migliore, in oltre trovata la direzione sulla quale attaccare inutile cambiarla, al massimo si pu cambiare il verso. Se effettuo un giro su me stesso allora significa che un altro agente ha colpito la nave che stavo colpendo io, quindi ne cerco un altra perché non posso sapere l'evoluzione degli attacchi.

Si nota quindi la presenza di due piani precompilati, uno che consente di sparare in maniera random, l'altro, piú efficace ma che necessita di un'informazione supplementare per essere utilizzato, che consente di sparare in un intorno, muovedo ad esempio in senso orario per trovare la dimensione e la direzione della nave posizionata. La deliberazione iniziale riguardante l'accettare oppure no il gioco é qui praticamente inesistente, infatti si ha la consapevolezza di essere adeguati a qualsiasi tipo di scenario, come ad esempio la sfida con agenti umani, pur potendo esistere agenti piú abili.

Un Agente termina la sua vita al termine del gioco, ovvero se viene sconfitto o se il vincitore.

Interfaccia grafica Tutti gli agenti sono legati ad una interfaccia grafica che evidenzia la disposizione delle navi e dei colpi subiti differenziandoli rispettivamente con il colore rosso e nero. Lo scopo della interfaccia é quindi di sola presentazione dell'attivitá del gioco.

3.4 Liguaggio

Il linguaggio che deve essere in grado di parlare un Agente che voglia partecipare al gioco é il linguaggio logico. Quindi come sintassi ci si rifá alla sintassi Prolog. Per quello che riguarda la semantica e l'ontologia i seguenti predicati hanno i seguenti significati all'interno del gioco:

- nw_booking(N,Struct) é attualmente aperta l'iscrizione ad un gioco per N utenti con le caratteristiche presenti nel campo Struct che ha forma
  - struct(Dimension,Size,Sheep) gioco a Dimension dimensioni, di grandezza Size con numero e tipo di barche specificato in Sheep che ha forma
    - sheep(Length,N) nave di lunghezza Length di istanze N;
- nw_num_player(Id,N) il numero di giocatori al gioco Id é N;
- nw_max_player(Id,Max_player) il numero massimo di giocatori per il gioco Id é Max_player;
- nw_ready_for_dice(Id) tutti i giocatori si sono iscritti quindi si attende il lancio del dado per ol gioco Id;
- nw_start_game(Id) é iniziato il gioco Id;
- nw_action(Id_nw,Id_player) é interpellato al gioco Id_nw il giocatore Id_player;
- nw_now_play(Id_nw,Id_player,N) puó lanciare nel gioco Id_nw il giocatore Id_player a cui restano N colpi
- nw_winner(Id_nw,Id_player) il giocatore Id_player ha vinto il gioco Id_nw.

Sono poi definiti i predicati dedicati al gioco utilizzabili anche dagli agenti umani:

- nw_attack(Id_nw,Id_player,Id_target,Coord) l'agente Id_player attacca Id_target per il gioco Id_nw alla coordinata Coord espressa come una classica lista Prolog;

- nw_response(Id_nw,Id_player,Response) l'agente Id_player dice il risultato dell'attacco ultimo subito per il gioco Id_nw con la variabile Response che puó valere:
  - nw_defeated si é stati sconfitti e si abbandona il gioco;
  - nw_sunk é stata affondata la nave a quella coordinata;
  - nw_hit é stata colpita la nave a quella coordinata;
  - nw_water non é stata colpita nessuna nave;

4 Conclusioni

Dalle prove effettuate su numerose partite facendo giocare le tre tipologie di agenti si nota che la differente *intelligenza* degli stessi risulta piú efficace nella vittoria finale tanto piú risulta poco denso di navi lo scenario. L'agente di primo livello risulta comunque praticamente sempre perdente rispetto agli altri due, tra i quali, come ci si aspettava, l'agente di terzo livello é normalmente il vincente.
Nella sfida tra agente umano ed agente di terzo livello si ha un maggior numero di vittorie da parte dell'agente umano effettuate un numero di partite sufficienti per capire che la disposione delle navi lungo la prima dimensione risulta scovabile con maggiore difficoltá (intesa come numero di mosse necessarie) dall'agente di terzo livello.

Estensioni Questa considerazione rende quindi individuabile con un numero di partite elevato un giocatore virtuale rispetto ad uno umano. Per migliorare questa situazione si piuó pensare di creare un piano per l'attacco che non lanci, una volta idividuato un pezzo di nave, sulle dimensioni in maniera randomatica e non seguendo un ordine preciso. Per quello che riguarda il gioco di squadra descritto in analisi si potrebbe realizzare fornendo agli agenti un informazione come l'appartenenza ad una squadra che puó essere ad esempio identificata da un range di numeri identificativi degli agenti, a quel punto gli agenti potrebbero non attaccare agenti che hanno identificativi compresi nel range della propria squadra. Questo imporrebbe una ulteriore specifica Respect per il centro di tuple in modo che questo possa decreta-

re il vincitore una volta rimasta in campo una sola squadra e non un solo giocatore.