Ticket To Ride

Denis Brighi
Luca Domeniconi

TTR - Il gioco

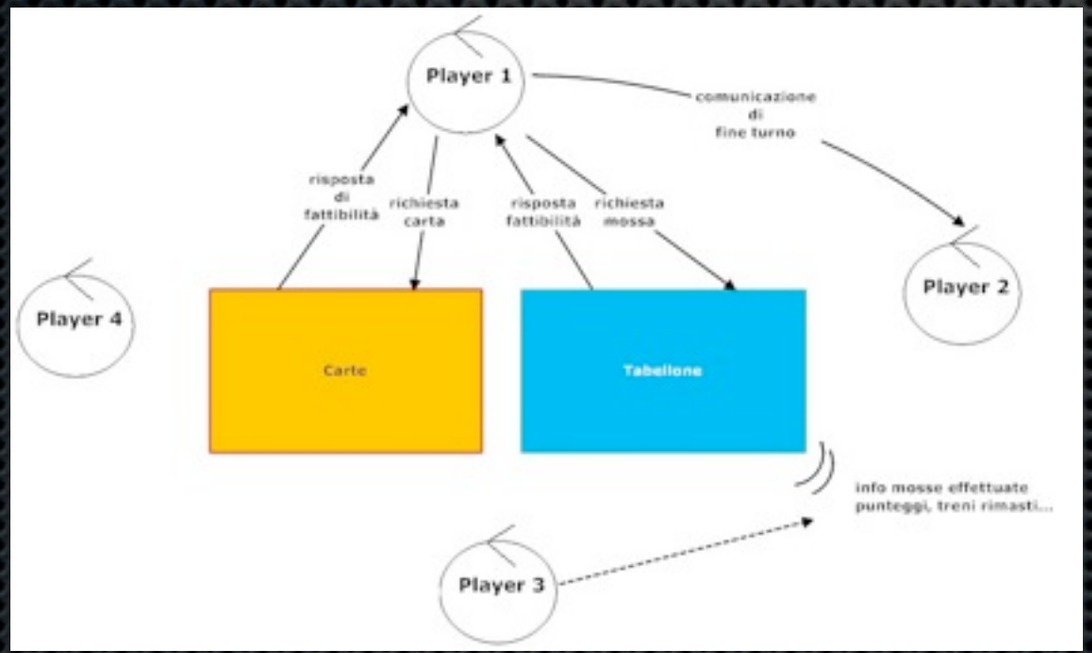
- Gioco da tavolo (2-5 giocatori):
  - Città collegate da itinerari
  - Ogni itinerario un numero (numero treni da inserire per completarlo)
- A disposizione (per ogni giocatore):
  - 45 treni
  - 5 carte vagone ferroviario coperte
  - 3 carte tratta (2 max scartabili a scelta)
- Obiettivo: miglior punteggio
- Come fare punti?
  - Inserendo trenini (scartando carte colore)
  - Collegando tratte obiettivo
  - (-) tratte obiettivo non completate

TTR - Regole

- Regole inserimento treni:
  - Tratta colorata - scartare carte stesso colore tratta
  - Tratta grigia (any) - carte qualsiasi ma stesso colore
- Regole gioco (ogni turno):
  - Scartare tratta
  - Pescare:
    - 2 carte colore scoperte
    - 2 carte coperte
    - 1 carta coperta e una scoperta
    - 1 jolly
    - 3 nuove carte tratta
- Partenza ultimo turno: giocatore rimane con 3 trenini

TTR - Analisi/Progettazione

- Entità:
  - Giocatori
  - Mazzo coperto
  - Mazzo scoperto
  - Mazzo tratte
  - Tabellone
- Chi gestisce la partita?



Controllo decentralizzato

Controllo centralizzato

TTR - Analisi/Progettazione

Gestione decentralizzata

- complessità

+ simula comportamento umano

+ migliori performance multi-core

Gestione centralizzata

+ semplicità

- Scelta: gestione centralizzata -> entità per controllo partita (tabellone)

TTR - Scelta Java/Prolog

- Controller:
  - Entità “intelligenti” (simulano comportamento umano): giocatori (cpu) -> Prolog
  - Entità “funzionali” (le rimanenti) -> OOP -> Java
- View:
  - Ottimo supporto Java per grafica -> Java
- Model:
  - più entità in Java che Prolog -> Java
  - Accesso in copia da Prolog

TTR - La Strategia

- Il raggiungimento del maggior numero di obbiettivi possibile
- Il completamento di un obbiettivo
- Lo scarto di un itinerario favorevole al raggiungimento di un obbiettivo
- La collezione di carte colore utili al conseguimento di un itinerario colorato
- La collezione di carte colore utili al conseguimento di un itinerario “any”

TTR - Creazione del piano

- plan(L):– findall([A,B],routeCard(A,B),L2),
  paths(L2, L1), quicksort(L1, L).
- paths([], []).

```
paths([[A,B]|T], O):–dijkstra(A,B,Sol,Val, ),!,
        (Val=0, !, O=O1;O=[[Val, Sol]|O1]),
        paths(T, O1).

paths([[A,B]|T], O):–retract(start( )), retract(stop( )), paths(T, O)
```

TTR - Dijkstra

1. **Inizializzazione:**
   Poniamo S={1}, T={2,3,...,n}, f(1)=0, J(1)=0.
   Poniamo f(i)=p(1,i), J(i)=1 per tutti i nodi adiacenti ad 1.
   Poniamo f(i)= ∞, per tutti gli altri nodi.

2. Assegnazione etichetta permanente:
   Se f(i)= ∞ per ogni i in T STOP.
   Troviamo j in T tale che f(j)=min f(i) con i appartenente a T
   Poniamo T=T{j} e S=S∪{j}
   Se T=Ø STOP

3. Assegnazione etichetta provvisoria:
   Per ogni i in T, adiacente a j e tale che f(i)>f(j) +p(j,i) poniamo:
   f(i)=f(j)+p(j,i)
   J(i)=j

4. Andiamo al passo 2

```
dijkstra(A,B,Sol,Val,Sol2) :-

    assert(start(A)),
    assert(stop(B)),

    setListInit([],T,S1),
    findNeighbour(A,L3),
    setLabel(A,L3,L4),
    setInitialLabel(T,L4,T2),
    cicle(S1,SO,T2,X),
    member([B,Val,_], SO),
    makeSol(A,B, SO, [B], Sol).

setInitialLabel(L,[[H,N]|T],LO) :-

    start(A),member([H,N1,N2],L),
    eraseNode([H,N1,N2],L,L1),
    append([[H,N,A]],L1,L2),
    setInitialLabel(L2,T,LO).
```

TTR - Route e Dist

- Route:

```
route1(X, IDA, IDB, NT, COL):-route(X, IDB, IDA, NT, COL).
route1(X, IDA, IDB, NT, COL):-route(X, IDA, IDB, NT, COL).
```

- Dist:

```
dist(IDA,IDB,0):-route1(ID,IDA,IDB,_,_),myId(X),routeOwner(ID,X).
dist(IDA,IDB,NTrains) :-
                route1(ID,IDA,IDB,NTrains,_),not (routeOwner(ID,_)).
```

TTR - Controllo del piano

- plan([
  [10,[21,22,23,17,11,7,14]],
  [8,[17,11,7,2,1]],
  [6,[34,35,25,15]]
  ]).
- checkPlan ([ ]).
  checkPlan([[_,Itinerario]|T]) :– checkPlan2(Itinerario), !,
  checkPlan(T),
- checkPlan2 ([H1]) .
  checkPlan2([H1,H2|T]) :– dist(H1,H2,A), !,
  checkPlan2([H2|T]).

TTR - Selezione Mossa

Prendere carte obbiettivo

1. Occupare un itinerario “Colorato”
2. Occupare un itinerario “Any”
3. Prendere carte dal mazzo scoperto
4. Prendere un jolly dal mazzo scoperto
5. Prendere carte dal mazzo coperto

```
selectMove(takeRouteCards):–
        not(cardTaked), plan([]), !.
```

1. 2.

```
selectMove(O):– not(cardTaked), plan(X),
        tryDiscardAll(X, O), !.
```

3.

```
selectMove (O):– plan (X),
        lookForVisible (X, O), !.

selectMove(O):– plan(X), checkAny(X),
        nonInterestingCards(X, RC),
        handColorCounter(RC,Out),
        quicksort(Out,O1),
        selectAVisibleCard(O1,O), !.
```

4.

```
selectMove(takeVisible(1)):–
        not(cardTaked),
        visibleCard(1).
```

5.

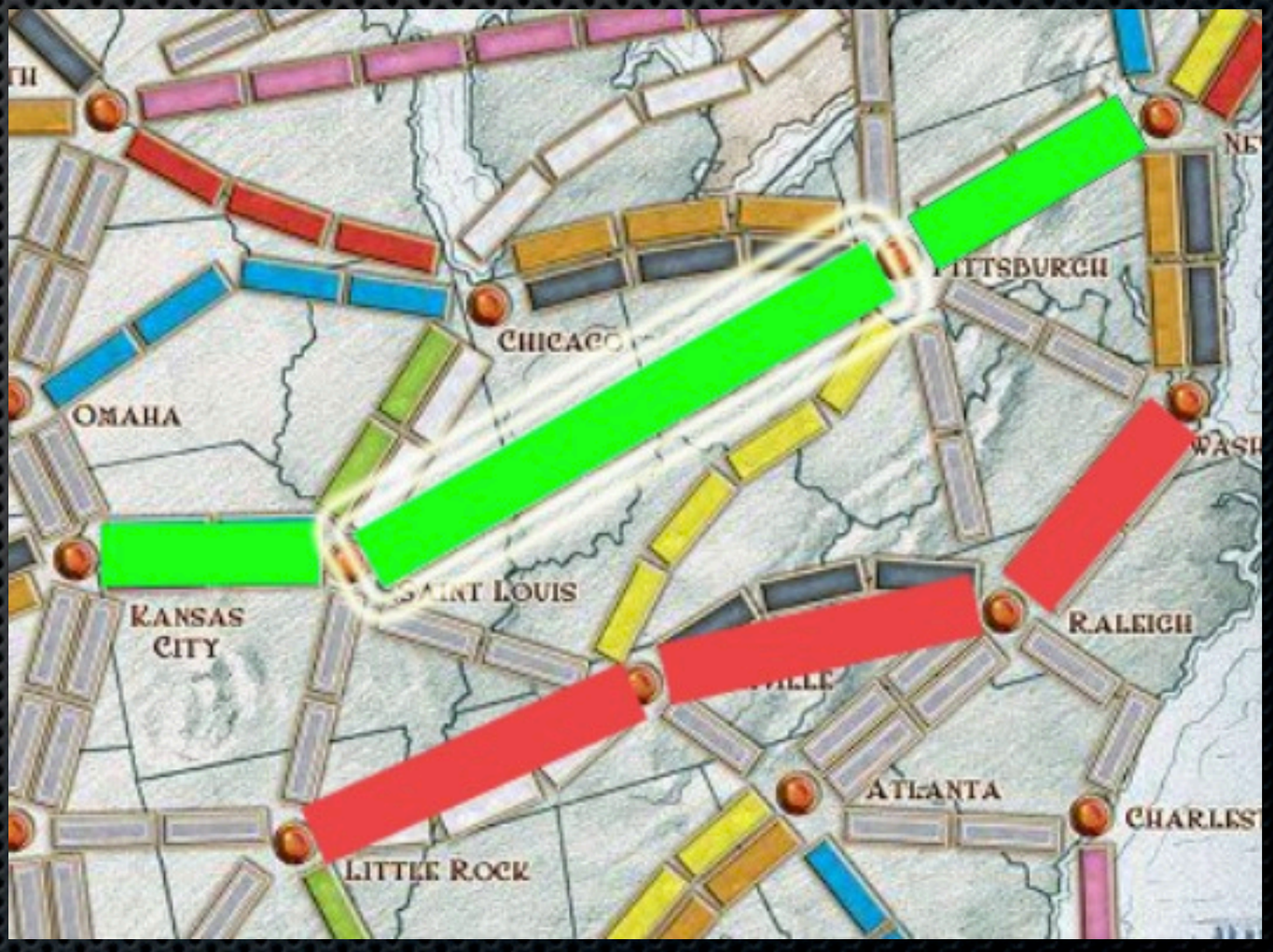
```
selectMove(takeHidden).
```

TTR - Ottimizzazioni Successive

- Situazione Iniziale

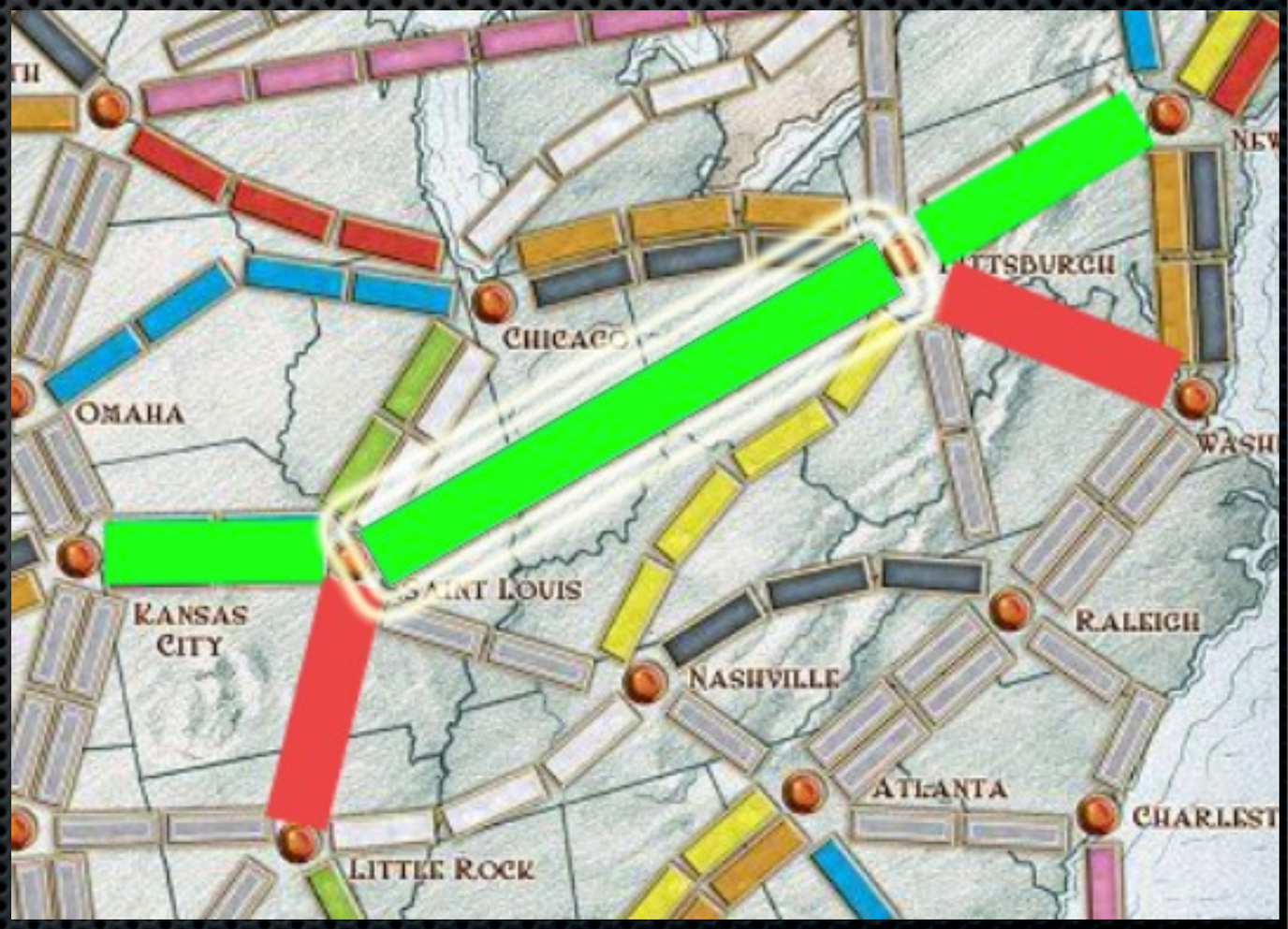
TTR - Ottimizzazioni Successive

- Occupazione di un itinerario



TTR - Ottimizzazioni Successive

- Piano Aggiornato



TTR - Integrazione Java-Prolog

- Java using Prolog, approccio a libreria
- Player.java
- IA Player Modulare:
  - dijkstra.pl
  - checkPlan.pl
  - plan01.pl

TTR - Integrazione Java-Prolog

- Costruzione della Teoria
- Creazione del piano
- Verifica del piano
- Selezione della prossima mossa
  - “takeRouteCards”
  - “takeRoute(IDR, Col, N, NUJ)”
  - “takeVisible(Col)”
  - “takeHidden”

Ticket To Ride

Denis Brighi
Luca Domeniconi