Projekat – Havana (Havannah)

Milena Frtunić Gligorijević
Nataša Veljković
Vladan Mihajlović

Osnovne informacije

- Cilj projekta:
  - Formulacija problema
  - Implementacija algoritama za traženje (algoritama za igre)
  - Implementacija procene stanja korišćenjem pravila i zaključivanja
- Jezik: Lisp
- Broj ljudi po projektu: 3
- Datum objavljivanja projekta: 5.11.2018.
- Rok za predaju: 5.1.2019.

Ocenjivanje

- Broj poena:
  - Projekat nosi maksimalno 20% od konačne ocene
  - Poeni se odnose na kvalitet urađenog rešenja, kao i na aktivnost i zalaganje studenta
- Status:
  - Projekat je obavezan!
  - Minimalni broj poena koji se mora osvojiti je 5!
  - Očekuje od vas da ozbiljno shvatite zaduženja!
  - Ukoliko ne uradite projekat u navedenom roku, naredna prilika je tek sa sledećom generacijom, po pravilima koja će biti tada definisana!

Takmičenje/turnir

- Posle predaje projekta biće organizovano takmičenje.
- Planirani termin takmičenja je sredina januara.
- Prva tri mesta na turniru donose dodatne poene: 5 za prvo mesto, 3 za drugo i 2 za treće mesto (računaju se kao dodatni poeni za angažovanje u toku semestra).

Pravila ponašanja

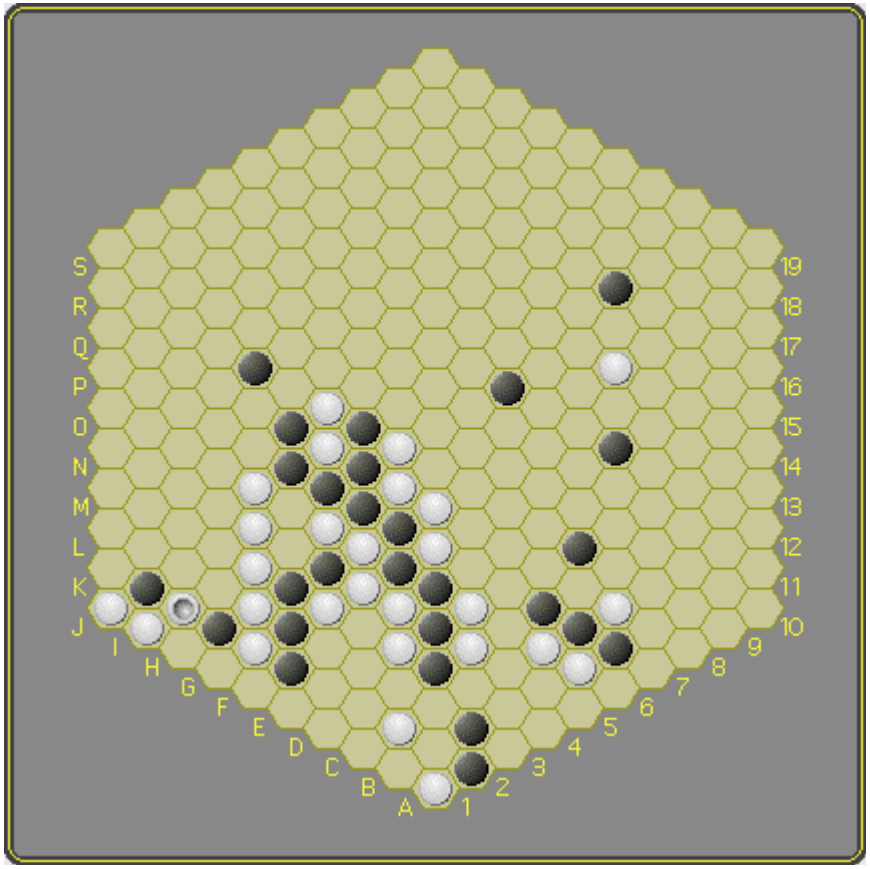
- Probajte da uradite projekat sami, bez pomoći kolega ili prepisivanja.
- Poštujte tuđi rad! Materijal sa Web-a i iz knjiga i radova možete da koristite, ali samo pod uslovom da za sve delove koda ili rešenja koje ste uzeli od nekog navedete referencu!
- Ne dozvolite da od vas neko prepisuje, tj. da neko od kolega koristi vaš rad i vaše rezultate!
- Ako radite u timu, ne dozvolite da vaš kolega iz tima ne radi ništa! Nađite mu zaduženja koja može da uradi – ako mu nešto ne ide, nađite mu druga zaduženja.

Faze izrade projekta

- Formulacija problema i implementacija interfejsa
  - Rok: 18.11.2018. godine
- Implementacija operatora promene stanja
  - Rok: 2.12.2018. godine
- Implementacija Min-Max algoritma za traženje sa alfa-beta odsecanjem
  - Rok: 16.12.2018. godine
- Definicija heuristike (procena stanja)
  - Rok: 5.1.2019. godine

- Rezultat svake faze je izveštaj koji sadrži dokument sa obrazloženjem rešenja i datoteku sa kodom.

Igra *Havana*



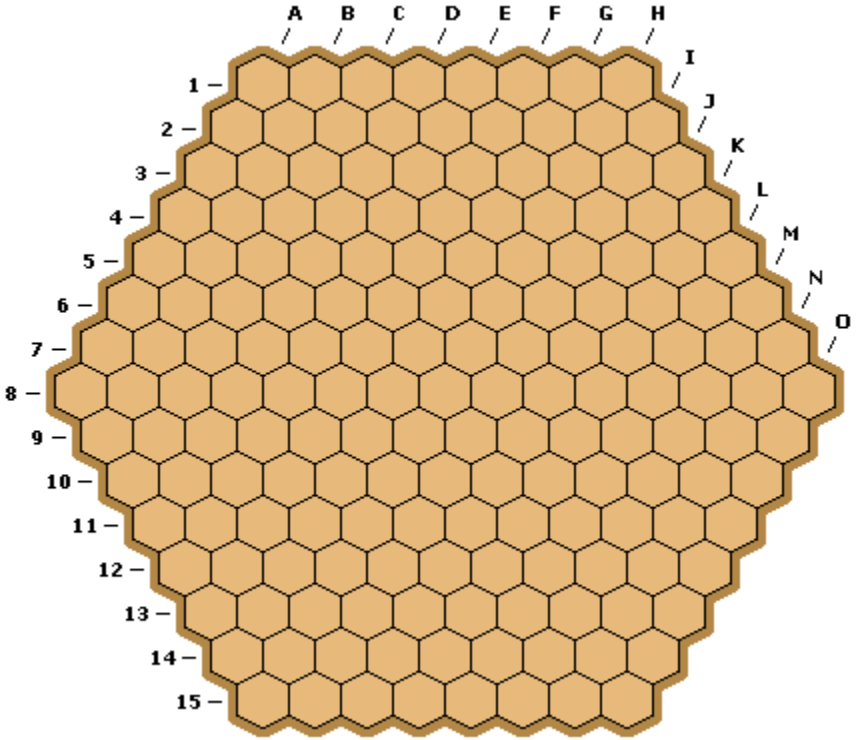
Opis problema *Havana*

- Problem je igra *Havana (Havannah)*
- Tabla je oblika šestougla stranice n koju zadaje korisnik (preporučeno n=6, iako je se u za igru koristi n=8; maksimalno n=12)
- Dva igrača crni i beli (X i O) naizmenično odigravaju po jedan potez stavljajući svoje perle na proizvoljno nepopunjeno polje
- Tabla je na početku prazna
- Pobednik je prvi igrač koji napravi prsten (*ring*), most (*bridge*) ili vilu (*fork*) od svojih perli
- Igra čovek protiv računara i moguće izabrati da prvi igra čovek ili računar

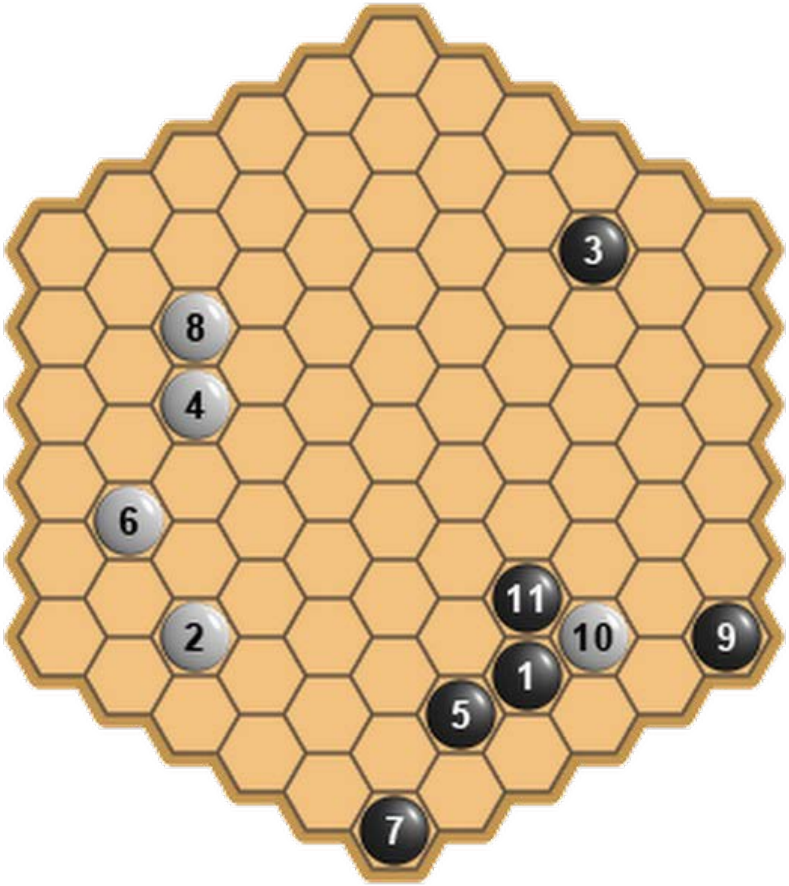
Pravila igre *Havana*

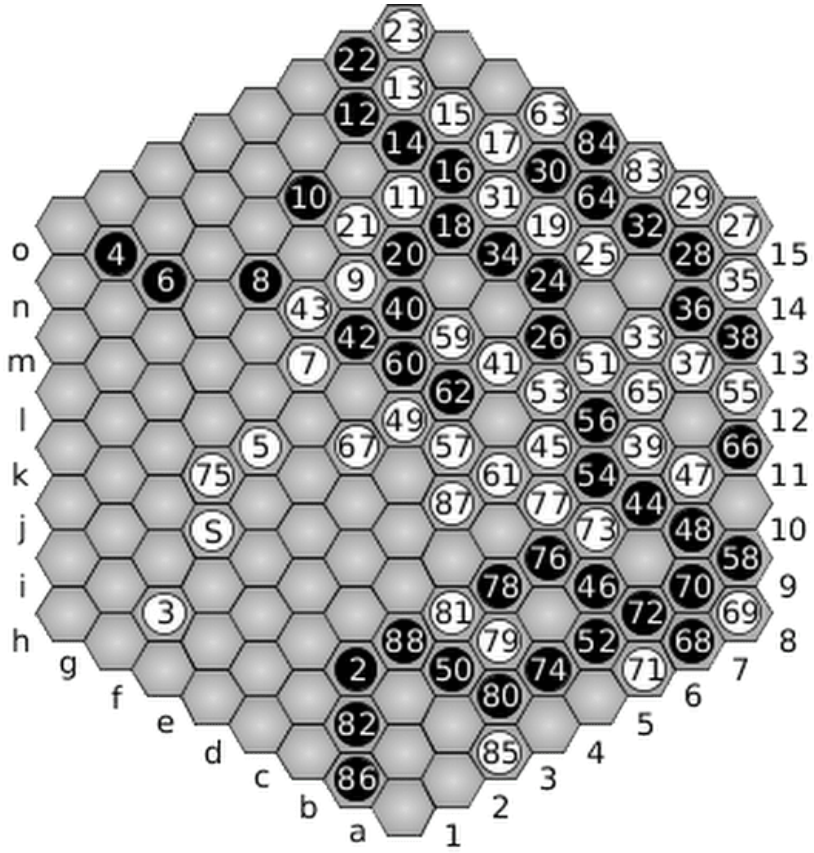
- Igrači povlače poteze naizmenično
- Igrač, u jednom potezu, može da postavi samo jednu perlu
- Perla se može postaviti samo na prazno polje
- Pobednik je prvi igrač koji od svojih perli napravi:
  - prsten (*ring*) – niz susednih perli koje opkoljavaju makar jedno polje (prazno ili popunjeno bilo čijom perlom)
  - most (*bridge*) – niz susednih perli koje povezuju barem dva temena (ugla) šestougaone table
  - vilu (*fork*) – niz susednih perli koje povezuju barem tri stranice šestougaone table (teme, odnosno ugao, se ne smatra delom stranice)

Havana – Početak igre

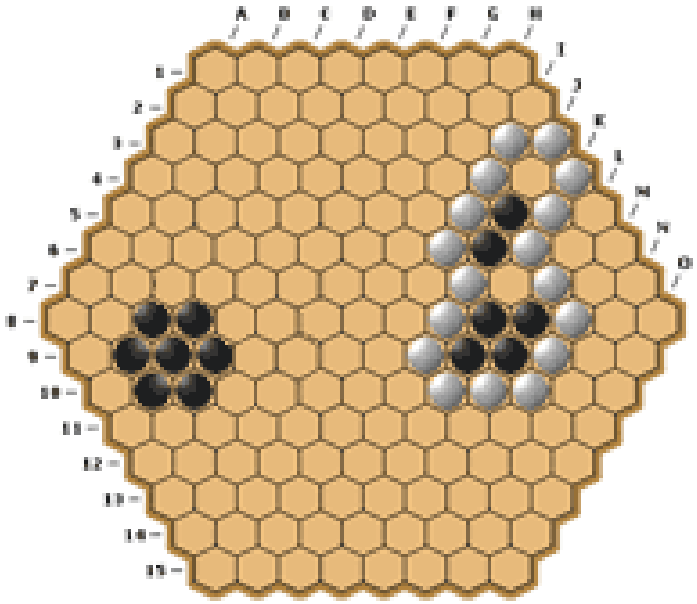


Havana – Primeri poteza



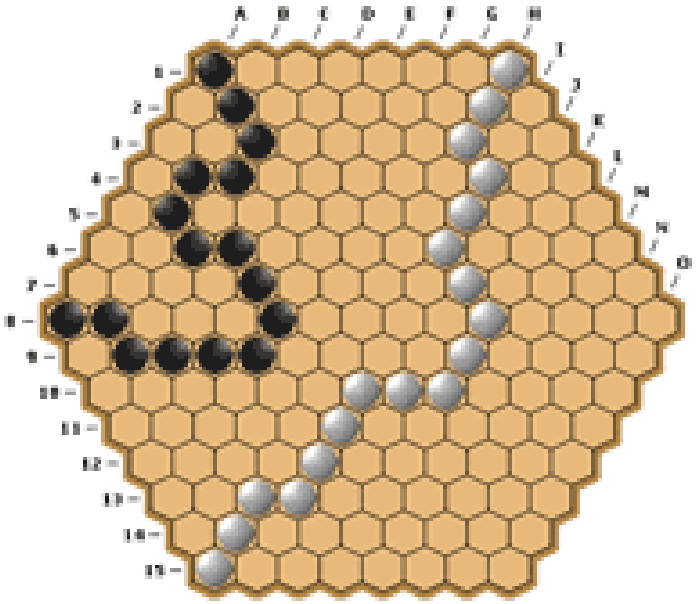


Havana – Kraj igre prsten (*ring*)



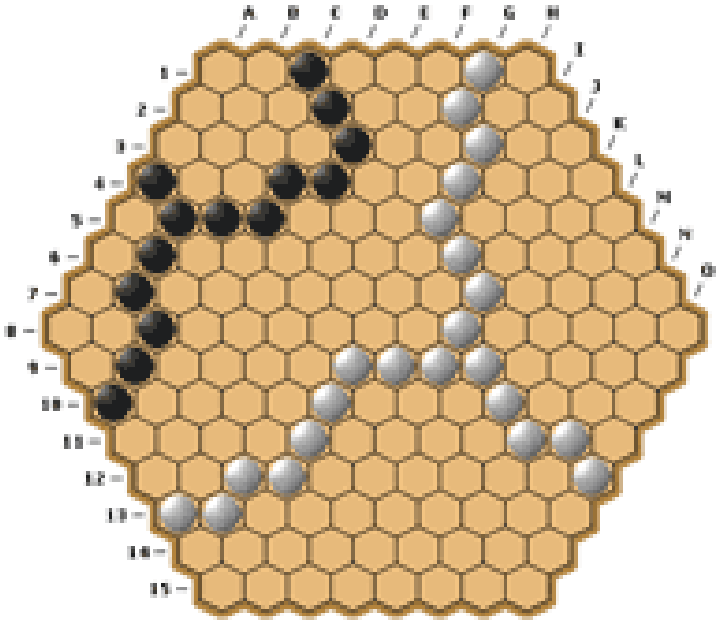
- Prsten može biti prazan
- Prsten može biti popunjen svojim i/ili protivničkim perlama
- Prsten može biti delimično popunjen svojim i/ili protivničkim perlama

Havana – Kraj igre most (*bridge*)



- Most povezuje najmanje dva bilo susedna bilo nesusedna temena (ugla) šestougaone table

Havana – Kraj igre vila (*fork*)



- Vila povezuje najmanje tri bilo susedne bilo nesusedne stranice šestougaone table
- Temena (uglovi) se ne smatraju delom stranice

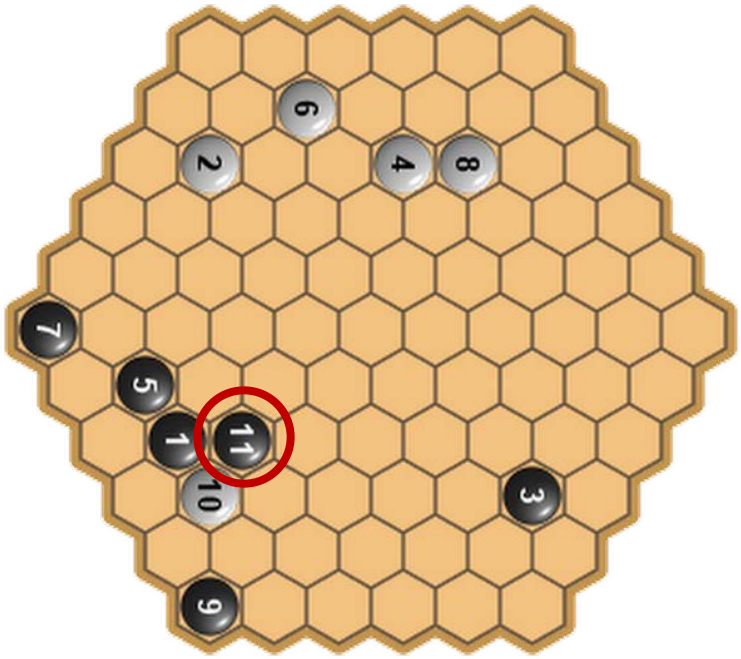
Zadatak I – Formulacija problema i interfejs (1)

- Definisati način za predstavljanje stanja problema (igre)
- Napisati funkciju za postavljanje početnog stanja na osnovu zadate veličine table
- Napisati funkcije za testiranje ciljnog stanja, tj. da li je neko od igrača napravio prsten (*ring*), most (*bridge*) ili vilu (*fork*)

Zadatak I – Formulacija problema i interfejs (2)

- Omogućiti izbor ko će igrati prvi (čovek ili računar)
- Prvi igra uvek igrač X, a drugi igrač O
- Implementirati funkcije koje obezbeđuju prikaz proizvoljnog stanja problema (igre)
- Realizovati funkcije koje na osnovu zadatog poteza igrača, u obliku (vrsta kolona), omogućavaju:
  - proveru da li je potez valjan
  - ako jeste, tj. promenu prosleđenog stanja problema (igre) odigravanjem poteza

Zadatak I – Interfejs




```
    0 1 2 3 4 5
A     - - - - - - 6
B   - - O - - - - 7
C  - O - - O O - - 8
D - - - - - - - - - 9
E - - - - - - - - - - 10
F X - - - - - - - - - -
G  - X - - - - - - - -
H   - X X - - - - - -
I    - O - - - - X -
J     - - - - - - -
K      X - - - - -
```

Potez X: (H 4)

Zadatak II – Operatori promene stanja

- Napisati funkcije za operatore promene stanja problema (igre) u opštem slučaju (proizvoljno stanje na tabli)
  - Na osnovu trenutne (proizvoljne) situacije na tabli (stanja) i zadatog (validnog) poteza formira novu situaciju na tabli (stanje). Ne menjati postojeće stanje već napraviti novo i na njemu odigrati potez.
  - Na osnovu trenutne (proizvoljne) situacije na tabli (stanja) i igrača koji je na potezu formira listu svih mogućih situacija na tabli (stanja), korišćenjem funkcije iz prethodne tačke
- Realizovati funkcije koje obezbeđuju odigravanje partije između dva igrača (dva čoveka, ne računara i čoveka)
  - unos poteza i provera da li je potez moguć
  - ukoliko nije moguć zahtevati unos novog poteza
  - ukoliko je moguć odigrati ga i promeniti trenutno stanje
  - prikazati novonastalo stanje sistema

Zadatak III – Min-max algoritam

- Implementirati Min-Max algoritam sa alfa-beta odsecanjem za zadati problem
- Obezbediti da funkcija Min-Max sa alfa-beta odsecanjem ima ulazni parametar kojim se definiše dubina pretraživanja
- Obezbediti da funkcija Min-Max sa alfa-beta odsecanjem vrati potez koji treba odigrati ili stanje u koje treba preći
- Funkciju za određivanje heuristike ne treba implementirati
  - Napraviti funkciju koja za odgovarajuća stanja vraća karakteristične vrednosti samo u svrhu testiranja ispravnosti napravljenog Min-Max algoritma

Zadatak IV – Heuristika

- U implementaciju Min-Max-a sa alfa-beta odsecanjem dodati funkciju za procenu stanja koja se poziva kada se dostigne zadata dubina traženja.
- Implementirati funkciju koja vrši procenu stanja na osnovu pravila zaključivanja
- Funkcija za procenu stanja kao parametre treba da ima oznaku igrača za kojeg računa valjanost stanja, kao i samo stanje za koju se računa procena.
- Procena stanja se mora vršiti isključivo korišćenjem mehanizma zaključivanja nad prethodno definisanim skupom pravila. Zadatak je formulisati skup pravila i iskoristiti ih na adekvatan način za izračunavanje heuristike.
- Za izvođenje potrebnih zaključaka (izvršavanje upita nad skupom činjenica kojima se opisuje stanje) koristiti mašinu za zaključivanje.
- Implementirati funkciju koja prevodi stanje u listu činjenica …