Capture the flag pacman rešenje primenom MinMax algoritma

Opis Projekta

Capture the flag pacman je igra sa dva tima, gde su timovi protivnici. Clij igre je da igrači sakupe hranu sa protivnikove strane mape i donesu je nazad na svoju stranu. Igrač, kad pređe na protivnikovu stranu mape, postaje *pacman*. Igrač je na svojoj strani mape duh. Duhovi mogu da pojedu *pacmane*, pa kad je igrač na protivnikovoj strani mape treba da se pazi protivnika. Takođe, u igri postoje *Power kapsule* koja kada su pojedene čine protivničke duhove uplašenima. Uplašenog duha *pacman* može da pojede.

Apstraktni pristup rešavanju problema

Igra spada u grupu zero sum game, zato što je pobeda uslovljena porazom protivničkog tima. Zbog ovoga iskorišćen je Min Max algoritam. Algoritam radi tako što pretražuje stablo. Stablo predstavlja sva stanja igre, gde su čvorovi pojedinačna stanja igre.

Čvorovi u stablu povezani su samo ako može da se stigne iz jednog stanja igre u drugo. Pretraživanjem tog grafa dodeljuju se vrednosti čvorovima, gde vrednost predstavlja pogodnost čvora, tj. stanja igre. Listovi ovog grafa su stanja gde se igra završila. U klasičnom Min Max algoritmu na početku pretraživanja jedino listovi imaju vrednosti, gde je ta vrednost rezultat igre. Ostali čvorovi dobijaju vrednost tako što preuzimaju jednu od vrednosti potomaka.

Čvor koji nije list može da bude jedan od dva tipa: min čvor ili max čvor. Min čvor je stanje u igri gde protivnik odlučuje. Max čvor je stanje u igri gde naš igrač odlučuje. U min čvorovima se preuzima vrednost potomka koja je najmanja. U max čvoru se preuzima vrednost koja je najveća. Na ovaj način koren stabla će imati vrednost koja je garantovani ishod igre ako su oba igrača savršena. Tok igre će ispratiti tok grana stabla sa kojih je došla vrednost u koren.

Problem koji se predstavlja u praksi sa Min Max algoritmom je da su stabla koja se pretražuju ogromna. Ovaj problem je u našem projektu rešen na 2 načina:
1. Iskorišćena modifikacija Min Max algoritma koji se zove Alfa Beta pruning
2. Ograničena dubina stabla.

Alfa Beta pruning modifikacija algoritma dodaje još dve vrednosti svakom čvoru. Tim vrednostima se tradicionalno dodaje naziv alfa i beta. Alfa predstavlja vrednost koju čvor može minimalno da garantuje, a Beta predstavlja vrednost koju čvor može maximalno da garantuje. Ako max čvor garantuje više nego njegov predak min čvor, onda je sigurno da taj predak neće ni izabrati tu putanju, jer bi tu gubio. Zbog ovoga možemo da ignorišemo deo stabla, sa čime štedimo na vremenu izračunavanja putanje.

Uprkos štednji koju dobijemo od *Alfa Beta pruning*, stablo je još uvek ogromno. Zbog ovoga ograničavamo dubinu stabla. Međutim, ograničavanjem dubine dolazi se do problema sa *Min Max algoritmom*. *Min Max algoritam* na početku pretrage dodeljuje vrednosti samo listovima, tj. krajnjim

stanjima igre. Pošto smo stablo ograničili dubinski, listovi više nemaju vrednosti, i algoritam ne može da otpočne. Ovo rešavamo heuristikama. Umesto da listovima dodelimo krajnje stanje igre, kao u klasičnom *Min Max algoritmu,* mi dodeljujemo procenjenu vrednost korišćenjem heuristike.

Konkretni pristup

U konkretnom pristupu ne možemo da izgenerišemo stablo jer bi zauzimalo previše memorije. To se rešava sa rekurzivnim funkcijama:

1. min_function – koji radi posao min čvora

2. max_function – koji radi posao max čvora.

Ove dve funkcije međusobno pozivaju jedna drugu. Takođe, min funkcija poziva samu sebe, jednom za svakog protivničkog igrača. Heuristika je izračunata u getWeightedEstimates. Heuristika zavisi od:

1. Udaljenosti *pacmana* od hrane koju treba da sakupi

Ovu vrednost čini negativna udaljenost između najbliže hrane i *pacamana*. Ona je množena sa goPelletWeight. goPelletWeight je ili 1 ili 0. Kada *pacmani* odluče da sakupljaju onda je goPelletWeight 1, inače je 0. Ova vrednost se stavlja na 0 u pojedinim situacijama da bi zbog heuristike pod 2. pakman iskazao želju da se vrati sa sakupljenom hranom.

2. Udaljenosti svoje strane

Ovu vrednost čini negativna udaljenost između pacmana i svoje strane. Ona je množena sa returnWeight. ReturnWeight može da bude 0 ili hasFoodWeight + 1. Kada *pacmani* odluče da sakupljaju onda je goPelletWeight hasFoodWeight + 1, inače je 0. Ova vrednost se stavlja na 0 u pojedinim situacijama da bi zbog heuristike pod 1. pakman iskazao želju da ide i sakuplja hranu umesto da samo čeka na svojoj strani. Inače, ova vrednost je hasFoodWeight + 1 – da je manje nastavio bi da sakuplja hranu jer bi smatrao da je sakupljati vrednije nego da se vrati sa onim što ima.

3. Sakupljenih bodova

Ovu vrednost čini zbir hrane koju drži kod sebe i hrane koju je vratio na svoju stranu pomnoženi sa težinama hasFoodWeight i returnedFoodWeight respektivno. ReturnedFoodWeight je veće od hasFoodWeight, jer inače pacman ne bi iskazao želju da se vrati na svoju stranu, tj. smatrao bi da hrana koja je kod njega je dragocenija nego hrana koju je preneo na svoju stranu terena i zapravo zaradio bodove.

4. Protivnika koji pokušava da pojede našeg igrača

Ova vrednost je -1000 jer nije povoljno biti pojeden.

5. Protivnika kojeg pokušava da pojede naš igrač

Ova vrednost je negativna i pada što je više protivnik sakupio hrane, takođe opada što više je protivnik udaljen. Ona je množena sa mustGetOpponent koja je 2 za pakmane koji više sakupljaju hranu, a 50 je za pakmane koji više brane svoju stranu.

Redosled rešavanja

Prvo se implementirao klasični *Min Max algoritam.* Nažalost nema podataka o ovom stanju projekta jer traje jako dugo da se prođe kroz celo stablo.

Posle toga se ograničilo dubina stabla na 10 (5 poteza). Za heuristiku se uzelo samo ishod igre u tom potezu. Vreme prolaska kroz stablo se spustio na 7 min, što je još uvek puno više od ciljanog 3–1s.

Posle toga se implementirao *Alfa Beta prunin.* Vreme prolaska kroz stablo se spustilo na oko 14s u opsegu od plus minus 6s.

Da se postigne žejeno vreme spustili smo dubinu stabla na 4 (2 poteza). Vreme prolaska kroz stablo se spustio na ispod 1s. U ovom trenutku *pacmani* nisu još radili ništa korisno jer za dubinu od 4 ili 10 *pacmani* ne stižu do hrane pa ni ne vide kuda bi trebali da idu.

Posle ovoga su isprobavani različite heuristike. Heuristika je naterala pacmana da ide do hrane, da igra na dosta dobar način.

Zaključak

Min Max algoritam radi savršeno u idealnim slučajevima kada imamo beskonačno memorije i beskonačnu procesorsku moć. U stvarnosti Min Max algoritam koji je ograničen na dubinu stabla 4 ima dosta mane. Npr. pacman može da uđe u ćorsokak od dužine 3 i više, misleći da je to dobar potez, neće predvideti da će biti zarobljen ako pokuša da izvadi hranu iz ćorsokaka.