Algoritmi određivanja Nashove ravnoteže u teoriji igara

Zadro, Marina

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Marina Zadro

ALGORITMI ODREĐIVANJA NASHOVE RAVNOTEŽE U TEORIJI IGARA

Diplomski rad

Voditelj rada:
doc. dr. sc. Lavoslav Čaklović

Zagreb, 2015.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Zahvaljujem mentoru doc.dr.sc. Lavoslavu Čakloviću na nesebičnoj pomoći pri izradi diplomskog rada.
Posebno se zahvaljujem svojim roditeljima i bratu na bezuvjetnoj podršci i razumjevanju.

Sadržaj

Uvod

Teorija igara je matematička disciplina koja proučava strateško odlučivanje. Ona nam predočava tok odluka u nekoj igri te kako i zašto je neki igrač donio određenu odluku. Teorija igara ima veoma široku primjenu. U primjenjenoj matematici primjenjujemo ju u cilju proučavanja ljudskog i životinjskog ponašanja. U ekonomiji ju koriste menadžeri radi lakšeg donošenja odluka. Primjena se sve više širi i u društvenim znanostima, posebno u politici, sociologiji i psihologiji. 1930 Ronald Fisher je primjenjivao teoriju igara u biologiji prilikom proučavanja životinjskog ponašanja. Također, veoma je bitna u umjetnoj inteligenciji. Ona se koristi da bi opisali, predvidili i dali objašnjenje za određeno ponašanje, da bi razvili teoriju o etičkom i normativnom ponašanju te da bi lakše kopirali takvo ponašanje.

Jedan veoma bitan koncept u Teoriji igara je Nashova ravnoteža koju je opisao John Forbes Nash Jr. Ona nam pomaže da bi predvidjeli što će se dogoditi ukoliko nekoliko ljudi ili institucija sudjeluju u donošenju neke odluke. Odluke mogu biti različite, od odabira najboljeg ureda koji će zadovoljiti zahtjeve neke tvrtke do odabira sljedećeg koraka u igri "Križić-kružić".

Ovaj rad se fokusira na igre s dva igrača. Na početku je opisana normalna forma zapisa igara te objašnjena razlika između čistih i mješovitih strategija. U drugom poglavlju je definirana Nashova ravnoteža, kako ju pronaći u igri zapisanoj u normalnoj formi te neke bitne strategije i njihov pronalazak u igrama u normalnoj formi. U trećem poglavlju objašnjen je zapis igara u proširenoj formi te kako pronaći Nashovu ravnotežu i čiste strategije u igrama koje su zapisane u proširenoj formi. Također je objašnjen pojam [subgame-perfect] ravnoteže koji je veoma bitan kod igara u proširenoj formi za pronalazak dominantnih strategija.

Četvrto poglavlje se bavi algoritmima za rješavanje stabla pretraživanja. Posebna pažnja je posvećena algoritmu α-β podrezivanje te su navedene prednosti tog algoritma naspram minimax algoritma.

U sklopu rada napisan je program za igru Križić-kružić u programskom jeziku Python. Igru može igrati čovjek protiv čovjeka, čovjek protiv računala ili računalo protiv računala. Računalo pomoću α-β podrezivanja odlučuje o svom sljedećem koraku. U poglavlju 5.2 je opisano što računalo napravi kada se nađe u situaciji u kojoj ne može pobjediti čovjeka.

Da bi došli do te situacije u programu je dozvoljeno da čovjek odabere prvi potez računala.

Cilj rada je opisati na koji način se odabiru strategije u igrama s dva igrača i na koji način računalo može odabrati svoju pobjedničku strategiju.

Poglavlje 1

Normalna forma

Definicija 1.0.1. *Konačna igra s dva igrača u normalnoj formi je uređena trojka* (N, A, u)*, gdje je*

- $N = \{1, 2\}$ *skup od* 2 *igrača;*
- $A = A_1 \times A_2$ *gdje je* A_i*,* $A_i = \{a_1^i, \ldots, a_{n_i}^i\}$*, konačan skup akcija (ili čistih strategija) koje su dostupne igraču* i*,* $i = 1, 2$*. Svaki vektor* $a = (a_1, a_2) \in A$ *nazivamo* ***profil akcija*** *(ili profil čistih strategija);*
- $u = (u_1, u_2)$ *gdje je* $u_i : A \to \mathbb{R}$ *funkcija isplate za igrača* i*,* $i = 1, 2$*.*

[2]

Igra u normalnoj formi[1] se zapisuje pomoću n-dimenzionalne matrice u kojoj redovi predstavljaju moguće akcije prvog igrača, a stupci moguće akcije drugog igrača. Svaka ćelija odgovara nekom od završnih stanja igre. U njima su zapisane korisnosti odgovarajućih završnih stanja za svakog od igrača na način da prvi broj u ćeliji predstavlja korisnost za prvog igrača, a drugi korisnost za drugog igrača.

Radi lakšeg razumjevanja igre u normalnoj formi pogledajmo sljedeći primjer.

Primjer 1.0.2. ***Zatvorenikova dilema***
Dva člana zločinačke organizacije su uhićena. Svaki zatvorenik je smješten u posebnu prostoriju tako da zatvorenici međusobno ne mogu komunicirati. Tužitelji nemaju dovoljno dokaza da osude oba zatvorenika na temelju glavne optužnice, ali se nadaju da će ih moći osuditi na godinu dana na temelju manje optužnice. Tužitelji istodobno ponude svakom zatvoreniku Faustovu nagodbu. Svakom zatvoreniku je ponuđeno da ili izda drugog zatvorenika svjedočeći da je drugi zatvorenik počinio zločin ili surađuje s drugim zatvorenikom šutnjom. Ponude su sljedeće:

[1] Normalna forma se zove još i strateška ili matrična forma.

- *Ako oba zatvorenika izdaju jedan drugog, svaki od njih služi dvije godine u zatvoru*
- *Ako jedan zatvorenik izda drugog, dok drugi šuti, onaj koji je izdao bit će oslobođen, a drugi će biti osuđen na tri godine*
- *Ako oba zatvorenika šute, svaki će biti osuđen na godinu dana u zatvoru*

Bitno je napomenuti da zatvorenici neće biti niti nagrađeni niti kažnjeni za svoju odluku i da njihova odluka neće utjecati na njihov ugled u budućnosti.

Zapis zatvorenikove dileme u normalnoj formi je dan pomoću tablice 1.1.

Prvi zatvorenik \ Drugi zatvorenik	šutnja	izdaja
šutnja	-1,-1	-3,0
izdaja	0,-3	-2,-2

Tablica 1.1: Zatvorenikova dilema prikaza u normalnoj formi

Primjetimo da se igra ne bi promjenila ako -3 zamjenimo s -5 ili ako bi svakoj korisnosti dodali neki cijeli broj. Štoviše, igre sa standardnim korisnostima su neosjetljive na bilo koju pozitivnu afinu transformaciju. Odnosno, svaka korisnost x u igri se može zamjeniti s $ax + b$ za svaka dva realna broja $a > 0$ i b.

1.1 Podjela igara u normalnoj formi prema obliku isplate

Uzimajući u obzir isplate, igre možemo podjeliti na dvije klase, a to su igre sa zajedničkom isplatom i igre sa sumom nula. Kao što samo ime govori, igre sa zajedničkom isplatom su igre u kojoj, za svaki profil akcija, svi igrači imaju istu isplatu. Preciznija definicija je dana u definiciji 1.1.1.

Definicija 1.1.1. ***Igra sa zajedničkom isplatom*** *je igra u kojoj za svaki profil akcija a, $a \in A_1 \times A_2$, i igrače 1 i 2 vrijedi $u_1(a) = u_2(a)$.*

Igre sa zajedničkom isplatom se još zovu igre čiste kordinacije[2] ili timske igre. U navedenim igrama igrači nemaju konfliktne interese. Najveći izazov u igri je koordinirati akciju koja će donijeti maksimalnu korisnost svim igračima.

Radi lakšeg shvaćanja, zamislimo dva vozača kako voze sredinom ceste jedan drugom u susret. U trenu kad se ugledaju odlučuju u koju stranu će skrenuti da bi izbjegli sudar. Ukoliko oba vozača izaberu da će skrenuti lijevo (ili desno) do sudara neće doći, u suprotnom dolazi do sudara. Matrica igre je dana tablicom 1.2.

[2]eng. pure-coordination games

Prvi vozač \ Drugi vozač	lijevo	desno
lijevo	1,1	0,0
desno	0,0	1,1

Tablica 1.2: Igra sa zajedničkom isplatom

Druga klasa su igre sa sumom nula. Općenitiji naziv za njih je *"igre s konstantnom sumom"*. Taj naziv opravdava činjenica da su korisnosti u igri neosjetljive na pozitivne afine transformaicje.

Definicija 1.1.2. *Igra, u normalnoj formi, s dva igrača je igra s* ***konstantnom sumom*** *ako postoji konstanta c takva da za svaki profil strategija* $a \in A_1 \times A_2$ *vrijedi* $u_1(a) + u_2(a) = c$.

Igre sa konstantnom sumom možemo još nazvati i igre čiste konkurencije jer korisnost jednog igrača znači gubitak drugog igrača.

U daljnjem tekstu umjesto igara s konstantnom sumom, promatrat ćemo igre sa sumom nula.

Primjer igre sa sumom nula je igra "Podudaranje novčića". U igri sudjeluju dva igrača. Svaki igrač baca svoj novčić. Kada novčić padne igrači ih uspoređuju. Ako oba novčića padnu na pismo (ili glavu) prvi igrač ih dobiva. U suprotnom drugi igrač dobiva oba novčića. Igra u normalnoj formi je prikazana tablicom 1.3.

Prvi igrač \ Drugi igrač	Glava	Pismo
Glava	1,-1	-1,1
Pismo	-1,1	1,-1

Tablica 1.3: Normalna forma igre "Podudaranje novčića"

U većini slučajeva igre kombiniraju elemente koordinacije i konkurencije. Zatvorenikova dilema je primjer takve igre. Drugi primjer je "Bitka spolova". To je igra u kojoj muž i žena donose odluku o tome koji film će gledati u kinu. U kinu se prikazuju dva filma: "Smrtonosno oružje" (SO) i "Opet ti" (OT). Bračni par bi više volio pogledati jedan film zajedno nego svaki odvojeno, no dok žena želi pogledati "Smrtonosno oružje", muž više preferira film "Opet ti". Matrica isplata je dana tablicom 1.4.

Žena \ Muž	SO	OT
SO	2,1	0,0
OT	0,0	1,2

Tablica 1.4: Normalna forma igre "Bitka spolova"

1.2 Mješovite strategije igre u normalnoj formi

Igrači tokom igre razvijaju strategiju koja će ih dovesti do njihove pobjede na način da u svakom koraku igre odaberu određenu akciju i odigraju ju. Takvu strategiju zvali smo **čista strategija**. Za zapis strategije koristit ćemo već opisani zapis za akciju. Izbor čistih strategija za svakog igrača zvali smo **profil čiste strategije**.

Igrači mogu slijediti i neku drugu, manje očitu strategiju: slučajni odabir nad skupom dostupnih akcija prema nekoj vjerojatnosnoj distribuciji. Takva strategija se naziva mješovita strategija [3].

Definicija 1.2.1. *Neka je* (N, A, u) *igra u normalnoj formi i za bilo koji skup* X *označimo* s $\Pi(X)$ *skup svih vjerojatnosnih distribucija nad* X*. Tada je* $S_i = \Pi(A_i)$ *skup* ***mješovitih strategija*** *igrača* i

Definicija 1.2.2. *Skup profila mješovitih strategija je* $S_1 \times S_2$.

Sa $s_i(a_i)$ označavamo vjerojatnost da će se odigrati akcija a_i u sklopu mješovite strategije s_i.

Definicija 1.2.3. ***Potpora*** [4] *mješovite strategije* s_i *za igrača* i *je skup čistih strategija* $\{a_i \mid s_i(a_i) > 0\}$.

Primjetimo da je čista strategija poseban slučaj mješovite strategije u kojoj je potpora jednočlan skup. Ukoliko je potpora potpuna (točnije, svakoj akciji je pridružena vjerojatnost različita od nule) tada je riječ o potpunoj mješovitoj strategiji.

Isplate kod korištenja mješovitih strategija nije moguće jednostavno pročitati iz matričnog zapisa igre. Ona se zasniva na očekivanoj korisnosti. Intuitivno, prvo računamo vjerojatnost postizanja svakog od završnih stanja danog profila strategija te zatim računamo prosječnu isplatu završnih stanja pomnoženih s vjerojatnošću svakog od završnih stanja. Formalna definicija slijedi.

[3] eng. mixed strategy
[4] eng. support

Definicija 1.2.4. ***Očekivana korisnost mješovite strategije***
Neka je (N, A, u) *igra u normalnoj formi. Očekivana korisnost* u_i *igrača i čiji je profil mješovite strategije* $s = (s_1, s_2)$ *je definirana s*

$$u_i(s) = \sum_{a \in A} u_i(a)(s_1(a_1) \cdot s_2(a_2)) \tag{1.1}$$

Poglavlje 2

Nashova ravnoteža

Promatramo igru u kojoj sudjeluju 2 igrača. Neka je $s = (s_1, s_2)$ profil strategija. S s_{-i} ćemo označiti profil strategija s bez strategije igrača i. Točnije, ako je $i = 1$, tada je $s_{-i} = (s_2)$. Analogno obrnuto. Vidimo da vrijedi $s = (s_i, s_{-i})$. Ako igrači različiti od i (označavamo ih s $-i$) odluče igrati s_{-i}, maksimizirajući igrač i će biti suočen s problem pronalaska svog najboljeg odgovora.

Definicija 2.0.5. ***Najbolji odgovor*** *igrača i na profil strategija s_{-i} je mješovita strategija $s_i^* \in S_i$ gdje je $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ za sve strategije $s_i \in S_i$.*

Najbolji odgovor nije nužno i jedinstven. Štoviše, osim u posebnim slučajevima gdje postoji samo jedan najbolji odgovor koji je čista strategija, broj najboljih odgovora je beskonačan. Kad potpora najboljeg odgovora s^* uključuje dvije ili više akcija, igrač mora biti indiferentan prema tim akcijama jer bi inače igrač htio smanjiti vjerojatnost igranja barem jedne akcije na nulu, a to nas vodi prema čistim strategijama. Svaka kombinacija tih akcija mora također biti najbolji odgovor, a ne samo određena kombinacija. Slično, ako postoje dvije čiste strategije koje su zasebno najbolji odgovor, tada je i kombinacija tih dviju strategija također najbolji odgovor.

Tokom igranja igre igrači ne znaju koje strategije će suparnici odabrati pa, u općenitom smislu, pojam najbolji odgovor ne određuje interesantni skup završnih stanja. No, možemo ga iskoristiti pri definiranju Nashove ravnoteže [1].

Definicija 2.0.6. *Profil strategija $s = (s_1, s_2)$ je* ***Nashova ravnoteža*** *ako, za sve igrače i, strategija s_i je najbolji odgovor na strategiju s_{-i}.*

Možemo reći da je Nashova ravnoteža stabilan profil strategija jer će igrači, ukoliko saznaju suparnikove strategije, i dalje ostati pri svojoj strategiji.

[1] Nashova ravnoteža je dobila ime po John Forbes Nash, Jr.-u, američkom matematičaru koji se bavio teorijom igara, diferencijalnom geometrijom i parcijalnim diferencijalnim jednadžbama.

Ovisno o tome da li je strategija svakog igrača jedinstven najbolji odgovor, Nashovu ravnotežu možemo podjeliti na strogu i slabu.

Definicija 2.0.7. *Profil strategija $s = (s_1, s_2)$ je **stroga Nashova ravnoteža** ako, za oba igrača i i za sve strategije $s_i^{'} \neq s_i$, vrijedi $u_i(s_i, s_{-i}) > u_i(s_i^{'}, s_{-i})$.*

Definicija 2.0.8. *Profil strategija $s = (s_1, s_2)$ je **slaba Nashova ravnoteža** ako, za oba igrača i i za sve strategije $s_i^{'} \neq s_i$, vrijedi $u_i(s_i, s_{-i}) \geq u_i(s_i^{'}, s_{-i})$ i s nije stroga Nashova ravnoteža.*

Nashova ravnoteža mješovitih strategija je uvijek slaba, dok Nashova ravnoteža čistih strategija može biti oboje.

2.1 Pronalazak Nashove ravnoteže

Promotrimo igru "Bitka spolova" te pronađimo Nashovu ravnotežu u njoj. Pretpostavimo da je muž odabrao film SO, tada žena također odabire film SO jer $u_{žena}(SO, SO) = 2 > 0 = u_{žena}(OT, SO)$. Ukoliko je muž odabrao film OT, tada žena također odabire film OT jer $u_{žena}(OT, OT) = 1 > 0 = u_{žena}(SO, OT)$. Ako promatramo što bi muž odabrao s obzirom na ženin odabir, dobijemo iste strategije. Iz toga lako vidimo da igra ima dvije Nashove ravnoteže ukoliko koristimo čiste strategije (pogledaj tablicu 2.1).

Žena \ Muž	SO	OT
SO	**2,1**	0,0
OT	0,0	**1,2**

Tablica 2.1: Nashova ravnoteža čistih strategija u igri "Bitka spolova"

Što se dogodi ako koristimo mješovite strategije? Pretpostavimo da oba igrača biraju film na slučajan način te da je muževa strategija odabir filma SO s vjerojatnošću p te OT s vjerojatnošću $1 - p$. Žena također bira između dva filma. Ona mora biti indiferentna prema odabiru uzimajući u obzir muževu strategiju (u suprotnom, žena prelazi na čistu strategiju). Dobili smo sljedeće jednadžbe:

$$U_{žena}(SO) = U_{žena}(OT)$$
$$2 \cdot p + 0 \cdot (1 - p) = 0 \cdot p + 1 \cdot (1 - p)$$
$$p = \frac{1}{3}$$

Dobili smo, da bi žena bila indiferentna prema svojim akcijama, muž mora odabrati SO s vjerojatnošću $\frac{1}{3}$ i OT s vjerojatnošću $\frac{2}{3}$. Muž također koristi mješovitu strategiju pa i on mora biti indiferentan između svojih akcija. Stoga analogno dobijemo da žena mora odabrati SO s vjerojatnošću $\frac{2}{3}$ i OT s vjerojatnošću $\frac{1}{3}$ da bi muž bio indiferentan prema svojim akcijama. Jer oba igrača igraju s ciljem da drugi igrač bude indiferentan prema svojim akcijama, oba igrača daju najbolji odgovor na akcije drugog igrača. Vidimo da odabir mješovitih strategija daje iste dvije Nashove ravnoteže. Očekivana isplata za oba igrača je $\frac{2}{3}$.

Teorem 2.1.1. ***Nash, 1951***
Svaka igra s konačnim brojem igrača i profilom akcija ima barem jednu Nashovu ravnotežu.

Nashov teorem ovisi o dostupnosti mješovitih strategija igračima. Kod mnogih igara samo mješovite strategije imaju ravnotežu. Primjer takve igre je "Podudaranje novčića". Lako uočavamo da čiste strategije u igri "Podudaranje novčića" nemaju ravnotežu. Ravnoteža je da svaki igrač izabire jednu od dvije akcije s vjerojatnošću od $\frac{1}{2}$.

2.2 Maxmin i minmax strategije

Neka u igri s konstantnom sumom sudjeluju 2 igrača. Maxmin strategija igrača i, $i \in \{1, 2\}$, je strategija koja maksimizira najgoru isplatu igraču i u situaciji u kojoj svi drugi igrači igraju strategiju koja uzrokuje najveću štetu igraču i. Maxmin strategija ne mora biti jedinstvena.

Definicija 2.2.1. ***Maxmin strategija*** *igrača i je arg $max_{s_i}\, min_{s_{-i}}\, u_i(s_i, s_{-i})$.*
Maxmin vrijednost [2] *igrača i je $max_{s_i}\, min_{s_{-1}}\, u_i(s_i, s_{-i})$.*

Strategija suprotna maxmin strategiji je minmax strategija. Minmax strategija igrača i protiv igrača $-i$ je strategija koja zadržava maksimalnu isplatu igrača $-i$ na minimumu.

Definicija 2.2.2. *U igri s dva igrača,* ***minmax strategija*** *igrača i protiv igrača $-i$ je arg $min_{s_i}\, max_{s_{-i}}\, u_{-i}(s_i, s_{-i})$.*
Minmax vrijednost *za igrača $-i$ je $min_{s_i}\, max_{s_{-i}}\, u_{-i}(s_i, s_{-i})$.*

Jer maxmin i minmax strategije jednog igrača ne ovise o odabranim strategijama drugog igrača, one na veoma jednostavan način daju koncept rješenja. Profil mješovitih strategija $s = (s_1, s_2)$ je **profil maxmin strategija** igre ako je s_1 maxmin strategija prvog igrača, a s_2 maxmin strategija drugog igrača. Analogno definiramo profil minmax strategija.

[2] drugi naziv je **razina sigurnosti**, eng. security level

Teorem 2.2.3. ***Minimax teorem (von Neumann, 1928)***
U bilo kojoj konačnoj igri s dva igrača i sumom nula postoji Nashova ravnoteža. U svakoj Nashovoj ravnoteži svaki igrač prima isplatu koja je jednaka njegovoj maxmin i minmax vrijednosti.

Iz teorema 2.2.3 lako zaključujemo da je, u igrama s dva igrača, igračeva minmax vrijednost jednaka njegovoj maxmin vrijednosti i igračeva minmax strategija je jednaka njegovoj maxmin strategiji. Prema dogovoru, maxmin vrijednost prvog igrača nazivamo *vrijednost igre*.

Svaki profil maxmin strategija je Nashova ravnoteža.

2.3 Dominantne strategije

Svaki igrač želi pobijediti u igri u kojoj sudjeluje. Da bi mogao pobijediti on pažljivo razmatra pojedine strategije i odabire onu koja ima najbolju isplatu za njega. Moguće je da se jedna strategija ističe kao najbolja, točnije jedna dominira nad svim ostalima, ili da više strategija imaju istu isplatu. U definiciji 2.3.1 definiramo tri razine dominacije u strategijama.

Definicija 2.3.1. *Neka su s_i i s_i' dvije strategije igrača i i neka je S_{-i} skup svih profila strategija preostalih igrača. Tada*

- s_i ***strogo dominira*** *strategijom s_i' ako za svaki $s_{-i} \in S_{-i}$ vrijedi $u_i(s_i, s_{-i}) > u_i(s_i', s_{-i})$*
- s_i ***slabo dominira*** *strategijom s_i' ako za svaki $s_{-i} \in S_{-i}$ vrijedi $u_i(s_i, s_{-i}) \geq u_i(s_i', s_{-i})$ i postoji barem jedna strategija $s_{-i} \in S_{-i}$ za koju vrijedi $u_i(s_i, s_{-i}) > u_i(s_i', s_{-i})$*
- s_i ***veoma slabo dominira*** *strategijom s_i' ako za svaki $s_{-i} \in S_{-i}$ vrijedi $u_i(s_i, s_{-i}) \geq u_i(s_i', s_{-i})$*

Ako postoji strategija koja dominira svim ostalim strategijama, za tu strategiju kažemo da je *dominantna*.

Definicija 2.3.2. *Igračeva strategija je strogo (odnosno slabo ili veoma slabo) dominantna ako strogo (odnosno slabo ili veoma slabo) dominira svim ostalim strategijama tog igrača.*

Očito je da je profil strategija (s_1, s_2), u kojem je s_i dominantna strategija igrača i, Nashova ravnoteža. Takav profil strategija formira *ravnotežu u strogo(odnosno slabo ili veoma slabo) dominantnim strategijama*. Ravnoteža u strogo dominantnim strategijama je jedinstvena Nashova ravnoteža.

Za primjer pogledajmo igru "Zatvorenikova dilema". Vidimo da, ukoliko prvi zatvorenik odabere šutnju, tada će drugi zatvorenik odabrati izdaju. Ako prvi zatvorenik odabere

izdaju, drugi zatvorenik će odabrati izdaju. Analogno dobijemo za prvog zatvorenika. Iz tablice 2.2 vidimo da "Zatvorenikova dilema" ima jednu strogo dominantnu strategiju i da je ona upravo i Nashova ravnoteža[3].

Prvi zatvorenik \ Drugi zatvorenik	šutnja	izdaja
šutnja	-1,-1	-3,**0**
izdaja	**0**,-3	**-2**,**-2**

Tablica 2.2: Dominantne strategije u zatvorenikovoj dilemi

Dominantne strategije igraju veliku ulogu u teoriji igara, no igre s dominantnim strategijama su rijetke. Mnogo češće su igre s dominiranim strategijama.

Definicija 2.3.3. *Strategija s_i igrača i je* ***strogo (odnosno slabo ili veoma slabo) dominirana*** *ako neka druga njegova strategija s_i' strogo (odnosno slabo ili veoma slabo) dominira nad s_i.*

[3]U kolovozu 2013. godine provedena je anketa među zatvorenicama ženskog zatvora. Rezultati te ankete su uspoređeni s anketom koja je provedena nad studentima. Zanimljivo je to da je 55% zatvorenica odabralo šutnju umjesto izdaje, dok je među studentima taj postotak bio manji, svega 37%. Rezultati u slučaju kada se igra izvodi sekvencijalno su nešto drugačiji nego u slučaju simultane igre. U sekvencijalnoj igri u prvom krugu postotak zatvorenica koje su odabrale šutnju nije se promijenio, on je i dalje 55%, dok je postotak studenata koji bi odabrali šutnju znatno porastao. čak 63% studenata bi odabralo šutnju. U drugom krugu, zatvorenicama i studentima su pokazani odgovori njihovog slučajno odabranog partnera. Očekivano, izdaja je uzvraćena izdajom osim u slučaju jedne zatvorenice. Također, suradnju su uglavnom uzvraćene.[3]

Poglavlje 3

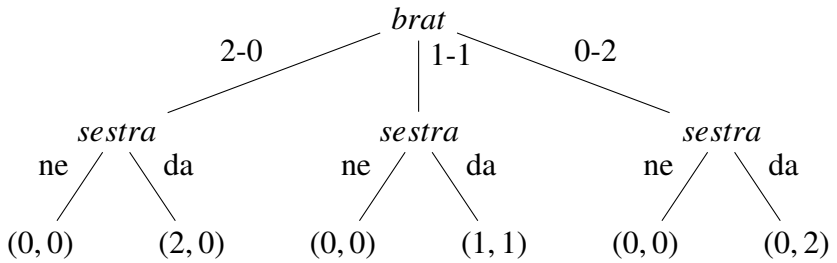
Igre u proširenoj formi

Intuitivno, igra sa savršenim informacijama u proširenoj formi (ili jednostavnije: igra sa savršenim informacijama) je stablo u kojem svaki čvor predstavlja izbor jednog od igrača, bridovi predstavljaju moguće akcije, a listovi završna stanja nad kojim svaki igrač ima funkciju korisnosti. Formalna definicija slijedi.

Definicija 3.0.4. ***Igra sa savršenim informacijama***
Konačna igra sa savršenim informacijama (u proširenoj formi) je uređena osmorka $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ *gdje je:*

- $N = \{1, 2\}$ *skup od dva igrača;*
- A *je skup akcija;*
- H *je skup čvorova koji nisu listovi stabla;*
- Z *je skup listova stabla i vrijedi* $H \cap Z = \emptyset$*;*
- $\chi : H \mapsto 2^A$ *je funkcija akcija koja svakom čvoru izbora pridružuje skup mogućih akcija;*
- $\rho : H \mapsto N$ *je funkcija igrača koja svakom čvoru koji nije list stabla pridružuje igrača* $i \in N$ *koji odabire akciju u tom čvoru;*
- $\sigma : H \times A \mapsto H \cup Z$ *je funkcija sljedbenika koja mapira čvor izbora i akciju novom čvoru izbora ili listu stabla tako da za svake* $h_1, h_2 \in H$ *i* $a_1, a_2 \in A$*, ako vrijedi* $\sigma(h_1, a_1) = \sigma(h_2, a_2)$ *tada* $h_1 = h_2$ *i* $a_1 = a_2$*;*
- $u = (u_1, u_2)$ *gdje je* $u_i : Z \mapsto \mathbb{R}$ *funkcija korisnosti za igrača i na listovima stabla* Z*.*

Promatrajući čvor izbora na stablu, lako pronalazimo njegovu *povijest*. To je upravo niz čvorova koji vodi od korijena stabla do promatranog čvora. Lako definiramo i *potomke* čvora h, a to su svi čvorovi, uključujući i listove, podstabla čiji je korijen h.



Slika 3.1: Proširena forma igre dijeljenja

3.1 Strategije i Nashova ravnoteža

Definicija 3.1.1. ***Čista strategija***
Neka je $G = (N, A, H, Z, \chi, \rho, \sigma, u)$ *igra sa savršenim informacijama u proširenoj formi. Čiste strategije igrača i su kartezijev produkt* $\prod_{h \in H, \rho(h)=i} \chi(h)$.

Primjetimo da igračeva strategija zahtjeva odluku za svaki čvor izbora, bez obzira da li je moguće doći do nekog čvora tokom igre.

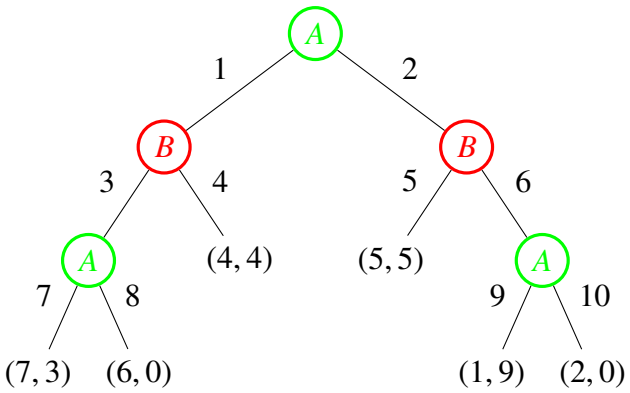
Radi lakšeg razumjevanja promotrimo igru "Igra dijeljenja". U igri sudjeluju brat i sestra koji slijede protokol za dijeljenje dva indentična nevidljiva poklona njihovih roditelja. U prvom koraku brat predlaže podjelu poklona. On može zadržati oba poklona ili sestri dati da zadrži oba poklona ili dati svakom po jedan poklon. Zatim sestra odabira da li će prihvatiti ili odbiti bratovu podjelu. Ako sestra prihvati podjelu, svatko dobiva poklon na način na koji su se dogovorili, a ako sestra odbije, nijedno ne dobiva poklon. Bitna pretpostavka je da brat i sestra oba poklona jednako cijene. Zapis igre u stablu možemo pogledati na slici 3.1.

Promatrajući stablo igre vidimo da brat ima tri čiste strategija, a sestra ima osam. Strategije su sljedeće: $S_1 = \{2-0, 1-1, 0-2\}$
$S_2 = \{(da, da, da), (da, da, ne), (da, ne, da), (ne, da, da), (da, ne, ne), (ne, da, ne), (ne, ne, da), (ne, ne, ne)\}$

Promotrimo igru prikazanu na slici 3.2.

Da bi definirali strategije za igru na slici 3.2, svaki igrač mora odabrati akciju za svakog od svojih čvorova izbora. Sa S_1 ćemo označiti strategije igrača A, a sa S_2 strategije igrača B. Strategije su sljedeće:
$S_1 = (1, 7, 9), (1, 8, 9), (1, 7, 10), (1, 8, 10), (2, 7, 9), (2, 8, 9), (2, 7, 10), (2, 8, 10)$ i
$S_2 = (3, 5), (3, 6), (4, 5), (4, 6)$.

Primjetimo da, ako igrač A prvo odabere 2, on neće dobiti priliku da bira između 7 i 8. Unatoč tome, strategije koje obuhvaćaju takve akcije moraju se definirati.



Slika 3.2: Igra sa savršenim informacijama u proširenoj formi

Najbolji odgovor i Nashova ravnoteža igre je jednak bez obzira da li je igra prikazana u proširenoj ili normalnoj formi. Slijedećim primjerom ćemo pokazati kako svaku igru u proširenoj formi možemo prikazati u normalnoj formi. Tablicom 3.1 je prikazana normalna forma igre na slici 3.2.

	(3,5)	(3,6)	(4,5)	(4,6)
(1,7,9)	7,3	7,3	4,4	4,4
(1,8,9)	6,10	6,10	4,4	4,4
(1,7,10)	7,3	7,3	4,4	4,4
(1,8,10)	6,10	6,10	4,4	4,4
(2,7,9)	5,5	1,9	5,5	1,9
(2,8,9)	5,5	1,9	5,5	1,9
(2,7,10)	5,5	2,0	5,5	2,0
(2,8,10)	5,5	2,0	5,5	2,0

Tablica 3.1: Prikaz igre sa slike 3.2 u normalnoj formi

Za svaku igru sa savršenim informacijama postoji odgovarajuća igra u normalnoj formi, ali struktura igre u proširenoj formi može rezultirati određenim viškovima kad je ista igra prikazana u normalnoj formi. Na primjer, na slici 3.2 postoji 32 različita završna stanja, dok u tablici 3.1 ih ima samo 6. Lako je primjetiti da, iako uvijek postoji transformacija iz proširene u normalnu formu, ona može rezultirati veoma velikim matricama.

Obrnuta transformacija ne postoji uvijek.

Teorem 3.1.2 daje razlog zašto smo se koncentrirali na čiste strategije i na čistu Nashovu ravnotežu u igrama u proširenoj formi.

Teorem 3.1.2. *Svaka konačna igra sa savršenim informacijama u proširenoj formi ima Nashovu ravnotežu čistih strategija.*

U proširenoj formi nema potrebe za mješovitim strategijama, odnosno za slučajnim odabirom jer igrači izmjenjuju poteze i svaki igrač može lako vidjeti što se dogodilo prije njegovog poteza.

3.2 *Subgame-perfect* ravnoteža

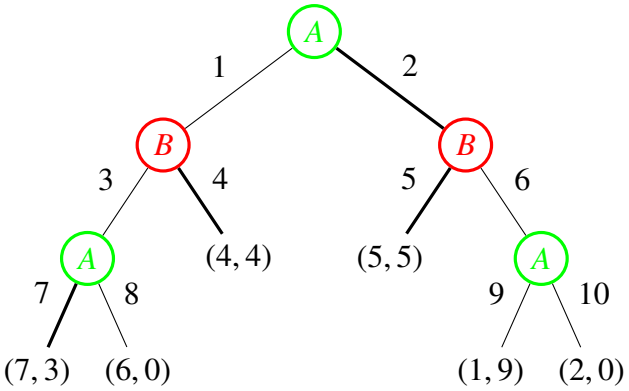
Iz primjera sa slike 3.2 i tablice 3.1 vidimo da Nashova ravnoteža može biti preslab pojam za proširenu formu. Ako pogledamo tablicu 3.2 vidimo da postoji četiri Nashove ravnoteže čistih strategija, a to su $\{(1,7,9),(4,6)\}$, $\{(1,7,10),(4,6)\}$, $\{(2,7,10),(4,5)\}$ i $\{(2,8,10),(4,5)\}$.

	(3,5)	(3,6)	(4,5)	(4,6)
(1,7,9)	**7**,3	**7**,3	4,**4**	**4**,**4**
(1,8,9)	6,**10**	6,**10**	4,4	**4**,4
(1,7,10)	**7**,3	**7**,3	4,**4**	**4**,**4**
(1,8,10)	6,**10**	6,**10**	4,4	**4**,4
(2,7,9)	5,5	1,**9**	**5**,5	1,**9**
(2,8,9)	5,5	1,**9**	**5**,5	1,**9**
(2,7,10)	5,**5**	2,0	**5**,**5**	2,0
(2,8,10)	5,**5**	2,0	**5**,**5**	2,0

Tablica 3.2: Nashove ravnoteže igre sa slike 3.2

Promotrimo strategije $\{(2,7,10),(4,5)\}$ i $\{(2,8,10),(4,5)\}$. Pokažimo zašto strategija $\{(2,8,10),(4,5)\}$ nije prava ravnoteža. U toj strategiji igrač A je rekao da će izabrati akciju 8 umjesto 7 iako je to za njega manje povoljno. Takvo ponašanje nazivamo **prijetnja**. Ako B povjeruje prijetnji, onda je njemu bolje da izabere 3 umjesto 4, a ako B izabere 3 onda je za igrača A bolje da izabere 1 umjesto 2 pa zaključujemo da strategija $\{(2,8,10),(4,5)\}$ nije u ravnoteži.

Ako je igrač A izabrao 7, a ne 8, što je i logičan potez za njega, tada će igrač B rađe izabrati 4 umjesto 3 jer mu ta akcija daje veću korisnost. U trećem čvoru izbora igrača A logičnije je da on izabere 10 umjesto 9 pa samim time igrač B izabire 5 umjesto 6. Na kraju igrač A izabire 2 umjesto 1. Odabir akcija je naglašen na slici 3.3. Ovime smo dobili strategiju $\{(2,7,10),(4,5)\}$ koja je u ravnoteži.



Slika 3.3: Igra sa savršenim informacijama u proširenoj formi

Da bi dali formalni razlog zašto ravnoteža $\{(2, 8, 10), (4, 5)\}$ nije zadovoljavajuća i da bi dali profinjenje koncepta ravnoteže koje izbjegava ovaj problem definirat ćemo pojam podigre[1].

Definicija 3.2.1. ***Podigra***
Neka je dana igra G sa savršenim informacija u proširenoj formi. Podigra igre G ukorijenjena u čvoru h je ograničenje igre G na potomke od čvora h. Skup podigra igre G se sastoji od svih podigra igre G koji su ukorijenjeni u nekom čvoru igre G.

Sada možemo definirati profinjenje Nashove ravnoteže u igri sa savršenim informacijama u proširenoj formi, točnije *subgame-perfect* ravnotežu.

Definicija 3.2.2. *Subgame-perfect ravnoteža igre G su svi profili strategija s takvi da za svaku podigru G' igre G, restrikcija od s na G' je Nashova ravnoteža igre G'.*

Jer je G sam svoja podigra, tada je svaka *subgame-perfect* ravnoteža također i Nashova ravnoteža. Također vrijedi da svaka igra sa savršenim informacijama u proširenoj formi ima barem jednu *subgame-perfect* ravnotežu.

Ova definicija izbjegava nestvarne prijetnje. Pronađimo sada *subgame-perfect* ravnoteže u igri na slici 3.2. Ako promatramo podigru čiji je korijen u drugom čvoru izbora igrača A vidimo da je jedinstvena Nashova ravnoteža za igrača A da odabere 7. U trećem čvoru izbora igrača A jedinstvena Nashova ravnoteža je akcija 10. S obzirom na odabrane akcije igrača A, igrač B da bi bio u ravnoteži bira strategiju $\{(4, 5)\}$. Na kraju igrač A bira akciju 2. Vidimo da je ova igra ima samo jednu *subgame-perfect* ravnotežu i to je upravo $\{(2, 7, 10), (4, 5)\}$.

[1] eng. subgame

Poglavlje 4

Algoritmi za rješavanje stabla pretraživanja

Nakon što smo konstruirali stablo pretraživanja za neku igru, iz njega je potrebno naći strategiju za maksimizirajućeg igrača. Postoje razni algoritmi koji pronalaze tu strategiju. Kojeg od njih ćemo odabrati ovisi o vrsti igre i veličini stabla pretraživanja. Algoritme djelimo na deterministčke i slučajne. Podjele su objašnjene u sljedećih podpoglavljima

4.1 Deterministički algoritmi

Pretpostavimo da za neku igru I s dva igrača imamo kompletno stablo T. Tada je moguće naći niz poteza kojeg mogu sljediti jedan od igrača i koji će mu garantirati pobjedu ili izjednačenje. Taj niz lako nalazimo nekim od deterministčkih algoritama. Deterministčke algoritme djelimo na proof-number i na conspiracy-number search. Glavna razlika između njih je ta što proof-number search određuje pravu vrijednost korijena stabla, dok conspiracy-number pretraživanje pretražuje stablo skroz dok nije siguran da se minimalna i maksimalna vrijednost korijena neće promjeniti. Neki od algoritama koji spadaju u conspiracy-number pretraživanje su minimax i α - β podrezivanje. Svi deterministički algoritmi se grade na principu indukcije unazad (ili retrogradna analiza). Neke od igara u kojima koristimo deterministčke algoritme su šah, križić-kružić, dama,...

Ponekad nije potrebno koristiti cijelo stablo igre da bi pronašli najbolju strategiju nego se možemo koncentrirati na neka podstabla. Podstabla koja koristimo u odlučivanju zovemo stabla odlučivanja. Veličine stabla odlučivanja se koriste za mjerenje složenosti igre.

Neke mjere složenosti igre prema [1] su:

- **Veličina stabla igre:** broj listova stabla igre kojoj je korijen početno stanje igre. Još kažemo da je to ukupan broj mogućih igara.

- **Složenost odlučivanja:** broj listova u najmanjem stablu odlučivanja pomoću kojeg utvrđujemo vrijednost korijena stabla u kojem je početno stanje igre.

- **Složenost stabla igre:** broj listova u najmanjem potpunom stablu odlučivanja pomoću kojeg utvrđujemo vrijednost korijena stabla u kojem je početno stanje igre. Potpuno stablo je stablo koje sadrži sve čvorove na svakoj dubini.

Proof-number search

Proof-number pretraživanje je deterministički algoritam pretraživanja stabla igre koje je izumio Victor Allis. Koristi se uglavnom za pronalazak rješenja neke igre ili nekih podciljeva tokom igre. On je zasnovan na conspiracy-number pretraživanju. Koristi i-ili stablo pretraživanja, a ishod mu je binaran. Maksimizirajući čvorovi postaju "*ili*" čvorovi, a minimizirajući postaju "*i*" čvorovi. Listovi mogu imati tri vrijednosti: dobitak i gubitak ili nepoznatu vrijednost.

Da bi lakše opisali algoritam uvedimo sljedeće definicije

Definicija 4.1.1. ***Granični čvor*** *je čvor čija djeca su listovi stabla.*

Definicija 4.1.2. ***Terminalni čvor*** *je čvor koji poprima neku numeričku vrijednost. Tu numeričku vrijednost nazivamo* ***broj dokaza****, odnosno* ***broj pobijanja****, ovisno o tome da li želimo pobiti ili dokazati stablo.*

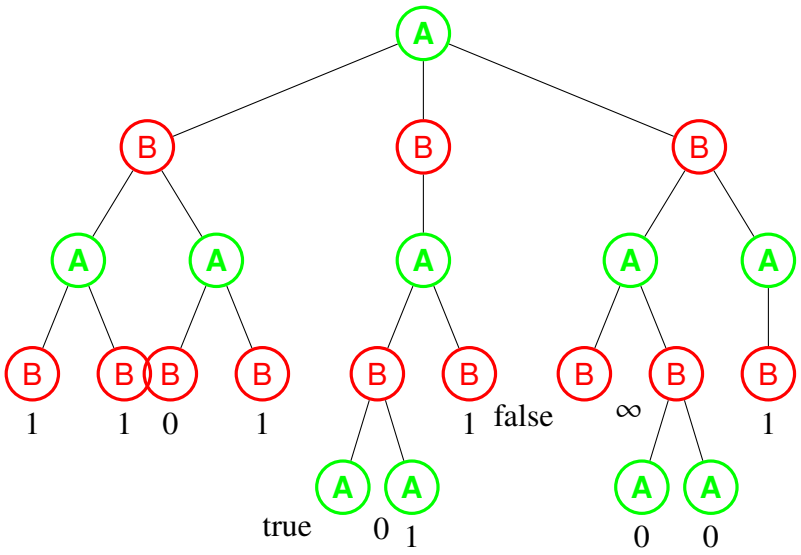
Definicija 4.1.3. ***Čvor dokaza*** *je čvor čija vrijednost je istina, dok je* ***čvor pobijanja*** *čvor čija je vrijednost laž. Oba čvora imaju broj dokaza (odnosno broj pobijanja).*

Neka je V čvor maksimizirajućeg igrača. To znači da je V upravo "*ili*" čvor. Čvor V je istinit ako je istinito barem jedno njegovo dijete, a ako neko njegovo dijete ima vrijednost ∞ tada i čvor V ima istu vrijednost. U svim ostalim slučajevim V je laž. Ukoliko je V čvor minimizirajućeg igrača, on je laž ako je barem jedno njegovo dijete laž. Ukoliko on nema djece sa vrijednosti laž, a ima barem jedno dijete s vrijednosti ∞ on poprima vrijednost ∞. U ostalim slučajevima je čvor V istinit.

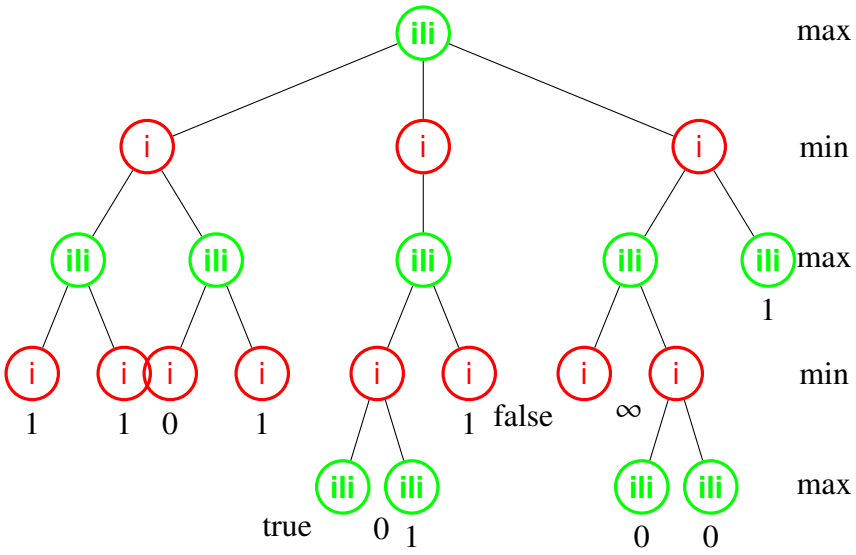
Primjer 4.1.4. *Prikažimo pronalazak broja dokaza i pobijanja na stablu 4.1:*

1. korak: Na početku moramo podjeliti čvorove na i-ili čvorove. Kao što je već rečeno, čvorovi maksimizirajućeg igrača su "*ili*" čvorovi, a čvorovi minimizirajućeg igrača su "*i*" čvorovi (pogledaj sliku 4.2).

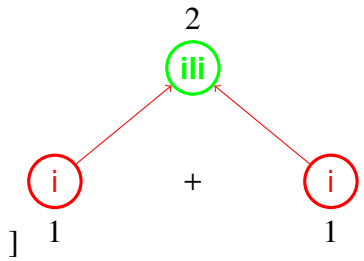
2. korak: Promatrajmo korijen stabla i označimo ga s K. On pripada maksimizirajućem igraču. Prvo nalazimo broj dokaza korijena stabla. Da bi ga pronašli pogledajmo njegove granične čvorove. Ako je granični čvor "*ili*" čvor, tada broj dokaza graničnog čvora dobivamo zbrajanjem brojeva dokaza njegove djece. (pogledaj sliku 4.3)



Slika 4.1: Primjer stabla za pronalaženje broja dokaza i pobijanja

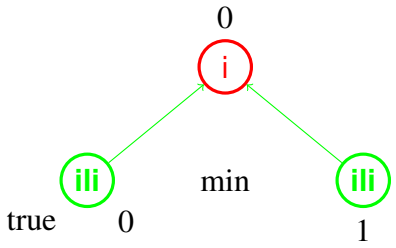


Slika 4.2: 1. korak pronalaženja broja dokaza i pobijanja



Slika 4.3: Broj dokaza "*ili*" čvora

Ako je granični čvor "*i*" čvor, tada je broj dokaza graničnog čvora jednak minimalnom broju dokaza njegove djece (pogledaj sliku 4.4).



Slika 4.4: Broj dokaza "*i*" čvora.

Na kraju dobijemo stablo prikazano na slici 4.5 s odogovarajućim brojevima dokaza:

Vidimo da je broj dokaza korijena stabla jednak 3.

3. korak: Nađimo sada broj pobijanja korijena stabla. Njega dobijemo analogno kao i broj dokaza tako da zamjenimo postupke za "*i*" i "*ili*" čvorove. Točnije, broj pobijanja "*ili*" čvora je jednak najmanjem broju pobijanja njegove djece (pogledaj sliku 4.6).

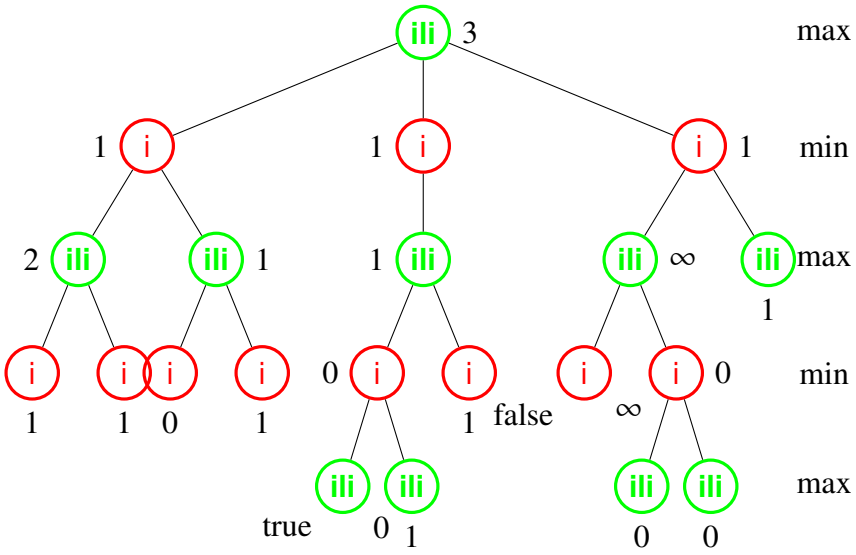
Dok je broj pobijanja "*i*" čvora jednak sumi brojeva pobijanja njegove djece (pogledaj sliku 4.7).

Stablo s pridruženim brojevima pobijanja je prikazano na slici 4.8:

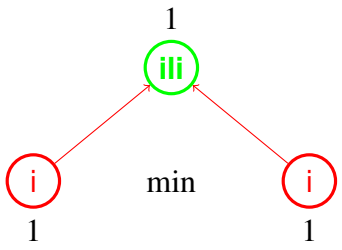
Iz slike vidimo da je i broj pobijanja jednak 3. Ti brojevi nisu nužno uvijek jednaki.

Victor Allis je koristio gornji algoritam pri rješavanju i analiziranju igre "Spoji četiri točke". Charles Elkan je primjenio algoritam kod automatskog dokazivanja teorema. Johathan Schaeffer je koristio algoritam 2007. kod rješavanja igre "Dama".

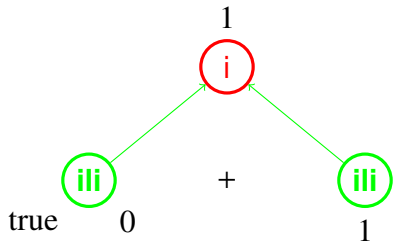
Najveći problem kod proof-number pretraživanja je problem s memorijom. Algoritam funkcionira po principu "najbolji prvi" (odnosno kao best-first search) pa kompletno stablo



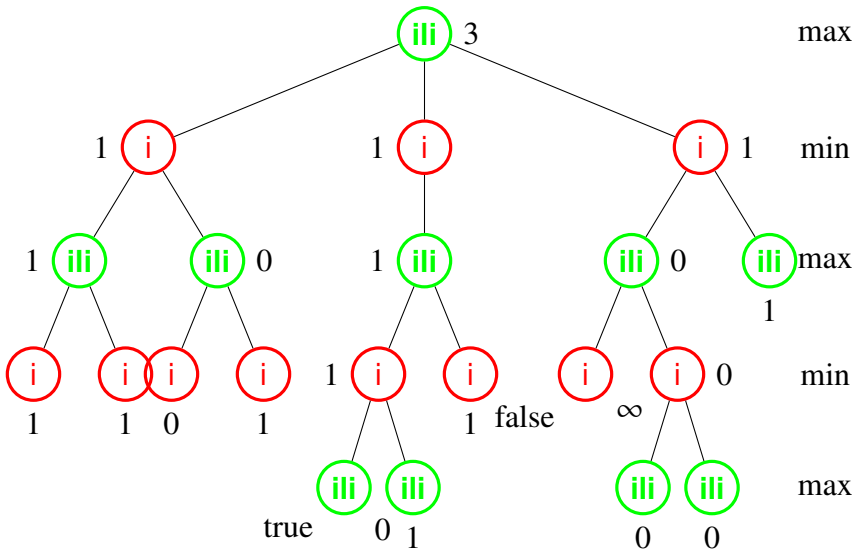
Slika 4.5: Stablo s pridruženim brojevima dokaza



Slika 4.6: Broj pobijanja "*ili*" čvora



Slika 4.7: Broj pobijanja "*i*" čvora



Slika 4.8: Stablo s pridruženim brojevima pobijanja

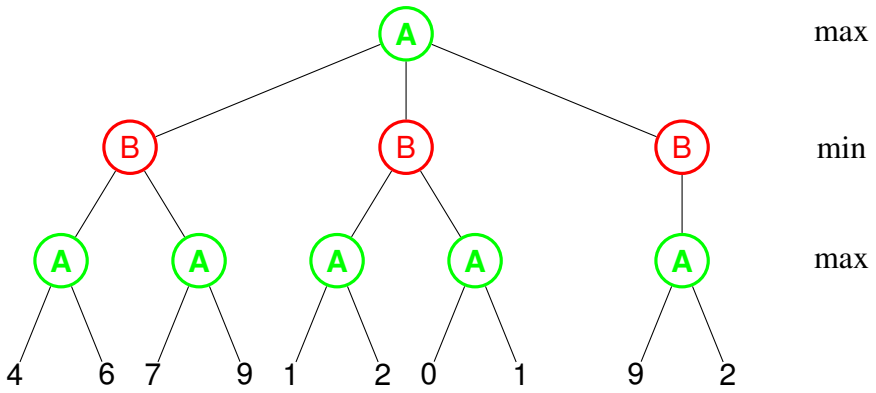
pretraživanja mora biti sačuvano u memoriji. U većini slučajeva dođe do manjka memorije. Taj problem se može donekle riješiti tako da se ne pamte već dokazana podstabla. Postoje još neki načini da se sačuva memorija, kao npr. brisanje čvorova koji bi mogli najmanje utjecati na dokaz igre.

Algoritam ima prednost kod igara u kojima stablo pretraživanja nije uniformno te pogotovo ako postoje potezi koji se moraju odigrati. Najbolji primjer primjene je u "Gubitničkom šahu". To je posebna vrsta šaha u kojem je cilj izgubiti što više figura. (vidi [4])

Minimax

Za pretraživanje stabla igre u igrama sa potpunim informacijama gdje znamo sve korake koje suparnik može napraviti možemo koristiti minimax algoritam. On je veoma jednostavan za razumjevanje i veoma je sličan načinu kako čovjek razmišlja dok igra. Točnije, algoritam se zasniva na principu: "Ako ja povučem ovaj potez, suparnik bi mogao povući taj potez" i na taj način traži strategiju koja je najbolja za maksimizirajućeg igrača. Na sljedećem primjeru ću detaljnije objasniti minimax algoritam. Pogledajte sljedeće poglavlje da bi lakše uočili razliku između minimax i alfa-beta pretraživanja.

Primjer 4.1.5. *Napravimo minimax na stablu prikazanom na slici 4.9:*

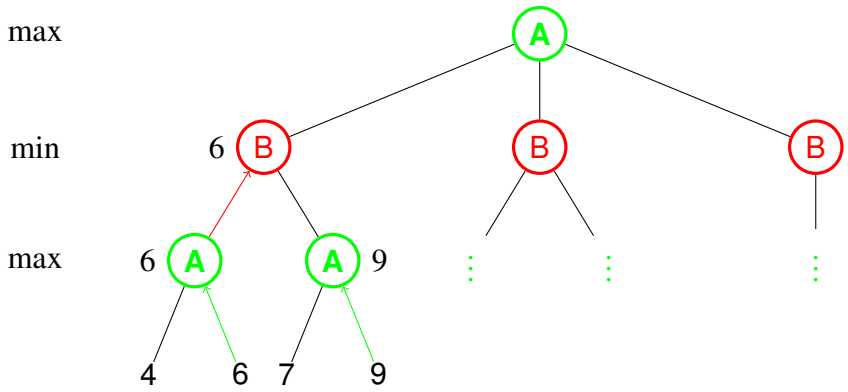


Slika 4.9: Stablo na kojem će biti objašnjen minimax algoritam

1. korak: Pretraživanje kreće od korijena stabla. Čvor korijena stabla pripada igraču *A* koji je maksimizirajući igrač. Vrijednost koju poprima je jednaka maksimalnoj isplati njegove djece pa, da bi ju našli, prvo pogledajmo koje vrijednosti poprimaju njegova djeca.

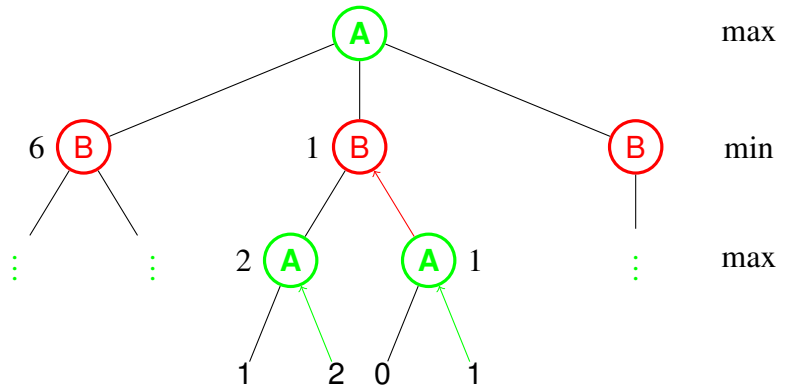
2. korak: Proširimo pretragu na prvo lijevo dijete korijena stabla. Taj čvor pripada minimizirajućegm igraču *B*. On poprima minimalnu vrijednost svoje djece pa da bi saznali njegovu vrijednost proširimo pretragu na njegovu djecu.

3. korak: Došli smo do djece igrača *B*. Njegovo lijevo dijete ima isplate 4 i 6, a desno 7 i 9. Pošto su njegova djeca čvorovi maksimizirajućeg igrača, oni poprimaju veću od dvije isplate. Tako lijevo dijete poprima vrijednost 6, a desno 9. Čvor *B* poprima manju od dvije vrijednosti, a to je upravo 6. (pogledaj sliku 4.10)



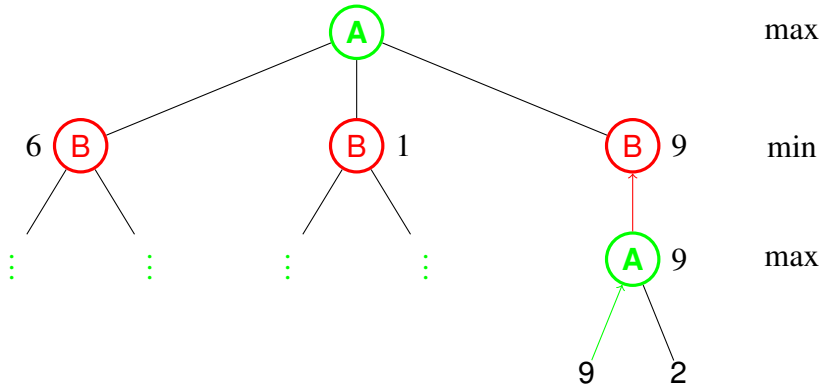
Slika 4.10: 3. korak pronalaženja minimax broja stabla na slici 4.9

4. korak: Pogledajmo sada srednje dijete korijena stabla te proširimo pretragu na njegovu djecu. Njegovo lijevo dijete poprima vrijednost 2jer $2 > 1$, dok desno dijete poprima vrijednost 1 jer je $1 > 0$ par B poprima vrijednost 1 jer $1 < 2$, a B je čvor minimizirajućeg igrača. (pogledaj sliku 4.11)



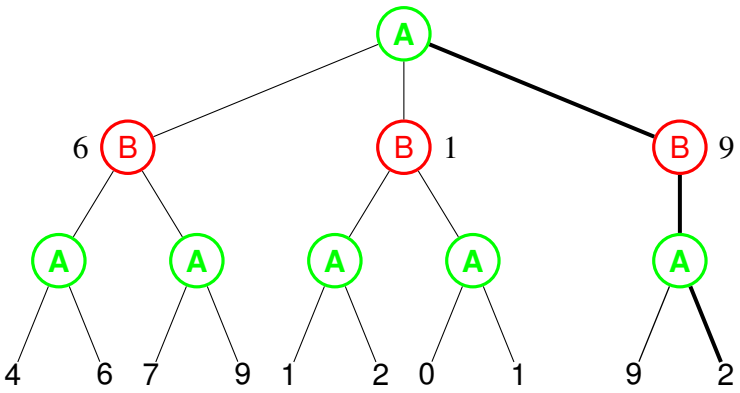
Slika 4.11: 4. korak pronalaženja minimax broja stabla na slici 4.9

5. korak: Pogledajmo posljednje dijete korijena stabla. On ima samo dijete koje poprima vrijednost 9 pa i on poprima vrijednost 9. (pogledaj sliku 4.12)



Slika 4.12: 5. korak pronalaženja minimax broja stabla na slici 4.9

6. korak: Saznali smo vrijednosti koje poprimaju djeca korijena stabla. Sada možemo lako odrediti koju vrijednost poprima korijen stabla. On je čvor maksimizirajućeg igrača te poprima najveću vrijednost među vrijednostima svoje djece, a to je upravo 9. Na stablu je podebljanom crtom označena najbolja strategija za igrača A. (pogledaj sliku 4.13)



Slika 4.13: 6. korak pronalaženja minimax broja stabla na slici 4.9

Vidimo da algoritam, da bi pronašao najbolju strategiju, mora obići sve čvorove stabla. Takav način pretraživanja traje dugo. On bi se trebao moći lako skratiti. Neki čvorovi ne mogu utjecati na konačni ishod pa nemamo potrebu ih niti obilaziti. Upravo takvo razmišljanje će biti temelj alfa-beta podrezivanja.

Alfa-beta podrezivanje

Algoritam "$\alpha - \beta$ podrezivanje" je jedan od algoritama pomoću kojeg tražimo najbolju strategiju u igrama s dva igrača s potpunim informacijama. Te igre mogu imati jako veliko stablo pretraživanja pa je bilo potrebno naći način kako u kratkom roku naći najbolju strategiju. Odgovor na to pitanje je upravo "$\alpha - \beta$ podrezivanje" jer on ne obilazi sve čvorove stabla. On obilazi samo one koji bi mogli utjecati na odabir strategije.

Algoritam je dobio ime po svojim dvijema granicama, α-i i β-i, koje nam pomažu pri odluci koje djelove stabla ćemo posjetiti. Njihovo značenje je:

- α je najveća vrijednost koju maksimizirajući igrač može poprimiti krečući se, po stablu pretraživanja, od trenutnog čvora prema korijenu stabla.

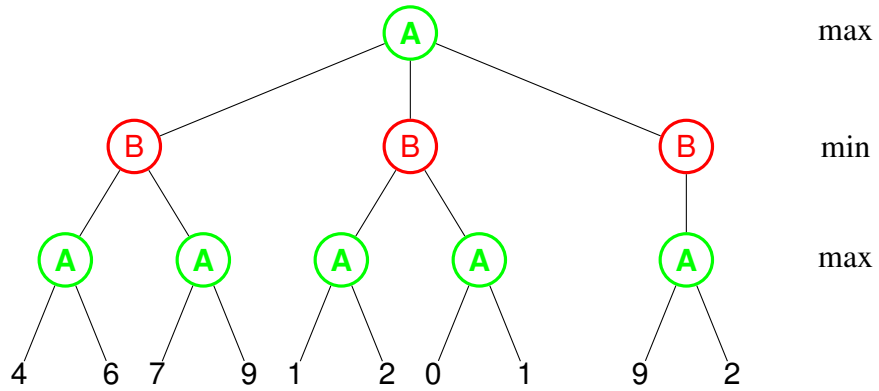
- β je najmanja vrijednost koju minimizirajući igrač može poprimiti krečući se od trenutnog čvora prema korijenu stabla po stablu pretraživanja.

Pretraživanje se izvršava po dubini krečući se od lijeva prema desno. Svakom čvoru stabla pridružujemo vrijednost V koja je na početku inicijalizirana na $-\infty$, ako je čvor čvor maksimizirajućeg igrača, odnosno na $+\infty$ ukoliko je čvor čvor minimizirajućeg igrača. Te vrijednosti se mjenjaju u skladu s alfa-beta uvjetom i s tim da li je trenutni čvor čvor minizirajućeg ili maksimizirajućeg igrača.

Prije nego proširimo pretraživanje na sljedeće dijete trenutnog čvora, potrebno je pogledati da li je to dijete potrebno analizirati. Ukoliko nije kažemo da je došlo do **podrezivanja**. Do podrezivanja dolazi ukoliko je uvjet $\beta \leq V \leq \alpha$ ispunjen.

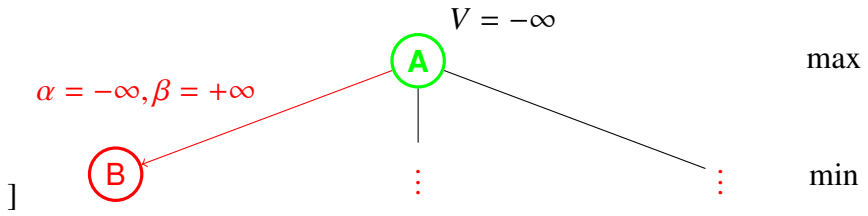
Algoritam će biti detaljnije objašnjen u primjeru koji slijedi. (Radi lakšeg snalaženja stavit ćemo oznake "min" i "max" s lijeve strane stabla.)

Primjer 4.1.6. *Napravimo alfa-beta podrezivanje na stablu prikazanom na slici 4.14:*



Slika 4.14: Stablo na kojem će biti prikazani koraci algoritma α-β podrezivanje

1. korak: Krećemo od korijena stabla. Korijen stabla je maksimizirajući čvor pa njega inicijaliziramo na vrijednost koja je najgora za maksimizirajućeg igrača, a to je $V = -\infty$. Nakon toga je potrebno istražiti djecu. (pogledaj sliku 4.15)



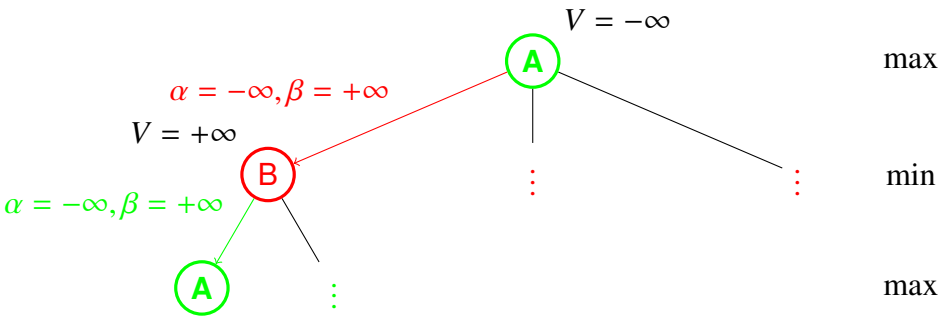
Slika 4.15: 1. korak algoritma α-β podrezivanje

2. korak: Spuštamo se na prvo dijete s lijeve strane. Bridu pomoću kojeg se spuštamo pridružujemo vrijednosti α i β na sljedeći način:

- α je najbolja istražena opcija duž puta do korijena stabla za maksimizirajućeg igrača. β je najbolja istražena opcija duž puta do korijena stabla za minimizirajućeg igrača

- pogledamo vrijednosti u čvorovima stabla do korijena stabla. U ovoj situaciji to je trenutno upravo sam korijen stabla
- korijen stabla nije poprimio nijednu vrijednost pa je najbolja vrijednost za maksimizirajućeg igrača upravo $-\infty$ ($\alpha = -\infty$), a za minimizirajućeg igrača $+\infty$ ($\beta = +\infty$).

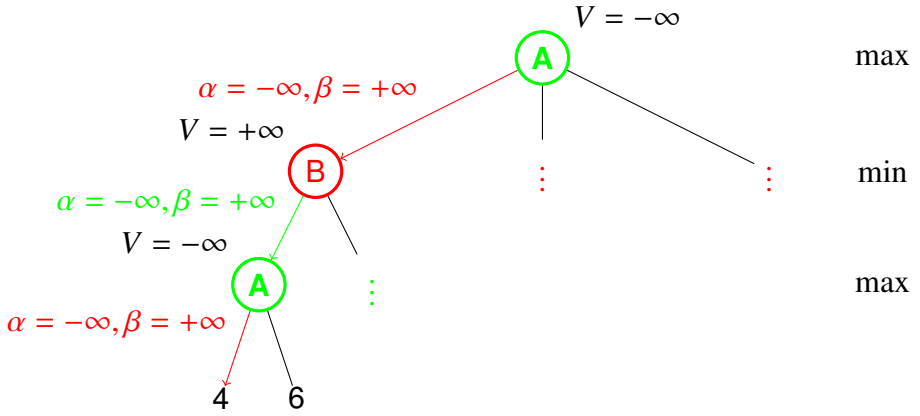
3. korak: Promatramo prvo lijevo dijete korijena stabla. To dijete je čvor minimizirajućeg igrača pa mu pridružujemo vrijednost koja je najgora za minimizirajućeg igrača, a to je $V = +\infty$. Spuštamo se na prvo lijevo dijete te bridu pomoću kojeg se spuštamo pridružujemo vrijednosti $\alpha = -\infty$ i $\beta = +\infty$ na način koji je opisan u prethodnom koraku. (pogledaj sliku 4.16)



Slika 4.16: 3. korak algoritma α-β podrezivanje

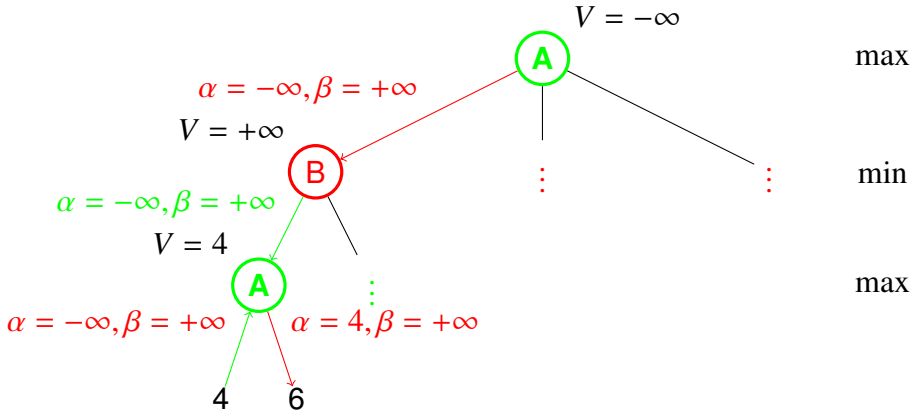
4. korak: Trenutni čvor je čvor maksimizirajućeg igrača pa mu pridružujemo najgoru vrijednost za njega, a to je $V = -\infty$. Bridu po kojem se spuštamo do prvog lijevog djeteta trenutnog čvora pridružujemo vrjednosti $\alpha = -\infty$ i $\beta = +\infty$ po principu koji je opisan u 2. koraku. (pogledaj sliku 4.17)

5. korak: Došli smo do čvora koji je list. Njegova vrijednost je četiri. Tu vrijednost prosljeđujemo do čvora koji je njegov roditelj te uspoređujemo trenutnu vrijednost čvora roditelja s vrijednosti koja je prosljeđenja od djeteta. Čvor roditelj je čvor maksimizirajućeg igrača pa gledamo da li je trenutna vrijednost čvora bolja za maksimizirajućeg od prosljeđene vrijednosti ili obrnuto. Trenutna vrijednost je $-\infty$, prosljeđena vrijednost je 4. Jer je $4 > -\infty$ zaključujemo da je 4 bolja vrijednost za maksimizirajućeg igrača te radi toga vrijednost trenutnog čvora mijenjamo na 4. Prije daljnjeg širenja stabla, provjeravamo da li je vrijednost čvora ($V = 4$) veća od trenute vrijednosti od β. Ako to je slučaj, zanemarujemo ostalu djecu trenutnog roditelja, a ako nije gledamo vrijednost sljedećeg dijeteta. Trenutna vrijednost β je $+\infty$ i ona je veća od trenutne vrijednosti čvora koja je 4 pa je potrebno gledati sljedeće



Slika 4.17: 4. korak algoritma α-β podrezivanje

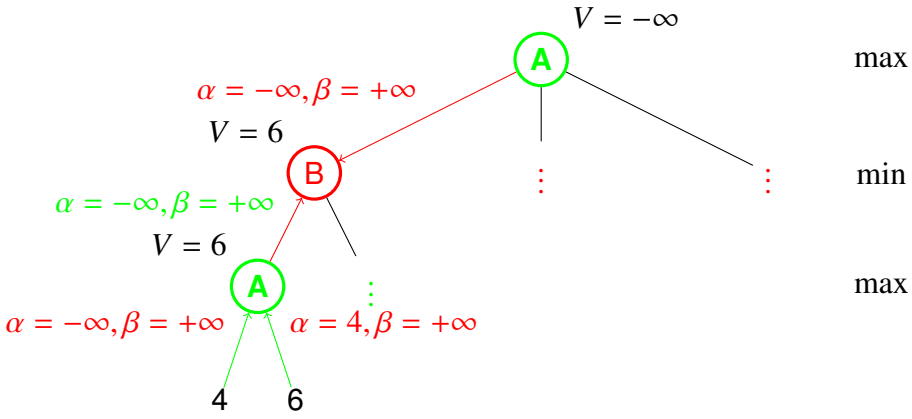
dijete trenutnog čvora, a to je list vrijednosti 6. Bridu kojim se spuštamo do sljedećeg dijeteta pridružujemo α i β. Jer je α jednak najboljoj istraženoj opciji duž puta od trenutnog čvora do korijena stabla za maksimizirajućeg igrača, α poprima vrijednost 4 (dakle $\alpha = 4$). β je nepromjenjen. (pogledaj sliku 4.18)



Slika 4.18: 5. korak algoritma α-β podrezivanje

6. korak: List stabla kojeg trenutno promartamo ima vrijednost 6. Tu vrijednost prosljeđujemo njegovom roditelju. Njegov roditelj je čvor maksimizirajućeg igrača te je za njega vrijednost 6 bolja od vrijednosti 4 pa nam je nova vrijednost čvora roditelja upravo 6 ($V = 6$). Ta vrijednost se prosljeđuje njegovom čvoru roditelju. Taj čvor je čvor minimizirajućeg igrača. Uspoređujemo vrijednost trenutnog čvora (tj. $+\infty$) s prosljeđenom vrijednosti (tj. s 6). Za minimizirajućeg igrača 6 je bolja opcija od

$+\infty$ pa se vrijednost čvora mijenja na 6 ($V = 6$). Čvor minimizirajućeg igrača gleda da li je potrebno analizirati svoju ostalu djecu. Da bi to odlučio uspoređujemo vrijednost trenutnog čvora, koja je 6, s najboljom opcijom koju maksimizirajući igrač može poprimiti od trenutnog čvora prema vrhu stabla, a to je $-\infty$. Jer $\alpha = -\infty$, a $-\infty < 6 = V$ analiziramo sljedeće dijete trenutnog čvora. U suprotnom bi došlo do podrezivanja, tj. ne bi bilo potrebno gledati ostalu djecu trenutnog čvora. (pogledaj sliku 4.19)



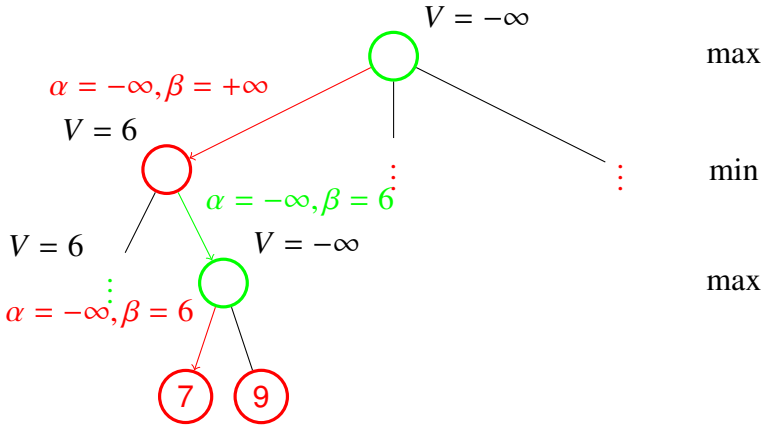
Slika 4.19: 6. korak algoritma α-β podrezivanje

7. korak: Brid kojim se spuštamo od roditelja do djeteta poprima vrijednosti α i β na sljedeći način:

- pogledamo do sad istražene vrijednosti čvorova za minimizirajućeg i maksimizirajućeg igrača od trenutnog čvora do vrha stabla
- α je jednak najboljoj istraženoj vrijednosti čvora za maksimizirajućeg igrača, a to je $-\infty$
- β je analogno za minimizirajućeg igrača, a to je 6

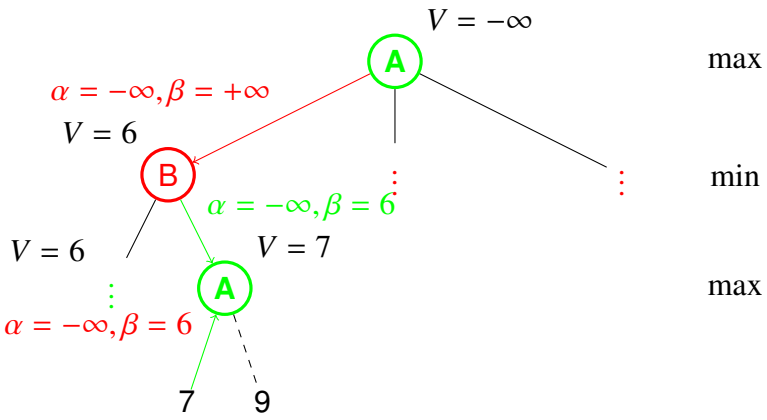
Dijete poprima vrijednost koja je najgora za maksimizirajućeg igrača, a to je $-\infty$. Spuštamo se do njegovog prvog djeteta s lijeva. Bridu po kojem se spuštamo dodijeljujemo vrijednosti α i β na način po kojem smo dodjelili vrijednosti i prethodnom bridu. (pogledaj sliku 4.20)

8. korak: Trenutni čvor je list vrijednosti 7. Njegovu vrijednost prosljedimo njegovom roditelju. Jer je 7 bolja opcija za vrijednost čvora maksimizirajućeg igrača, taj čvor poprima vrijednost 7. Maksimizirajući igrač provjerava da li se isplati gledati njegovu ostalu djecu. Da bi to znao uspoređuje svoju trenutnu vrijednost, 7, s najboljom



Slika 4.20: 7. korak algoritma α-β podrezivanje

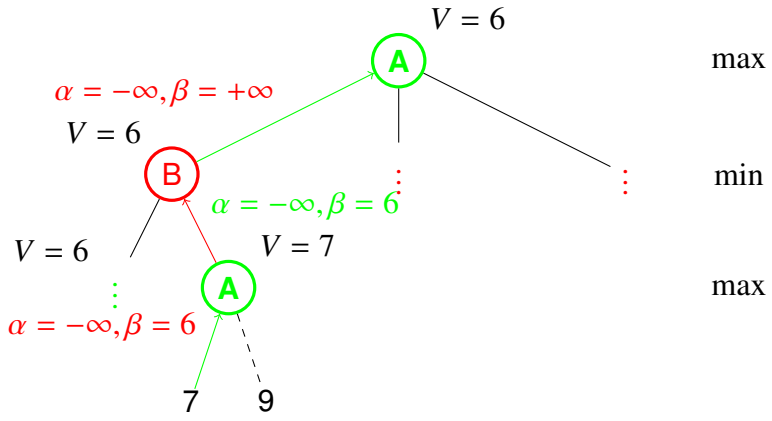
vrijednosti koju minimizirajući igrač može garantirati. Ta vrijednost je pohranjena u varijabli β i ona je jednaka 6. Ako maksimizirajući igrač može poprimiti vrijednost 7 ili veću u trenutnom čvoru, minimizirajući igrač nikada neće dopustiti da maksimizirajući igrač odigra tu opciju. Dobili smo da je $V = 7 > 6 = \beta$ što znači da ne trebamo gledati ostalu djecu trenutnog čvora, odnosno došlo je do podrezivanja. (pogledaj sliku 4.21)



Slika 4.21: 8. korak algoritma α-β podrezivanje

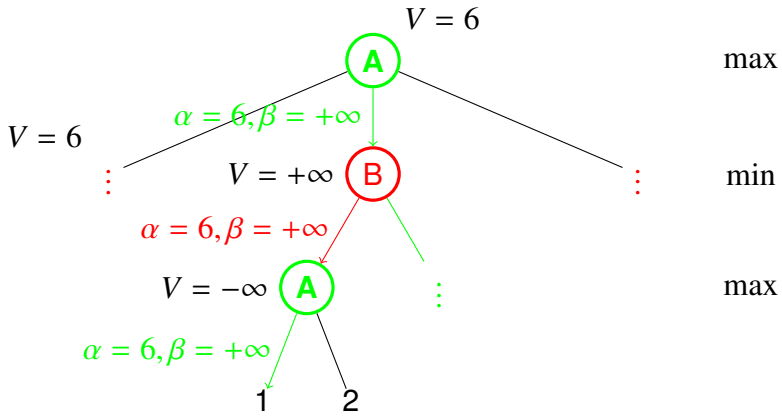
9. korak: Vrijednost čvora $V = 7$ prosljeđujemo njegovom roditelju. Roditelj je čvor minimizirajućeg igrača i njegova trenutna vrijednost je 6. Za minimizirajućeg igrača 6 je bolja opcija od 7, pa taj čvor zadržava trenutnu vrijednost. $V = 6$ se prosljeđuje roditelju čvora. Roditelj čvora je korijen stabla. On je čvor maksimizirajućeg igrača i

jer je 6, što je prosljeđena vrijednost, bolje za maksimizirajućeg od njegove trenutne vrijednosti koja je jednaka $-\infty$, vrijednost korijena stabla se mijenja u 6. (pogledaj sliku 4.22)



Slika 4.22: 9. korak algoritma α-β podrezivanje

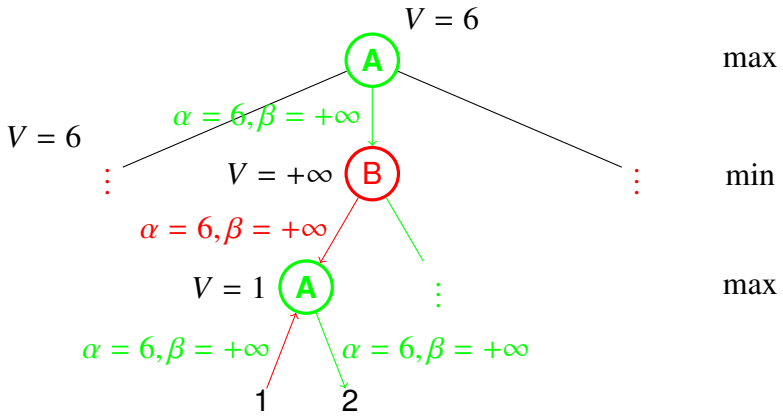
10. korak: Vrijednost korijena stabla uspoređujemo s vrijednosti od α i β. α i β još nisu poprimili nijednu vrijednost stoga neće doći do podrezivanja, odnosno potrebno je analizirati sljedeće dijete korijena stabla. Analiza se provodi na način opisan u koracima od 1. do 8. iz čega dobijemo stablo prikazano na slici 4.23)



Slika 4.23: Analiza srednjeg djeteta stabla pomoću α-β podrezivanja

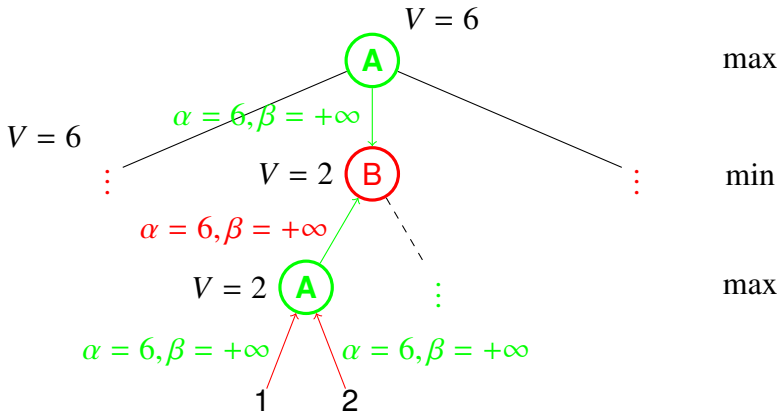
Vrijednost lista, čija je vrijednost 1, smo proslijedili njegovom roditelju. Njegov roditelj je čvor maksimizirajućeg igrača te on poprima vrijednost $V = 1$. Jer $V =$

$1 < +\infty = \beta$ potrebno je analizirati i drugo dijete roditelja, tj. list s vrijednosti 2. (pogledaj sliku 4.24)



Slika 4.24: Analiza srednjeg djeteta stabla pomoću α-β podrezivanja

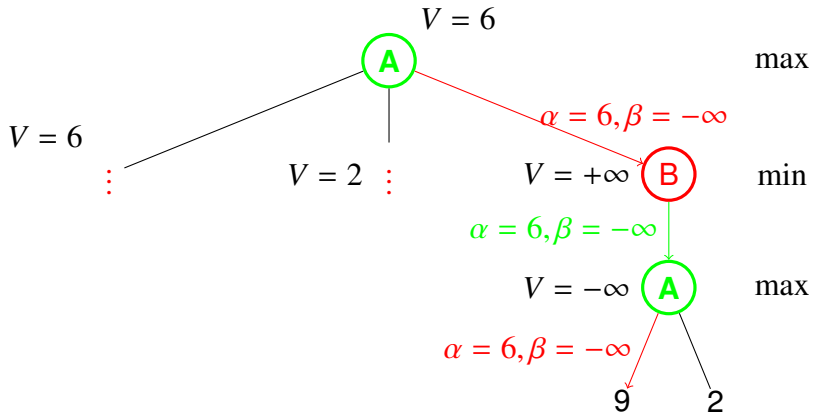
Vrijednost čvora maksimizirajućeg igrača čija je trenutna vrijednost jednaka 1 će biti promjenjena na 2 jer je $2 > 1$, pa je 2 bolja opcija za maksimizirajućeg igrača. Tu vrijednost zatim prosljeđujemo do njegovog roditelja. Roditelj poprima vrijednost 2 jer je $2 < +\infty$ što je bolje za vrijednost čvora minimizirajućeg igrača. (pogledaj sliku 4.25)



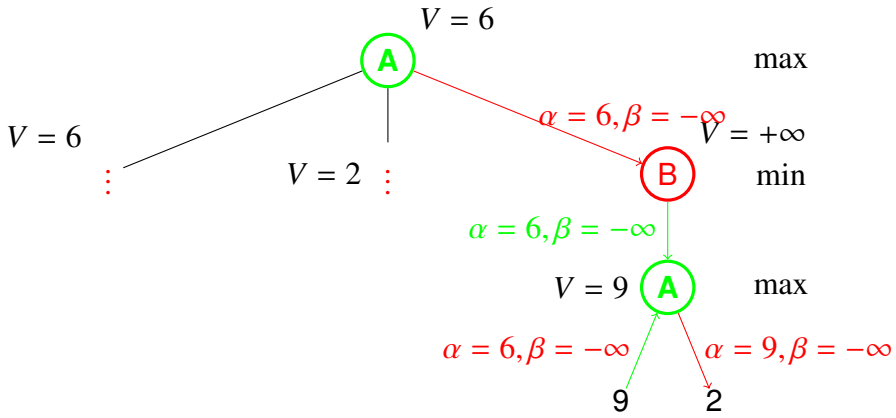
Slika 4.25: Analiza srednjeg djeteta stabla pomoću α-β podrezivanja

Čvor je poprimio vrijednost 2. On ima još djece no nije ih potrebno analizirati jer je vrijednost 2 najveća vrijednost koju će minimizirajući igrač dozvoliti da trenutni čvor poprimi. Tu vrijednost prosljedimo korijenu stabla. Korijen stabla ne mijenja svoju vrijednost jer je on čvor maksimizirajućeg stabla, a 2 je manje od 6.

11. korak: Preostalo nam je još jedno dijete korijena stabla kojeg nismo analizirali. Da bismo ga analizirali ponovimo postupak opisan u koracima od 1. do 8.. Koraci su prikazani na slikama 4.26 do 4.28.



Slika 4.26: Analiza desnog djeteta stabla pomoću α-β podrezivanja

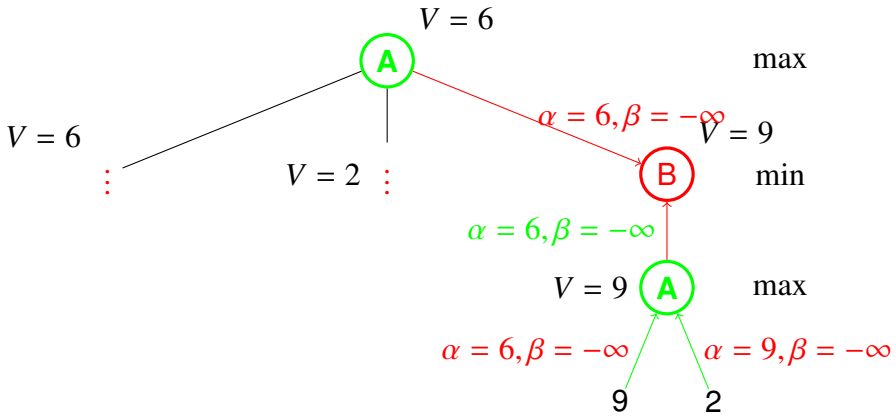


Slika 4.27: Analiza desnog djeteta stabla pomoću α-β podrezivanja

Treće dijete korijena stabla je poprimilo vrijednost 9.

12. korak: Konačna vrijednost korijena stabla je 9. Ta vrijednost je prosljeđena od njegovog desnog djeteta, što znači da bi algoritam alfa-beta podrezivanje odabrao upravo taj čvor kao najbolji potez.

Vidimo da algoritam nije obišao sve čvorove stabla, ali je svejedno pronašao najbolju strategiju. Ta stratgija je jednaka kao i odabrana strategija u minimax algoritmu.



Slika 4.28: Analiza desnog djeteta stabla pomoću α-β podrezivanja

Prednosti algoritma "Alfa-beta podrezivanje" nad minimax algoritmom

Prednost alfa-beta podrezivanja nad minimax-om se zasniva na tome da se u alfa-beta podrezivanju određeni dijelovi stabla pretraživanja mogu eliminirati dok je u minimax algoritmu potrebno proći sve čvorove stabla. Na taj način se orjentiramo na podstabla koja su značajnija za konačni rezultat. Optimizirani oblik smanjuje efektivnu dubinu za nešto više od polovice u odnosu na jednostavni minimax, ako su čvorovi ocjenjeni u optimalnom ili blizu optimalnog poretka.

Neka je b prosjek grananja, a d broj slojeva u stablu. U najgorem slučaju, broj listova u stablu koje ćemo ocjenjivati je $O(b * b * \ldots * b) = O(b^d)$ što je jednako kao i kod jednostavnog minimax pretraživanja. Kod optimalnog poretka, odnosno kada se najbolji koraci prvi pretražuju, broj listova u stablu koje ćemo ocjenjivati je $O(b * 1 * b * 1 * \ldots * b)$ ako je dubina parna, odnosno $O(b * 1 * b * 1 * \ldots * 1)$ ako je dubina neparna. Točnije, broj listova u stablu koje ćemo ocjenjivati je $O(b^{d/2}) = O(\sqrt{b^d})$.

Pretpostavimo da je broj slojeva u stablu neparan. Za isti broj operacija možemo pretražiti duplo više čvorova stabla. Kod neparnog broja slojeva u stablu je potrebno promatrati sve poteze prvog igrača da bi se pronašao najbolji. Potreban je samo jedan potez drugog igrača (njegov najbolji) da bi pobio prvog igrača, radi toga dobijemo izraz $b * 1 * b * 1 * \ldots * b$. Alfa-beta osigurava da nije potrebno gledati ostale poteze drugog igrača. Kada se čvorovi slažu slučajnim poretkom, prosječan broj ocjenjenih čvorova je oko $O(b^{3d/4})$.

4.2 Križić-kružić

U igri križić-kružić imamo dva igrača. Jedan igrač je čovjek, a drugi računalo. Računalo koristi $\alpha - \beta$ podrezivanje za odabir svoje strategije. Prvi na potezu je čovjek, pa zatim računalo. Čovjek može odabrati prvi potez računala. S d ćemo označiti broj čvorova koje trebamo proći da bi došli od korijena do trenutnog čvora. Najveća vrijednost od d je 9. Isplate su sljedeće:

- Ako je maksimizirajući igrač pobjedio isplata je $9 - d$
- Ako je minimizirajući igrač pobjedio isplata je $-9 + d$
- Ako je neriješeno isplata je 0

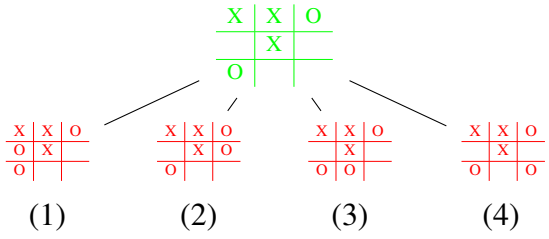
Neka igru igra računalo protiv računala. Prvi na potezu je X pa zatim O. Prvi potez računala čiji je znak O je povukao čovjek i on je stavio O na mjesto $(3, 1)$. To je napravljeno da bi mogli dobiti jednu veoma specifičnu situaciju prikazanu na tablici 4.1.

X	X	O
	X	
O		

Tablica 4.1: Početno stanje na tabli za igru križić-kružić

Promotrimo tablicu 4.1. Sljedeći na potezu je "O". Vidimo da X, bez ozbira na to što "O" napravi, ima sigurnu pobjedu. "O" će staviti svoj znak na poziciju $(2, 1)$. Zašto je "O" to učinio vidjet ćemo analizirajući stablo igre kojoj je korijen tablica 4.1.

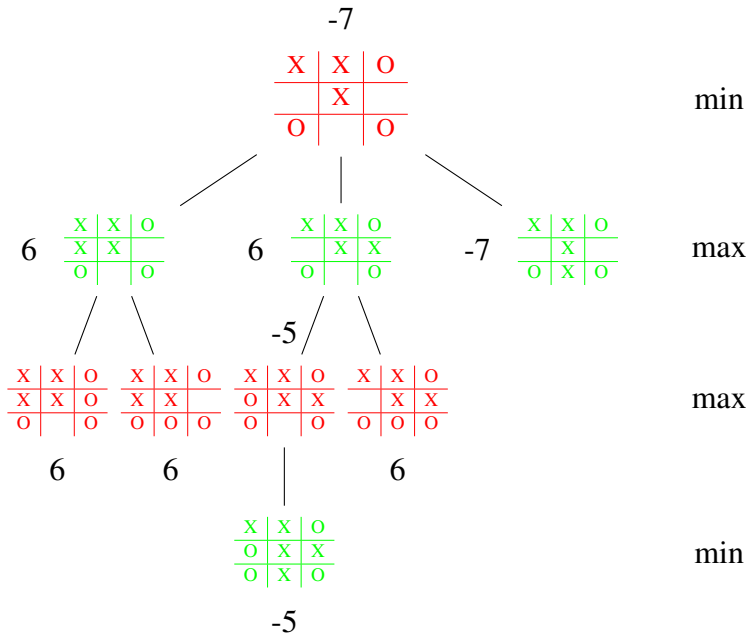
Stablo s korijenom 4.1 i prvom djecom je prikazano na slici 4.29.



Slika 4.29: Početno stanje ploče je korijen stabla. Djeca korijena su mogući potezi znaka "O"

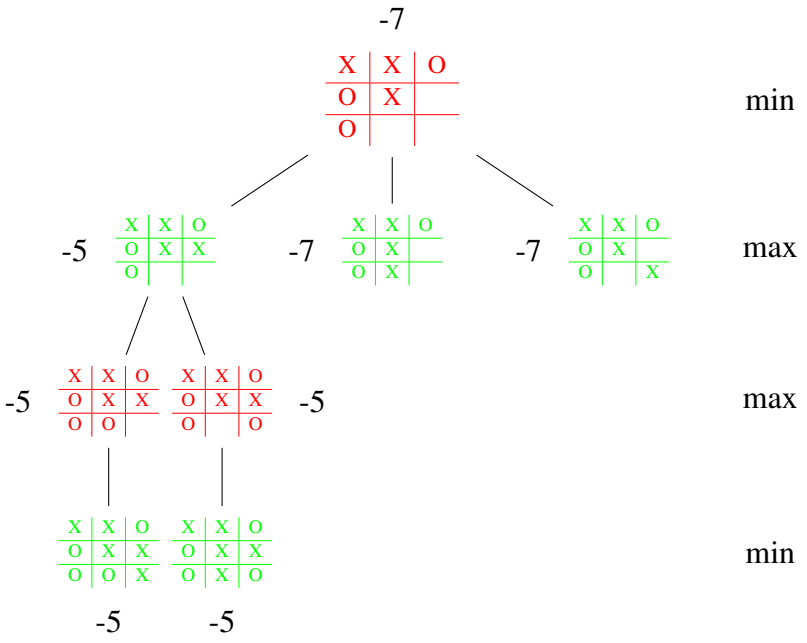
Čovjeku bi bilo logično da je računalo odabralo strategiju (3) ili (4), a ne (1). Štoviše strategija (4) izgleda kao najbolji izbor jer odabiranjem strategije (4) ”O”, ukoliko ”X” povuče krivi potez, stvara za sebe sigurnu pobjedu.

Analizirajmo prvo čvor (4) da vidimo koju vrijednost on poprimi. Analizu tog čvora možemo vidjeti na slici 4.30. Na toj slici vidimo da je ”O” pronašao strategije u kojima može pobjediti, no minimizirajući čvor ”X” će ga spriječiti u tome.

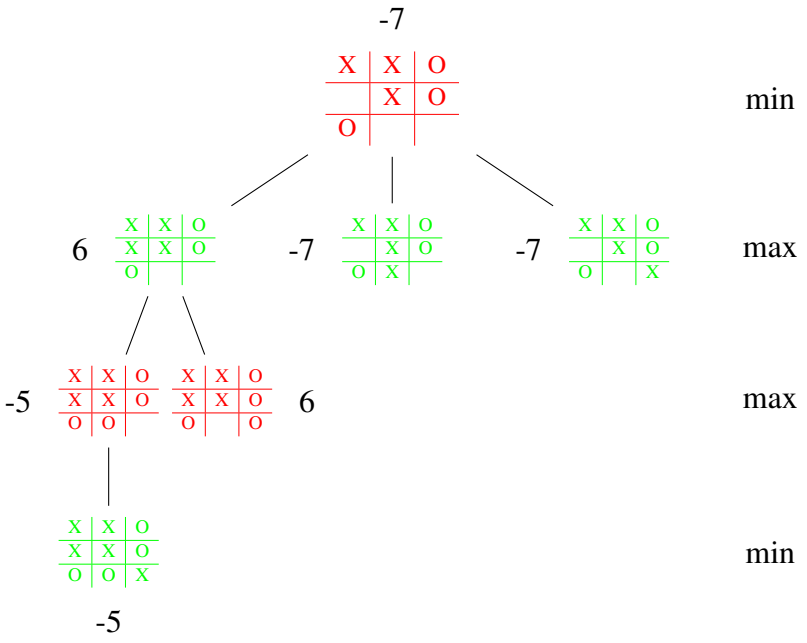


Slika 4.30: Detaljan prikaz vrijednosti koje poprima dijete (4) sa slike 4.29.

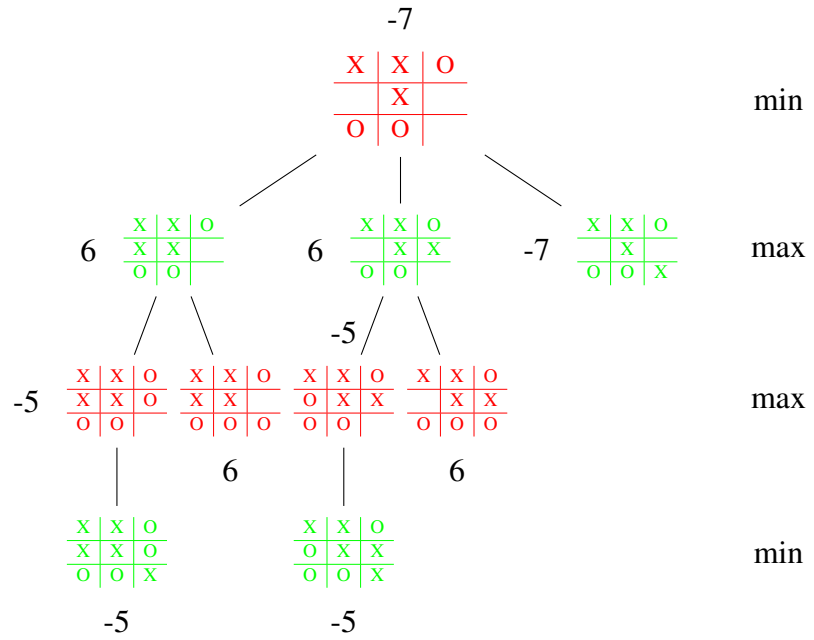
Na slikama 4.31, 4.32 i 4.33 vidimo detaljnu analizu čvorova (1), (2) i (3) sa slike 4.29. Vidimo da sva 4 čvora poprimaju iste vrijednost, tj. ne postoji slučaj u kojem će ”X” dozvoliti da”O” pobjedi. Implementirati algoritam opisan u poglavlju 5 u ovom slučaju uzima prvu lijevu vrijednost pa je radi toga ”O” odabralo poziciju $(1, 2)$. ”O” je moglo odabrati bilo koju od ostalih pozicija, no to ne bi utjecalo na njegovu pobjedu jer u svakom slučaju ono gubi igru u sljedećem potezu igrača ”X”.



Slika 4.31: Detaljan prikaz vrijednosti koje poprima dijete (1) sa slike 4.29.



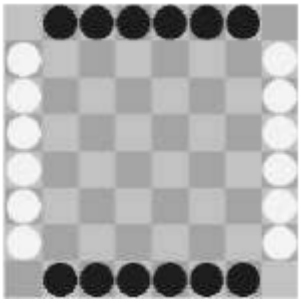
Slika 4.32: Detaljan prikaz vrijednosti koje poprima dijete (2) sa slike 4.29.



Slika 4.33: Detaljan prikaz vrijednosti koje poprima dijete (3) sa slike 4.29.

4.3 Usporedba algoritama u igri ”Linije akcije”

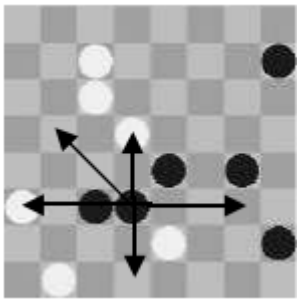
Linije akcije je igra za dva igrača sume nula. Igra je veoma slična šahu, a igra se na ploči s 8×8 crno-bijelih kvadratnih polja. Na svakoj strani se nalazi 6 figura tako da su figure jednog igrača smještene jedna nasuprot druge, a u samim kutovima ploče nije smještena niti jedna figura (pogledaj sliku 4.34).



Slika 4.34: Početne pozicije u igri ”Linije akcije”.

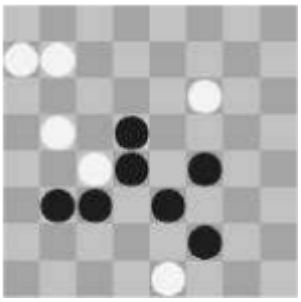
Igrači naizmjenično pomiču figure. Prvi na potezu je igrač sa crnim figurama. Igrač se može pomicati horizontalno, dijagonalno ili vertikalno onoliko polja koliko duž linije po

kojoj se kreće ne naiđe na neku drugu figuru (pogledaj sliku 4.35). Igrač može preskakati svoje figure, ali ne i suparnikove. Igrač može zarobiti suparnikove figure tako da stane na njih.



Slika 4.35: Potezi u igri "Linije akcije".

Cilj igre je da su sve igračeve figure povezane (pogledaj sliku 4.36). Ukoliko neki igrač ne može pomaknuti svoje figure, on preskače potez. Ako se neka pozicija za jednog igrača pojavljuje treći put za redom, igra je neriješena.



Slika 4.36: Završno stanje igre "Linije akcije" u kojem je igrač sa crnim figurama pobjedio.

U [5] vidimo da je igra računana pomoću nekoliko algoritama pa i pomoću već opisanih proof-number pretraživanja i α-β podrezivanja te je napravljena usporedba među algoritmima s obzirom na broj pozicija koje je pojedini algoritam točno riješio (od mogućih 488), broj posjećenih čvorova i ukupno vrijeme potrebno pojedinom algoritmu da bi došlo do riješenja. Prilikom računanja broja posjećenih čvorova u α - β podrezivanju na dubini i, brojali su se čvorovi samo prilikom prve iteracije u kojoj je ta dubina dosegnuta, a u proof-number pretraživanju brojali su se svi posjećeni čvorovi.

Navedene složenosti su prikazane u tablici 4.2. Iz tablice vidimo da proof-number pretraživanje zahtjeva manje vremena za pronalazak rješenja i posjećuje manje čvorova, no α - β podrezivanje daje točno rješenje za više pozicija.

Algoritam	Broj riješenih pozicija (od 488)	ukupni čvorovi	ukupno vrijeme
α - β podrezivanje	383	1 711 578 143	22 172 320
proof-number pretraživanje	356	89 863 783	830 367

Tablica 4.2: Usporedba algoritama α - β podrezivanje i proof-number pretraživanje

Poglavlje 5

Implementacija algoritma α - β podrezivanje

U sklopu rada implementirala sam igru za dva igrača križić-kružić. Igru mogu igrati dva čovjeka, čovjek i računalo ili računalo protiv samog sebe. Računalo koristi α-β podrezivanje da bi odlučio koji potez će napraviti. Na slici 5 je prikazan algoritam kojeg računalo koristi pri donošenju odluke o svom sljedećem potezu.

```
def alphaBeta (self, alpha, beta, sign, maxP, pathCoordinates):
        childNodes = self.openAll(pathCoordinates)


        #sadasnji cvor se sastoji od pathCoordinates, childNodes i oznacenih po
        mm = self.putInMatrix(pathCoordinates, sign)
        win,winSign = self.CheckWins(mm)
        hSign = 1
        if sign == 'X':
            hSign = 0
        else:
            hSign = 1

        #pobjede
        if win == True and hSign == winSign:
            Row, Col = pathCoordinates[0]
            return Row, Col, 9-len(pathCoordinates)
        if win == True and hSign != winSign:
            Row, Col = pathCoordinates[0]
            return Row, Col, -9+len(pathCoordinates)
```

```
if childNodes == []:
    Row, Col = pathCoordinates[0]
    return Row, Col, 0

i = 0
j = 0
if maxP == True:
    for node in childNodes:
        pathCoordinates.append(node)
        Row, Col, V = self.alphaBeta(alpha, beta, sign, False, pathCoordinate
        pathCoordinates.pop()
        if i == 0:
            alpha = V
            maxRow = Row
            maxCol = Col
        else:
            if alpha < V:
                alpha = V
                maxRow = Row
                maxCol = Col
        i = i + 1
        r,c = node
        if alpha >= beta:
            break
    return maxRow, maxCol, alpha
else:
    for node in childNodes:
        pathCoordinates.append(node)
        Row, Col, V = self.alphaBeta(alpha, beta, sign, True, pathCoordinates
        pathCoordinates.pop()
        if j == 0:
            beta = V
            minRow = Row
            minCol = Col
        else:
            if beta > V:
                beta = V
                minRow = Row
                minCol = Col
```

```
            j = j+1
            r,c = node
            if alpha >= beta:
                break
        return minRow, minCol, beta
```

pathCoordinates je lista tuplova koja pamti koordinate koje smo prošli da bi došli do trenutnog stanja križić-kružić table (kraće: table). To je jedan put u stablu pretraživanja od korijena do lista stabla. U njega se redom naizmjenično spremaju odabrane pozicije za oba znaka, prvo za jedan pa onda za drugi. Funkcija *putInMatrix* popunjava matricu nulama i jedinicama ovisno o tome da li se na određenoj poziciji u tabli ili listi *pathCoordinates* nalazi znak *X* ili *O*. Funkcija *CheckWins* provjerava da li matrica koju je vratila funkcija *putInMatrix* sadrži pobjedu za neki znak i koji je to znak. Ako je došlo do pobjede funkcija *alphaBeta* vraća prvi tuple sačuvan u listi *pathCoordinates* i vrijednost $9 - len(pathCoordinates)$ ili $-9 + len(pathCoordinates)$ ovisno o tome da li je maksimizirajući ili minimizirajući igrač pobjedio. Ako nitko nije pobjedio vraćamo 0. To su upravo situacije kada je algoritam u stablu pretraživanja došao do lista stabla.

U daljnjim linijama programa slijede rekurzivni pozivi algoritma. Algoritam se naizmjenično poziva za minimizirajućeg i maksimizirajućeg igrača i ovisno o tome za kojeg igrača trenutno računamo, dodjeljujemo vrijednosti varijablama α, β i V.

Bibliografija

[1] *Game complexity*, http://en.wikipedia.org/wiki/Game_complexity.

[2] Kevin Leyton-Brown i Yoav Shoham, *Essentials of Game Theory*, 2008, str. 3–18, 31–38.

[3] Geoffrey Mohan, *Economists finally test prisoner's dilemma on prisoners*, http://www.latimes.com/science/sciencenow/la-sci-sn-prisoners-dilemma-cooperation-20130725-story.html.

[4] Martin Schijf, *Proof-Number Search and Transpositions*, (1993), 14.

[5] Mark HM Winands, Jos WHM Uiterwijk i Jaap van den Herik, *PDS-PN: A new proof-number search algorithm*, Computers and Games, Springer, 2003, str. 61–74.

Sažetak

U radu je objašnjeno kako pronaći Nashovu ravnotežu i *subgame-perfect* ravnotežu u igrama s dva igrača. Opisan je zapis igara u normalnoj i proširenoj formi te pronalazak dominantnih strategija u svakoj od formi.

U četvrtom poglavlju su objašnjeni neki algoritmi pomoću kojih pronalazimo Nashovu ravnotežu u igrama s dva igrača. Pomoću tih algoritama računalo donosi odluku koju strategiju će odabrati. Poseban fokus smo stavili na algoritam α-β podrezivanje. Taj algoritam je korišten za razvoj softvera u igri križić-kružić koji je priložen uz rad.

Radi ograničenih računalnih resursa, efikasnost računala u pronalasku svoje najbolje strategije u nekoj igri ovisi o veličini stabla pretraživanja. Stablo pretraživanja je veće ukoliko je broj mogućih poteza u igri veći. Na primjer, igra križić kružić ima manji broj poteza od igre šah, pa je njeno stablo pretraživanja mnogo manje nego stablo pretraživanja u šahu. Što je stablo pretraživanja manje to će računalo točnije odrediti svoju najbolju strategiju i samim tim čovjeku će biti teže pobjediti računalo u nekoj igri.

Summary

In this thesis, the finding of Nash equilibrium and subgame-perfect equilibrium was explained for two-player games. We considered normal-form and extended-form game representation (and their respective dominant strategies.

Algorithms for finding Nash equilibrium in two-player games, as explained in the fourth chapter, are the base for optimal computer strategy decision making. The focus of this dissertation is the α - β pruning algorithm used for the making of the tic-tac-toe game attached to the thesis.

The efficiency of finding optimal game strategy is greatly affected by limited computer resources. Therefore it is also closley affiliated with the size of the game search tree. The greater the game search tree, so grows the number of possible outcomes. Therefore, the smaller the search tree, the computer accuracy of finding the optimal strategy grows, and so does the difficulty of defeating the computer.

Životopis

OBRAZOVANJE

Diplomskii sveučilišni studij Računarstvo i matematika 2013-2015

Prirodoslovno - Matematički fakultet Zagreb, Matematički odjel

Preddiplomski sveučilišni studij Matematika 2006-2013

Prirodoslovno - Matematički fakultet Zagreb, Matematički odjel

2002-2006

Prirodoslovno - Matematička gimnazija Požega

RADNO ISKUSTVO

Python software developer 08.12.2014-danas

AVL - AST d.o.o.

Kreiranje automatiziranih test skripti u programskom jeziku Python.

PHP developer 23.03.2015-03.04.2015

Bolja ideja d.o.o

Razvoj PHP dijela stranice www.boljaideja.hr koji ažurira podatke na navedenoj stranici o raspoloživim proizvodima i njihovim detaljima. Potrebne podatke čita iz XML dokumenta.

SPOSOBNOSTI

Jezici	Engleski (fluent)
Računalne sposobnosti	Microsoft Office paket, LaTeX, Python, wxPython, MySQL, PHP, JavaScript, HTML, CSS, C, C++, C#