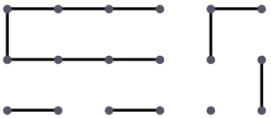
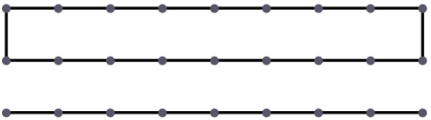
Otvoreni lanci predstavljaju one kod kojih poslednji kvadrat u lancu ima dve dovršene stranice, a sledeći kvadrat nije moguće dovršiti zato što ima manje od dve dovršene stranice, ili sledeći kvadrat ne postoji, jer je poslednji kvadrat na ivici table.



Sl 3. Primer otvorenog lanca

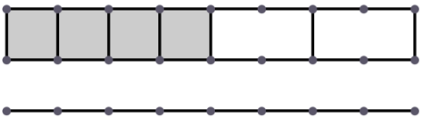
Ovi lanci su bitni, jer pri njihovom dovršavanju igrač može da odluči da ih završi u potpunosti, ili da pak kraj lanca prepusti protivniku ne bi li sebi garantovao više kvadrata iz nekog drugog lanca.



Sl 4. Primer partije u kojoj je bolje prepustiti dovršavanje lanca protivniku

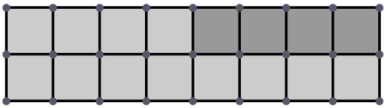
Na slici 4 možemo videti jednostavan primer ove situacije. Stanje na tabli je takvo da postoji jedan horizontalan zatvoreni lanac u prvom redu, dok je u drugom dovoljno povući jednu liniju ne bi li nastao jedan (ukoliko se odabere linija na ivici table) ili dva (ukoliko se odabere neka druga linija) otvorena lanca.

Logičan potez za prvog igrača jeste da počne da dovršava prvi lanac, i to može učiniti u potpunosti. Međutim, njegov sledeći potez bi tada omogućio protivniku da dovrši sve kvadrate u donjem redu, rezultujući u nerešenoj partiji. Zbog toga, optimalan potez je da prvi igrač dovrši samo prva 4 kvadrata, a onda da odabere liniju između šestog i sedmog kvadrata.



Sl 5. Primer prepuštanja lanca protivniku

Na ovaj način, drugi igrač je primoran da dovrši dva novonastala lanca, ali kako su oni dužine 2 kvadrata, drugi igrač će dobiti samo 4 poena. Njegov sledeći potez će, samim tim, morati da bude odabir linije u donjem redu, efektivno dajući prvom igraču mogućnost da dovrši sve ostale kvadrate i pobedi sa 8 poena razlike.



Sl 6. Pobeda sa 8 poena razlike usled prepuštanja lanca protivniku

Razlika između otvorenih i zatvorenih lanaca je bitna zbog načina na koji je moguće prepustiti kvadrate protivniku. Kod otvorenih lanaca dužine$N$, moguće je dovršiti prvih $N-2$ kvadrata i nakon toga odabrati liniju na kraju lanca. Kod zatvorenih lanaca iste dužine, moguće je dovršiti $N-4$ kvadrata i onda odabrati liniju između kvadrata na pozicijama $N-2$ i $N-1$. Samim tim, očigledno je da odluku o prepuštanju lanca protivniku ima smisla praviti samo kod zatvorenih lanaca dužine veće od 3, odnosno otvorenih lanaca dužine veće od 1.Ne bi li igrač maksimizovao broj osvojenih poena, takođe je bitno primetiti da lanac ne treba prepuštati protivniku sve dok njegova dužina ne postane 4 (kod zatvorenih lanaca), odnosno 2 (kod otvorenih).

Imajući ova ograničenja na umu, implementiran je algoritam koji u situaciji kada treba razmotriti da li prepustiti protivniku lanac koristi minimaks pristup ne bi li doneo odluku o optimalnosti. Pri izvršavanju minimaks dela, algoritam rekurzivno nastavlja simulaciju igre tako što određuje optimalni potez za protivnika, što, u zavisnosti od trenutnog stanja table, može dovesti do dalje rekurzije poruzrokovane minimaks pristupom.

Kako je jedina odluka koju minimaks deo algoritma treba da donese da li prepustiti lanac protivniku ili ne, krajnji rezultat se može predstaviti binarnim stablom u kojem svaki čvor ima maksimalan broj poena koji igrač može osvojiti optimalnijim potezom. Kako se igrači smenjuju naizmenično, tako će se smenjivati i za kog igrača se vrši optimizacija na određenom nivou stabla.

Na primer, algoritam u određenom čvoru će se izvršiti rekurzivno za njegova dva deteta, i onda izabrati dete sa manjom vrednošću – razlog za ovo je zato što vrednosti u deci predstavljaju optimalne poteze za protivnika, što

znači da za trenutni čvor zapravo treba odabrati dete sa lošijim rezultatom.

Sa porastom veličine table, ovo binarno stablo može postati veoma veliko, jer broj čvorova raste eksponencijalno sa brojem preostalih poteza. Iz ovog razloga, rekurzija je ograničena maksimalnom dubinom, a kasnije u radu će biti reči o drugim metodama optimizacije.

III. IMPLEMENTACIJA

Algoritmi analizirani u ovom radu implementirani su u okviru veb aplikacije ne bi li bio omogućen jednostavan vizuelni prikaz odigravanja partije, kao i intuitivan korisnički interfejs za upravljanje simulacijama. Ovo je postignuto upotrebom jezika JavaScript, uz koji su korišćene biblioteke React i Flux za implementaciju interfejsa i kontrolu toka podataka, respektivno.

Sami algoritmi su realizovani u klasama koje su organizovane tako da klase kompleksnijih algoritama nasleđuju klase jednostavnijih algoritama. Ukoliko određeni algoritam nije u mogućnosti da napravi odluku o potezu na osnovu logike koja je specifična za taj algoritam, on tu odluku delegira roditeljskoj klasi koja predstavlja sledeći, jednostavniji algoritam – na primer, klasa za pohlepni algoritam nasleđuje klasu za nasumični algoritam, pa se tako odabir linije u slučaju da nije moguće dovršiti nijedan kvadrat prepušta nasumičnom algoritmu.

Kao što je pomenuto, radi jednostavnosti, ovi algoritmi ne čuvaju stanje između različitih poteza, pa je tako logika implementirana u okviru statične funkcije koja za ulaz ima trenutno stanje table, a za izlaz potez igrača. Ovakav pristup ostavlja prostora za poboljšanje performansi algoritama, ali fokus ovog rada je prvenstveno na samim rezultatima algoritama, pa optimizacije nisu uzete u obzir.

Iako ovaj pristup zahteva da određene kalkulacije budu ponovljene pri svakom potezu, on takođe olakšava implementaciju minimaks algoritma. Naime, minimaks algoritam je implementiran upotrebom statične, rekurzivne funkcije koja, slično glavnoj funkciji, za ulaz ima trenutno stanje table, ali, za razliku od glavne funkcije, za izlaz nema potez koji treba napraviti, već maksimalan broj poena koji je moguće osvojiti počevši od tog stanja. Na osnovu ovih informacija, algoritam je u stanju da odluči koji potez je najoptimalniji, i to čini izvršavanjem sledećih koraka:

1. Algoritam vrši analizu trenutnog stanja table i identifikuje sve otvorene i zatvorene lance. Ukoliko ne postoji nijedan lanac, odluka se prepušta roditeljskoj klasi, odnosno odbrambenom algoritmu.
2. Ukoliko postoje otvoreni lanci dužine 1 ili zatvoreni lanci dužine manje od 4, optimalan potez je zatvaranje tih lanaca, jer oni ne mogu biti prepušteni protivniku, pa se bira potez koji dovršava prvi kvadrat u jednom od tih lanaca.
3. Ukoliko postoje otvoreni lanci dužine veće od 2 ili zatvoreni lanci dužine veće od 4, bira se potez koji dovršava prvi kvadrat u jednom takvom lancu, jer za cilj imamo da otvorene lance dovedemo do dužine 2, a zatvorene lance do dužine 4.
4. Ukoliko su preostali isključivo lanci tih dužina, optimalan potez je dovršavanje svih lanaca osim jednog koji je najmanje dužine.
5. Najzad, ukoliko je preostao samo jedan lanac, došli smo do trenutka gde je neophodno izvršiti minimaks deo algoritma ne bi li se došlo do optimalnog poteza. Da bi to bilo postignuto, algoritam pravi dve kopije trenutnog stanja table, i u jednoj dovršava lanac, dok u drugoj prepušta lanac protivniku.
6. Ova izmenjena stanja table se prosleđuju rekurzivnoj funkciji koja onda izvršava istu logiku, a za rezultat vraća dvevrednosti koje predstavljaju maksimilan broj poena koje protivnik može da osvoji ukoliko krene od dva prosleđena stanja table.
7. Pošto algoritam bira potez koji protivniku donosi manje poena, biće odabrana manja od te dve vrednosti, i na osnovu toga odlučeno da li je optimalnije dovršiti lanac ili prepustiti ga protivniku.
8. Radi smanjivanja vremenske kompleksnosti, dubina ove rekurzije je ograničena na 5 poteza. Uprkos ovome, što je pokazano kasnije u ovom radu, minimaks algoritam i dalje postiže optimalnije ponašanje u odnosu na druge algoritme.

Algoritmi su izvršavani na procesoru Intel Core i7sa radnom frekvencijom od $3{,}1GHz$ i 4 jezgra, dok je za radnu memoriju korišćeno $16GB$ memorije LPDDR3 tipa radne frekvencije $2{,}1GHz$. Trajanje izvršavanja simulacije bilo je najduže pri korišćenju dva minimaks algoritma, i na tabli veličine 55 kvadrata ono je bilo približno $5s$ po simulaciji. Simulacije na većim tablama nisu vršene, jer je demonstrirana uspešnost minimaks algoritma kada je upoređen sa ostalim algoritmima.

IV. REZULTATI

Pri testiranju ovih algoritama, vršene su simulacije gde je svaki algoritam upoređen sa svakim drugim, kao i sa samim sobom, na tablama različitih veličina. Simulacije sa različitim algoritmima izvršene su dva puta ne bi li bilo provereno da li prvi igrač ima prednost. U svim simulacijama odigrano je 100 partija, i praćena je prosečna razlika u poenima na kraju partije, kao i procenat partija u kojima je prvi igrač pobedio. Algoritam korišćen za prvog igrača je predstavljen u prvoj koloni tabele, dok je protivnikov algoritam predstavljen u prvom redu tabele.

	Nasumični	Pohlepni	Odbrambeni	Minimaks
Nasumični	-0,50 (42%)	-7,72 (1%)	-8,26 (1%)	-6,04 (1%)
Pohlepni	5,99 (99%)	-0,92 (37%)	-5,34 (7%)	-3,68 (13%)
Odbrambeni	7,04 (100%)	3,32 (84%)	-0,65 (44%)	-0,40 (46%)
Minimaks	6,10 (100%)	1,70 (73%)	-1,31 (45%)	-0,40 (49%)

Tabela 1. Rezultati simulacije na tabli veličine 3×3

Kod table veličine 3×3kvadrata već je očigledno da kompleksniji algoritmi imaju veći uspeh. Međutim, zbog veličine table, broj situacija u kojima ima smisla prepustiti lanac protivniku je mali, pa tako minimaks pristup nije primetno bolji od pohlepnih algoritama.

	Nasumični	Pohlepni	Odbrambeni	Minimaks
Nasumični	-0,62 (49%)	-22,88 (0%)	-23,56 (0%)	-19,88 (0%)
Pohlepni	22,72 (100%)	-0,65 (54%)	-13,10 (4%)	-12,56 (3%)
Odbrambeni	23,48 (100%)	11,44 (94%)	0,44 (55%)	-3,28 (29%)
Minimaks	20,48 (100%)	12,32 (98%)	3,12 (66%)	0,04 (45%)

Tabela 2. Rezultati simulacije na tabli veličine 5×5

Povećavajući tablu do veličine5×5 kvadrata, razlike između algoritama postaju još očiglednije, pa je tako svaki algoritam imao veći procenat pobeda kada je uparen sa jednostavnijim algoritmom. Minimaks pristup ima najbolje rezultate, ali i dalje nije nepobediv, posebno kada je uparen sa odbrambenim algoritmom.

	Nasumični	Pohlepni	Odbrambeni	Minimaks
Nasumični	0,00 (50%)	-46,48 (0%)	-46,56 (0%)	-42,34 (0%)
Pohlepni	6,60 (100%)	0,30 (48%)	-26,18 (1%)	-27,2 (0%)
Odbrambeni	46,74 (100%)	24,86 (99%)	-1,47 (46%)	-11,32 (16%)
Minimaks	42,84 (100%)	27,58 (99%)	11,60 (87%)	-3,48 (41%)

Tabela 3. Rezultati simulacije na tabli veličine 7×7

Sa tablom veličine 7×7kvadrata dolazimo do trenutka gde je minimaks algoritam u mogućnosti da odnese pobedu u većini partija, pa ekstrapolacijom možemo zaključiti da bi razlike u poenima na većim tablama takođe bile veće. Performanse ostalih algoritama su takođe u skladu sa očekivanjima.

	Nasumični	Pohlepni	Odbrambeni	Minimaks
Nasumični	0,74 (53%)	-20,34 (0%)	-20,00 (0%)	-17,54 (0%)
Pohlepni	18,94 (100%)	-0,56 (35%)	-11,34 (2%)	-10,34 (2%)
Odbrambeni	19,88 (100%)	10,92 (92%)	-0,56 (38%)	-1,26 (40%)
Minimaks	17,58 (100%)	9,28 (91%)	1,14 (48%)	-0,42 (40%)

Tabela 4. Rezultati simulacije na tabli veličine 2×11

Takođe su izvršene simulacije na tablama koje nisu kvadratnog oblika, ne bi li bila izbegnuta pristrasnost pri analizi rezultata. Rezultati na tabli veličine 2×11 kvadrata su nešto lošiji od rezultata na tabli veličine 5×5, iako obe konfiguracije imaju sličan ukupan broj poena (22 naspram 25).

Jedno moguće objašnjenje jeste da je na tabli veličine 2×11 kvadrata potencijalni raspored lanaca (kako otvorenih, tako i zatvorenih) manje kompleksan nego na tablama kod kojih su obe dimenzije veće, pa tako minimaks algoritam ima manje prostora za manevrisanje i optimizaciju.

	Nasumični	Pohlepni	Odbrambeni	Minimaks
Nasumični	-1,40 (47%)	-52,18 (0%)	-53,18 (0%)	-48,38 (0%)
Pohlepni	52,30 (100%)	0,54 (54%)	-26,34 (1%)	-32,00 (0%)
Odbrambeni	53,14 (100%)	29,54 (100%)	-0,84 (44%)	-13,90 (13%)
Minimaks	48,48 (100%)	30,70 (99%)	11,54 (86%)	-0,44 (45%)

Tabela 5. Rezultati simulacije na tabli veličine 5×11

Zaista, nakon povećavanja kraće dimenzije sa 2 na 5, performanse minimaks algoritma su veoma slične performansama na tabli veličine $7 \times$

7 kvadrata, iz čega možemo zaključiti da je za ovaj algoritam neophodno dovoljno prostora ne bi li se javio veliki broj lanaca raznovrsnih oblika.

V. ZAKLJUČAK

Minimaks algoritam u ovom radu koristi dosta jednostavnu implementaciju koja se oslanja isključivo na rekurziju ograničene dubine ne bi li bila demonstrirana njegova efikasnost, ali to znači da bi upotrebom nešto kompleksnijih pristupa bilo moguće ostvariti još veću razliku u poenima.

Na primer, moguće je uvesti alfa-beta odsecanje kojim bismo eliminisali određena podstabla tokom rekurzije za koja je očigledno da nisu optimalno rešenje. Smanjivanjem ukupnog broja čvorova bi nam takođe omogućilo da povećamo maksimalnu dubinu rekurzije, a možda i da u potpunosti uklonimo ovo ograničenje i oslonimo se isključivo na alfa-beta odsecanje za kontrolisanje vremenske složenosti algoritma.

Dalja unapređenja performansi ovih algoritama mogla bi biti postignuta čuvanjem stanja table, odnosno prethodnih odluka, između poteza. Umesto analize čitave table pri svakom potezu, možemo čuvati stanje table tokom čitave igre i inkrementalno ga menjati na osnovu poteza igrača, a pritom koristiti naprednije (tj. efikasnije) strukture podataka za čuvanje određenih činjenica o igri ili izvršenih i planiranih poteza.

Najzad, u [6] i [7] su objašnjena određena svojstva igre koja mogu biti iskorišćena za bolju optimalnost algoritama, dok je u [8] objašnjen nešto drugačiji (ali i dalje uspešan) pristup koji je baziran na genetskim algoritmima.

LITERATURA

[1] H. Bottomley, "How many Tic-Tac-Toe (noughts and crosses) games are possible?," 2018. Available: http://www.se16.info/hgb/tictactoe.htm.

[2] A. Bhatt, P. Varshney, K. Deb, "Evolution of No-loss Strategies for the Game of Tic-Tac-Toe,"in *Kanpur Genetic Algorithms Laboratory Reports.* Indian Institute of Technology, Kalyanpur, India, 2007. Available: https://www.iitk.ac.in/kangal/papers/k2007002.pdf.

[3] M. J. Osborne, A. Rubinstein,"A Course in Game Theory". Cambridge, MA: MIT, 1994.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, ""Introduction to Algorithms," 3rd ed. Cambridge, MA: MIT, 2009.

[5] S. J. Russel, P. Norvig, "Artificial Intelligence: A Modern Approach," 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2003, pp. 163–171.

[6] S. Li, D. Li, X. Yuan, "Research and Implementation of Dots-and-Boxes Game System," in *Journal of Software*, vol. 7, no. 2. San Bernardino, CA: Academy Publisher, 2012, pp. 256–262.

[7] J. K. Barker, R. E. Korf, "Solving Dots-And-Boxes," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. Palo Alto, CA: The AAAI Press, 2012, pp. 414–419.

[8] T. Bossomaier T, A. Knittel,M. Harré, A. Snyder,"An evolutionary agent approach to Dots-and-Boxes," in *SEA 2006- Software Engineering Applications. 10th IASTED International Conference on Software Engineering and Applications*. Calgary, AB: ACTA Press, 2006, pp. 54.

ABSTRACT

In this paper, the two-player game "Dots and boxes" is presented and several different algorithms for playing this game are analyzed. The main approach, which aims to simulate the behaviour of advanced players, was implemented by using the minimax algorithm, whereas the other algorithms, implemented for comparison purposes, aim to simulate players of different, smaller skillsets. These algorithms were compared by simulating them on game boards of varying sizes and those results are presented here. Besides simulating the algorithms, this paper also covers several approaches that could further improve both the efficiency and the success rate of these algorithms.

APPLICATION OF THE MINIMAX ALGORITHM IN THE GAME "DOTS AND BOXES"

Nenad Božidarević