Nikola Jovanović

Upoređivanje efikasnosti algoritama za snalaženje u lavirintu

Poređeni su A, težeći, intuitivni i slučajni algoritam koje agenti mogu da koriste za snalaženje u lavirintu. A* algoritam služi za određivanje optimalnog puta do cilja. Ostala tri algoritma su posebno osmišljena tako da agenti uče konfiguraciju lavirinta dok ga obilaze. Težeći algoritam podrazumeva da je agentu poznat smer ka cilju i on se kreće u tom smeru. Slučajni algoritam u svakom potezu nasumično bira smer kretanja. Intuitivni algoritam, zasnovan na kombinaciji težećeg i A* algoritma, reprezentuje čovekov način snalaženja u lavirintu. U ovom slučaju agenti znaju koordinate cilja i imaju vidokrug u kome im je konfiguracija terena poznata. Efikasnost pomenutih algoritama je poređena imajući u vidu najkraći put koji pronalazi A*. Pokazano je da je intuitivni algortam znatno efikasniji od težećeg i slučajnog algoritma dajući rezultate približne optimalnom A* algoritmu. Odnos efikasnosti različitih algoritama ostaje konstantan pri variranju različitih parametara.*

Uvod

Pod lavirintom se podrazumeva prostor ispunjen preprekama koje su ili nepremostive ili usporavaju kretanje. Lavirint je moguće predstaviti celobrojnom matricom, gde element matrice određuje stepen prohodnosti na datoj poziciji u prostoru. Postoji veliki broj algoritama za pronalaženje puta u lavirintima koji se razlikuju po ulaznim parametrima kao što su smer cilja, koordinate cilja i poznavanje pozicija prepreka. Uobičajeno rešenje za snalaženje u lavirintu je A* (A star) algoritam, koji služi za nalaženje najkraćeg puta od tačke A do tačke B u potpuno poznatim lavirintima (Dechter i Judea 1985). Međutim, često informacije o konfiguraciji celog prostora nisu dostupne. U takvim situacijama agent koji se kreće po lavirintu može imati različite načine otkrivanja lavirinta, na primer vidi teren samo u određenom vidokrugu ili se samo kreće po lavirintu i pamti pređena polja.

Jedan od takvih algoritama koji je korišćen u ovom radu je težeći (seeking) algoritam. On radi tako što znajući koordinate cilja usmerava agenta direktno ka njemu. Tokom potrage, on pamti putanju kojim se kretao, kako bi izbegao da dva puta krene istim putem. Takođe, on ponekad odabira i neke nasumične pokrete. O njemu će biti reči u posebnom poglavlju.

Drugi sličan algoritam je posebno osmišljen u ovom radu. Baziran je na težećem algoritmu, koji sada ima i informacije o izgledu lavirinta u određenom vidokrugu. Kako bi maksimalno iskoristio ovu prednost, on za pretragu unutar poznate oblasti koristi verziju A* algoritma.

Implementiran je i slučajni algoritam koji u svakom potezu potpuno nasumično odabira smer u kome će se kretati. On je zamišljen da pruži ideju efikasnosti najgoreg mogućeg rešenja.

Cilj ovog rada je upoređivanje intuitivnog algoritma sa preostala tri u slučajevima lavirinata različitih površina i načina generisanja.

Opis korišćenih algoritama

Agent se kreće u matrici tako što iz trenutnog polja može preći na proizvoljno prohodno polje u Murovom susedstvu. Težina prelaza sa jednog na drugo polje računa se kao aritmetička sredina stepena prohodnosti tih polja ukoliko je agent

Nikola Jovanović (1996), Novi Beograd, Antifašističke borbe 2/14, učenik 3. razreda Devete gimnazije „Mihailo Petrović – Alas”

MENTOR: Miloš Savić, Prirodno-matematički fakultet Univerziteta u Novom Sadu

napravio pomeraj samo po *x* ili samo po *y* osi. Najlakše prohodno polje ima težinu jedan, dok se za potpuno neprohodna polja može smatrati da imaju težinu neograničeno. Veća vrednost elementa u matrici samim tim označava manji stepen prohodnosti. Ukoliko se agent kreće ukoso (napravi pomeraj i po *x* i po *y* osi) težina prelaza se dodatno množi sa koren iz dva. Težina nekog puta kroz matricu se računa kao zbir težina prelaza sukcesivnih polja na tom putu. Efikasnost algoritma za kretanje agenta u lavirintu je obrnuto srazmerna težini ukupnog pređenog puta od početne do ciljne tačke.

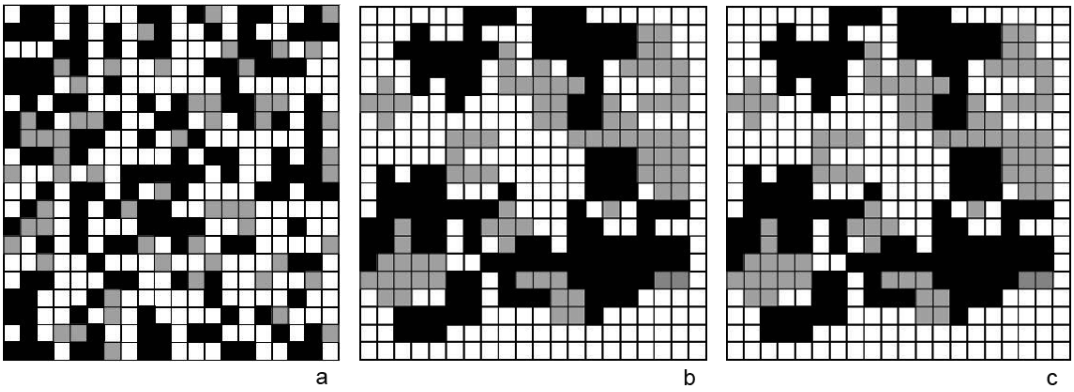
A* algoritam (Dechter i Judea 1985) je nadogradnja Dajkstrinog algoritma (Cormen *et al*. 2001), koja koristi heuristiku za odabir favorizovanih tačaka. Za heurstiku je korišćena euklidska distanca između određenog polja u matrici i cilja. To znači da se svakom polju u matrici dodeli atribut čija vrednost predstavlja rastojanje do cilja. Dajkstrin algoritam funkcioniše tako što se polja u matrici obeležavaju minimalnom težinom puta do početnog polja dokle god se ne obeleži ciljno polje. U početnom trenutku sva polja matrice su neobeležena osim početnog. Dajkstrin algoritam održava red opsluživanja (engl. queue) koji se inicijalizuje početnim poljem. U svakoj iteraciji algoritma iz reda opsluživanja se uklanja prvi element p i posmatra se njegovo susedstvo *S* koje se dodaje na kraj reda opsluživanja. Za svako polje s iz *S* se proverava da li je obeleženo ili ne. Ukoliko je s neobeleženo, tada se ono obeleži sa *T,* gde je *T* težina puta do *p* uvećana za težinu prelaza sa *p* na *s*. Ukoliko je pak *s* obeleženo, tada se proveri da li je *T* manje od trenutno obeležene težine. Ako jeste tada se s obeleži sa *T* jer je pronađen efikasniji put. Ukoliko se u Dajkstrin algoritam inkorporira heuristika tada se elementi u redu opsluživanja sortiraju prema toj heuristici. Drugim rečima, ukoliko je heuristika euklidsko rastojanje do cilja tada je od svih elemenata u redu opsluživanja prvi onaj koji je najbliži cilju. Samim tim se povećava verovatnoća da će efikasniji put biti brže nađen jer se „forsiraju" smerovi kretanja ka cilju

Slučajnim algoritmom agent se kreće u ciklusima. U svakom ciklusu nasumično se određuju smer kretanja i broj poteza (prelaza sa polja na polje) koje će agent načiniti u cilkusu. Zatim agent počinje da izvršava pokrete, a kada naiđe na prepreku ili kad završi predviđeni broj pokreta počinje sledeći ciklus. Ovaj algoritam je potpuno slučajan tako da se ne može predupredi ponavljanje istih pokreta i vrtenje u krug. Da bi se izbeglo vrtenje u krug broj ciklusa je ograničen na 100 000. U simulacijama vršenim u radu ovaj algoritam je uvek nalazio rešenje pre prekoračenja dozvoljenog broja ciklusa.

Težeći algoritam, nasumično bira u kojem od osam mogućih smerova će se agent pokrenuti, ali tako da ima najveću verovatnoću da se pokrene u pravcu cilja. Svaki sledeći par smerova simetričnih u odnosu na prvi odabrani ima duplo manju verovatnoću da bude izabran od prethodnog. Time se dobija mogućnost da se slučajno zaobiđu putevi bez izlaza koji se nalaze na putu ka cilju. Pamte se već posećena polja na koja nije moguć povratak, kako ne bi došlo do toga da program uđe u beskonačnu petlju. Takođe se održava lista svih polja koja je agent već posetio. To je jedini način da se agent vrati na polje na kome je već bio i koristi ga kada nema više prostora za istraživanje i mora da se vrati i pokuša drugačiji put. Tada se njegove predhodne koordinate brišu sa liste.

Pošto su slučajni i težeći algoritmi probabilistički, postoje razlike u rezultatima nakon više pokretanja programa sa identičnim ulaznim parametrima i konfiguracijom lavirinta. Zbog toga se vrši veliki broj pokretanja programa sa istim parametrima i lavirintima. Za upoređivanje se koriste srednje vrednosti dobijenih rezultata.

U ovom radu je dizajniran intuitivni algoritam za snalaženje u lavirintu kod koga agent pronalazi put znajući poziciju cilja i izgled mape u određenom vidokrugu. On se može posmatrati kao kombinacija težećeg i A* algoritma. Ovaj algoritam nije probabilistički, tj. nema nasumičnih poteza i za iste konfiguracije lavirinata uvek pronalazi isti put. Parametri ovog algoritma su poluprečnik vidokruga, koordinate cilja i broj N koji označava broj tačaka koje su kandidati da budu označene kao privremeni cilj. Tačke se biraju tako što se na obodu dela lavirinta koji je već istražen pronalazi N tačaka koje su najbliže cilju po euklidskoj distanci. Zatim se pomoću A* algoritma nalazi dužina puta do svake od njih i ona se sabira sa njihovim euklidskim distancama do cilja. Odabira se tačka sa najmanjim zbirom, što intuitivni algoritam čini gramzivom (engl.



Slika 1. a – nasumično generisan lavirint, b – lavirint sa predefinisanim objektima, c – Primov algoritam.

Figure 1. a – random generated maze, b – maze with predefined objects, c – Prim's algorithm.

greedy) heuristikom. Nakon što je određena pozicija ka kojoj treba da se kreće, agent se pokreće u smeru koji je A* odabrao. Pamti se pređeni put i koordinate novog polja se dodaju na stek. To je bitno zato što u toku simulacije agent može da bude primoran da se vrati nazad, a to mu je onemogućeno obeležavanjem polja na kojima se već nalazio. Na prethodno polje moguće je vratiti se samo skidanjem koordinata tog polja sa vrha steka. Nakon kretanja, sve radnje se ponavljaju za novu poziciju.

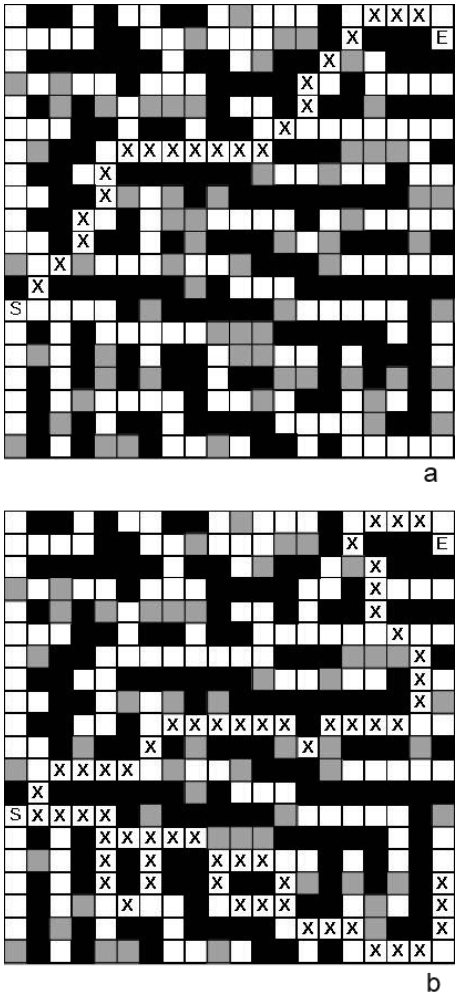
Eksperiment

Svako polje lavirinta može da bude u jednom od tri stanja. Prazno polje je težine 1, usporavajuće polje je težine 3, a za nepremostivu prepreku se može smatrati da ima težinu neograničeno. Težine prelaza između dva polja se računaju na način opisan u prethodnom poglavlju.

Lavirint je generisan na tri načna. Prvi način podrazmeva da se po praznoj matrici nasumično postavljaju nepremostive ili usporavajuće prepreke površine jednog polja matrice (slika 1a). Drugi način podrazumeva da se po matrici raspoređuje nekoliko različitih oblika, od kojih je svaki površine više polja u matrici. Nasumično se određuje koji će oblik biti postavljen, njegove dimenzije, težina polja od kojih je sastavljen i njegove koordinate u matrici. Postavlja se veliki broj takvih objekata, koji mogu i da se preklapaju, tako da se dobiju kompleksniji oblici (slika 1b). Treći način je korišćenje randomizovanog Primovog algoritma (Prim 1957). Pošto ovaj način generiše prave lavirinte u radu je na njega stavljen poseban akcenat. On je implementiran tako da se počinje od matrice u kojoj svaki element predstavlja nepremostivu prepreku, a zatim se od nasumično odabrane tačke grana prolaz. Svako novo polje prolaza se dodaje u listu polja. U svakoj iteraciji se nasumično bira tačka iz liste i proverava se da li se može uz nju dodati nova tačka tako da ona ne poveže dve nesusedne, već dodate tačke. Ukoliko nije moguće dodati novu tačku, trenutna se briše iz liste. Kada lista ostane prazna lavirint je formiran (slika 1c).

Algoritmi su implementirani u programskom jeziku Python. Izrađene su pojedinačne funkcije za svaki od algoritama i njima su prosleđivani isti parametri radi kasnije analize. Program je pokretan sa različitim kombinacijama parametara i računate srednje vrednosti dobijenih dužina pređenih puteva. To je rađeno da bi se umanjio uticaj nasumično određivanih parametara. Promenljivi parametri su površina lavirinta, poluprečnik vidnog polja i način generisanja lavirinta. U fajl se ispisuju posebni slučajevi koje program prepoznaje po lošem rezultatu intuitivnog algoritma (slike 2a i 2b). Tu se pamte svi parametri koji su potrebni za analizu i rekonstrukciju slučaja.

Simulacija je vršena tako što je program pokretan više puta sa istim ulaznim vrednostima, ali na različitim lavirintima, a zatim sa izmenjenim ulaznim vrednostima. Identične simulacije su izvršeni za sva tri načina definisanja lavirinta.



Slika 2. a – optimalno rešenje, b – rešenje sa velikom greškom.

Figure 2. a – optimal solution, b – solution with a big error.

Kako su nasumučni algoritam i težeći algoritam probabilistički, njihova efikanost je računata korišćenjem srednjih vrednosti efikanosti za sve simulacije sa istim konfiguracijama lavirinata.

Simulacije su izvršavane za sva tri načina definisanja lavirinta. U svakoj simulaciji je merena efikasnost algoritma za određenu kombinaciju parametara. Kretanje agenata je simulirano na kvadradnim lavirintima različite površine pri čemu je stranica kvadrata varirana po formuli:

$$L = X \cdot A$$

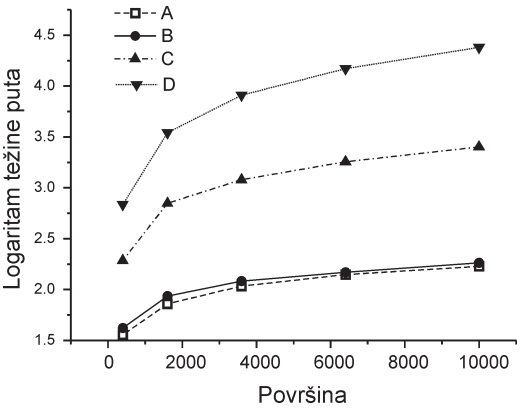
gde je L stranica kvadrata, X prirodan broj variran od 1 do 5, a A najmanja dužina stranice koja je iznosila 20 polja. Parametar poluprečnik vidokruga intuitivnog algoritma je variran po formuli:

$$R = \frac{L}{2 \cdot N}$$

gde je R poluprečnik, L dužina stranice matrice, a N prirodan broj variran od 1 do 5. Broj koraka u nasumičnom algoritmu je određivan kao nasumičan boroj od 1 do 10. Broj tačaka na obodu istraženog dela lavirinta koji se koristi kod intuitivnog algoritma je bio fiksiran na 10.

Rezultati

Rezultati simulacija su pokazali da je odnos efikasnosti svih korišćenih algoritama ostajao konstantan pri menjanju parametara kao što su površina lavirinta ili način generisanja lavirinta. Slučajni algoritam se svuda pokazao ubedljivo najneefikasnijim. Težeći algoritam se pokazao efikasnijim, ali ipak lošijim od preostala dva. Težine puteva koje je nalazio intuitivni algoritam veoma su bliske opimalnim (minimalnim) vrednostima koje je izračunavao A*. Vizualizovan

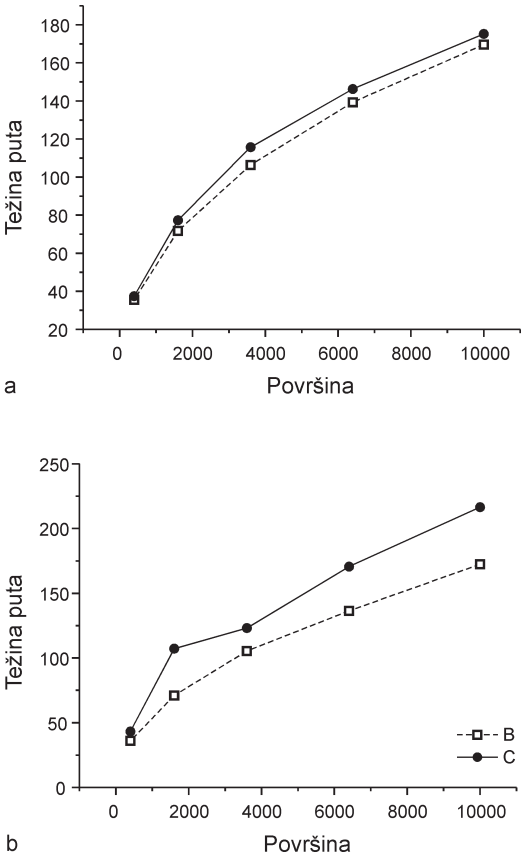


Slika 3. Srednje vrednosti rezultata sva četiri algoritma u odnosu na povećanje površine lavirinta: A – A*, B – intuitivni, C – težeći, D – nasumični

Figure 3. All four algorithms average values in relation to increasing maze area: A – A*, B – special intuitive algorithm, C – seeking algorithm, D – random walk algorithm

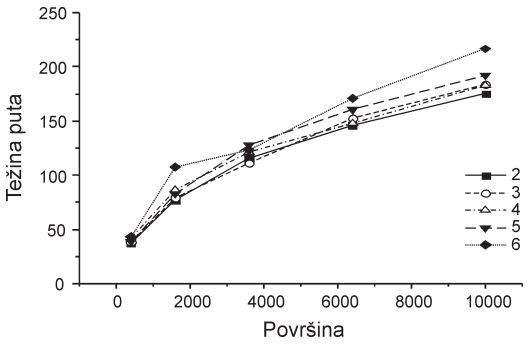
odnos prosečnih težina puteva koje su nalazili pomenuti algoritmi, kao i brojne vrednosti mogu se očitati sa grafika na slici 3.

Dužine puteva koje je pronalazio intuitivni algoritamm za velike vidokruge bile su veoma blizu optimalnim. Sa smanjenjem vidokruga razlika rezultata ituitivnog i A* algoritma se povećavala. Odnosi težina puteva intuitivnog i A* algoritma za maksimalni i minimalni vidokrug prikazani su na slici 4.



Slika 4. Srednje vrednosti rezultata za A* i intuitivni algoritam. U prvom slučaju poluprečnik vidokruga intuitivnog algoritma je polovina stranice lavirinta. U drugom sulčaju poluprečnik vidokruga intuitivnog algoritma je šestina stranice lavirinta.

Figure 4. A* and intuitive algorithm's average values. In the first case intuitive algorithms's sight range is half of the maze side. In the second case intuitive algorithm's sight range is one sixth of the maze side.



Slika 5. Efikasnost intuitivnog algoritma u zavisnosti od poluprećnika vidokruga. Poluprečnik je predstavljan kao dužina stranice lavirinta podeljena sa *X*.

Figure 5. Intuitive algorithm's efficiency in relation to the sight range. Sight range is given as maze side divided by *X*.

Primećeno je da je za veće poluprečnike vidokruga grafik težina puteva koje nalazi intuitivni algoritam pravilniji, dok se skraćivanjem vidokruga na grafiku pojavljuju neočekivana odstupanja. Slika 5 prikazuje rezultate intuitivnog algoritma za sve vidokruge za koje su vršena merenja.

Zaključak

U radu je ispitivana efikasnost intuitivnog algoritma za kretanje u lavirintu, u odnosu na algoritme čijom kombinacijom je osmišljen (A* i težeći) i slučajnog algoritma koji predstavlja rešenje u najgorem mogućem slučaju.

Slučajni algoritam je, kao što je i očekivano, dolazio do ubedljivo najlošijih rešenja. Iako vidokrug nije mogao da pomogne u slučajevima kada agent pokuša da skrene u put bez izlaza u kome prepreka nije u vidokrugu, ipak je pomagao u izbegavanju sitnih grešaka. Velika razlika između rezultata težećeg i intuitivnog algoritma objašnjava se time što je vidokrug omogućavao intuitivnom algoritmu da na puno mesta uštedi po malo poteza, tako da je to dovelo do velike ukupne uštede.

Pokazano je da za lavirinte male površine intuitivni algoritam ima veoma blisku efikanost

optimalnom rešenju, dok je težeći algoritam nešto lošiji.

Intuitivni algoritam dobija najveću prednost u odnosu na težeći u potezu kada cilj ulazi u vidokrug agenta. Tada više nije potrebno da se primenjuje ceo algoritam, već se koriti A* da bi se izračunala preostala udaljenost do cilja. Male razlike između efikasnosti intuitivnog algoritma za različite dužine vidokruga posledica su činjenice da je korist od vidokruga najviše izražena na kraju simulacije.

Za uslove koji su korišćeni u eksperimentu, svi algoritmi su pokazali zanemarljive razlike u efikasnosti pri testiranju sa različitim načinima generisanja lavirinta.

Literatura

Cormen T., Leiserson C., Rivest R, Stein C. 2001. *Introduction to Algorithms* (second ed.). Section 24.3: Dijkstra's algorithm. McGraw–Hill, str. 595–601.

Dechter R., Judea P. 1985. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, **32** (3): 505.

Prim R. C. 1957. Shortest connection networks and some generalizations. *Bell System Technical Journal*, **36**: 1389.

Nikola Jovanović

Comparing Efficiency of Algorithms for Navigating through the Labyrinth

This paper presents a comparison between several methods of dynamic labyrinth walk-through. The first algorithm is originally designed in this paper. Based on the combination of a greedy heuristic and the A* algorithm, it mimics intuitive, human-like exploring behavior. That is achieved by providing the algorithm with information about the maze configuration. in particular view range and the goal direction. The second one realizes a random walk oriented toward the goal. The efficiency of previously mentioned algorithms is compared considering the shortest path found by the A* algorithm which was provided by the complete configuration of the maze. A completely stochastic algorithm which realizes a random walk through the maze is also included in the comparative analysis in order to estimate the worst case efficiency. The results of simulations on random generated mazes show that the algorithm which mimics intuitive human-like behavior outperforms probabilistic algorithms giving results close to the optimal.