Nikola Jovanović
Novi Beograd, Antifašističke borbe 2/14
1996. godište, IV razred, Deveta gimnazija „Mihailo Petrović – Alas“

Upoređivanje efikasnosti algoritama za snalaženje u lavirintu

Ovaj rad se bavi poređenjem efikasnosti više algoritama koje agenti mogu da koriste za snalaženje u lavirintu. Poređena su 4 algoritma, jedan od njih je A* koji služi za nalaženje optimalng rešenja. Ostala tri algoritma su posebno dizajnirana tako da agenti uče konfiguraciju lavirinta dok ga obilaze.

Intuitivni algoritam, zasnovan na kombinaciji gramzive strategije i A* algoritma, je namenjen da reprezentuje čovekov način snalaženja u lavirintu. U ovom slučaju agenti znaju koordinate cilja i imaju vidokrug u kome im je konfiguracija terena poznata. Težeći algoritam podrazumeva da je agentu poznat smer ka cilju i on se kreće u tom smeru. Efikasnost pomenutih algoritama je poređena imajući u vidu najkraći put koji pronalazi A*. Algoritam koji realizuje kompletno nasumično kretanje je takođe uključen u komparativnu analizu kako bi se ocenila efikasnost u najgorem slučaju. Lavirinti su generisani na 3 načina, dva načina su bila varijente nasumičnog raspoređivanja prepreka po praznom prostoru, dok je treći način bio generisanjem pravog lavirinta.

Zaključeno je da je intuitivni algortam znatno efikasniji od probabilističkih algoritama dajući rezultate približne optimalnim. Pokazano je da način generisanja lavirinata ne utiče mnogo na rezultate izračunavanja.

1. Uvod

Postoji veliki broj algoritama za pronalaženje puta u lavirintima koji se razlikuju po ulaznim parametrima kao što su smer cilja, koordinate cilja, poznavanje pozicija prepreka. 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. 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. Informacije o konfiguraciji celog prostora nisu uvek dostupne. 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. Zbog toga se uvode težeći i intuitivni algortam. Težeći algoritam usmerava agenta pravo ka cilju i ne dozvoljava mu da prođe dva puta istim putem. Intuitivni algoritam podrazumeva da agent ima vidokrug unutar koga vidi konfiguraciju lavirinta, to mu pomaže da odredi efikasniji put ili da zaobiđe slepi put. Efikasnost algoritma je ukupno vreme koje je potrebno agentu da krenuvši iz starta izuči lavirint i dođe do cilja. Cilj rada je upoređivanje efikasnosti navedenih algoritama sa efikasnošću A* i nasumičnog algoritma. A* predstavlja optimalno rešenje, a nasuumični algiritam rešenje u najgorem slučaju. Upoređivana je efikasnost algoritma u različitim situacijama i traženi su posebni slučajevi gde su određeni algoritmi veoma efikasni ili neefikasni.

2. 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 se računa

kao aritmetična sredina stepena prohodnosti tih polja ukoliko je agent napravio pomeraj samo po *x* ili samo po *y* osi. Potpuno prohodno polje se u matrici obeležava sa 1 dok potpuno neprohodno polje (prepreka) ima vrednost beskonačno. 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(2) je nadogradnja Dajkstrinog algoritma(1) 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 polja. Dajkstrin algoritam održava red opsluživanja (engl. *queue*) koje se inicijalizuje početnim poljem. U svakoj iteraciji algoritma se iz reda opsluživanja 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 se agent kreće u ciklusima. U svakom ciklusu se nasumično određuje smer kretanja i koliko poteza (prelaza sa polja na polje) 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 ti ponavljanje istih pokreta i vrtenje u krug. Da bi se izbegao prethodni efekat broj ciklusa je ograničen na maksimalnu vrednost celobrojnog tipa podataka u programskom jeziku. U simulacijama vršenim u radu ovaj algoritam je uvek nalazio rešenje pre prekoračenja dozvoljenog broja ciklusa.

Težeći algoritam, nasumično bira u kojem smeru će se agent pokrenuti, ali tako da ima najveću verovatnoću da se pokrene u pravcu cilja, zatim duplo manju verovatnoću za smerove levo i desno od prvog, četri puta manju za sledeći par smerova i tako dalje. 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.

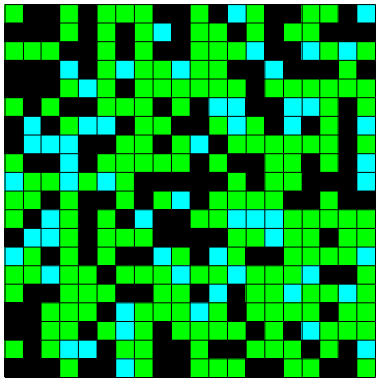
Pošto su ova dva algoritma probabilistička, to za posledicu ima velika odstupanja rezultata nakon više pokretanja programa, tako da se za upoređivanje koriste srednje vrednosti rezulatata nakon velikog broja simulacija sa istim parametrima.

U radu je dizajniran **intuitivni algoritam** za snalazenje u lavirintu kod koga agent pronalazi put znajući poziciju cilja i izgled mape u određenom vidokrugu. Ovaj algoritam

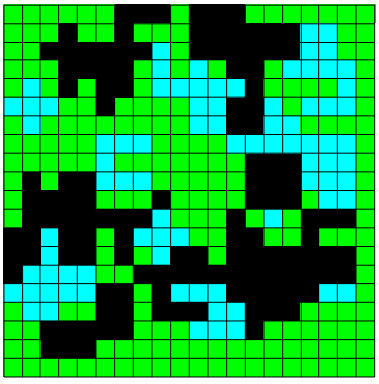
nije probabilistički, tj. nema nasumičnih poteza i za iste ulazne parametre uvek pronalazi isti put . Parametri ovog algoritma su poluprečnik vidokruga, koordinate cilja. On radi tako što na obodu dela lavirinta koji je već istražio, pomoću euklidske distance pronalazi N tačaka koje su najbliže cilju po euklidskoj distanci, gde je N takođe ulazni parametar algoritma . Zatim se pomoću A* nalazi dužina puta do svake od nih i ona se sabira sa njihovim euklidskim distancama do cilja. Odabira se tačka sa najmanjim zbirom, što intuitivni algoritam čini gramzivom (engl. 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 je 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.

3. Eksperimenti i rezultati

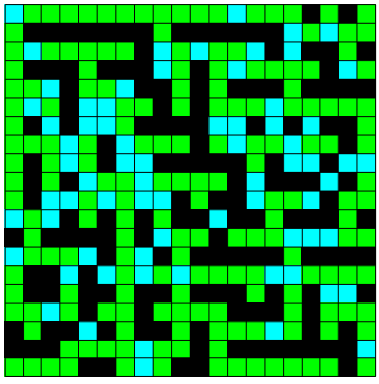
Lavirint je generisan na tri načna. U prvom se po praznoj matrici nasumično postavljaju nepremostive ili usporavajuće prepreke površine jednog polja matrice (Primer 1). U drugom se po matrici raspoređuje nekoliko različitih oblika površine više polja u matrici. Nasumično se određuje koji će oblik biti 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 (Primer 2). U trećem načinu se koristi randomizovani Primov algoritam(3). On je implementiran tako da se počinje od matrice u kojij 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 (Primer 3).



Primer 1 – nasumično generisan lavirint



Primer 2 – predefinisani objekti

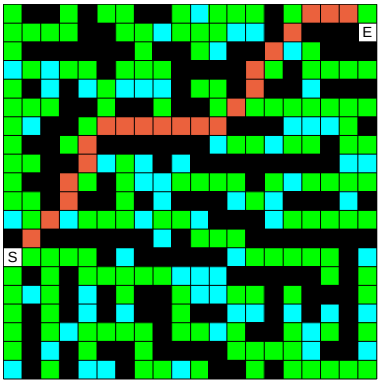


Primer 3 – Primov algoritam

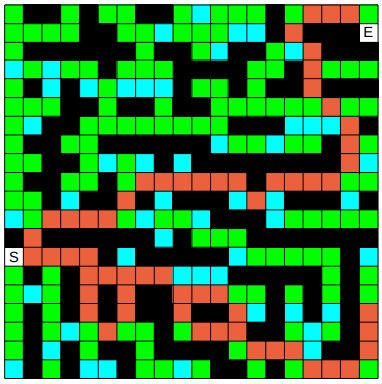
Za simulacije korišćen je programski jezik Python. Izrađene su pojedinačne funkcije za svaki od algoritama i njima su prosleđivani isti parametri radi kasnije analize rezultata. Program je pokretan sa različitim kombinacijama parametara i izračunavane su srednje vrednosti dobijenih dužina pređenih puteva. 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 1 i 2). Tu se pamte svi parametri koji su potrebni za analizu i rekonstrukciju sučaja.



Slika 1 – optimalno rešenje



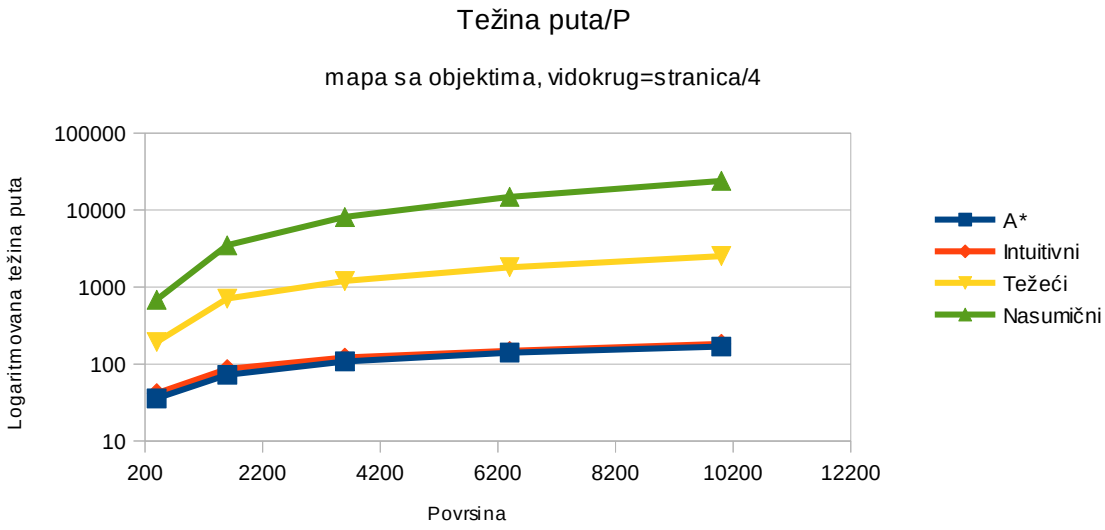
Slika 2 – rešenje sa velikom greškom

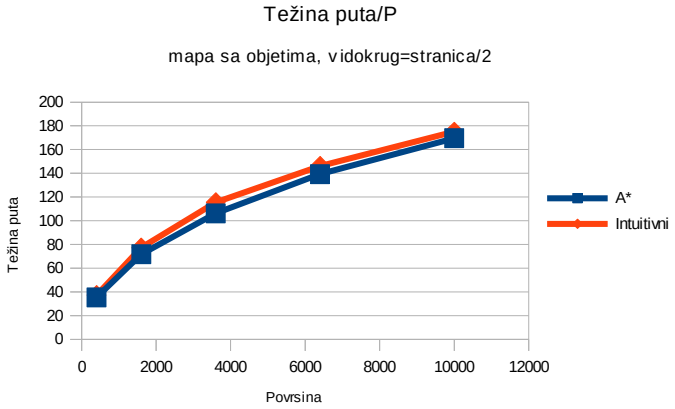
Simulacija je vršena tako što je program pokretan više puta sa istim ulaznim vrednostima, ali na različitim lavirintima, a zatim sa izmenjenim ulaznim vrednostima. Identične simulacije su izvršeni za sva tri načina definisanja lavirinta. Kako su nasumučni algoritam i težeći algoritam probabilistički, njihova efikanost se meri uzimajući srednju vrednost efikanosti za sve simulacije sa istim ulaznim parametrima.

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 * A, gde je L stranica kvadrata, X prirodan broj, a A najmanja dužina stranice. Parametar poluprečnik vidokruga intuitivnog algoritma je variran po formuli R = L / (2*N), gde je R poluprečnik, L dužina stranice matrice, a N prirodan broj. U simulacijama je najmanja dužina stranice bila 20, za X i N su korišćei brojevi 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.

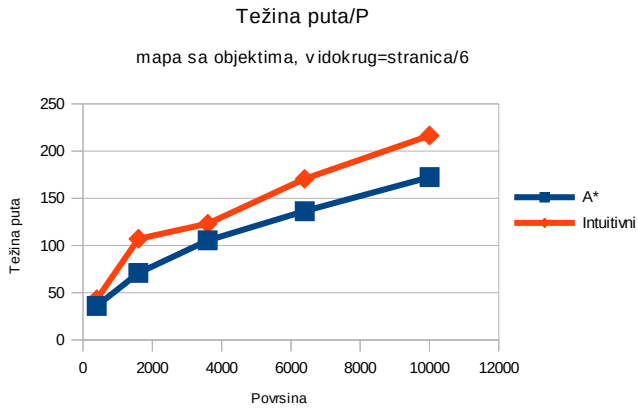
5. Rezultati

Rezultati za sve primere pokazali su da težine puteva koje nalaze slučajni i težeći algoritam rastu mnogo brže nego dužine puteva druga dva algoritma (Grafik 1). Dužine puteva koje nalazi intuitivni algoritam su za velike vidokruge veoma malo iznad optimalnih, a razlika se povećava sa smanjenjem vidokruga (Grafici 2 i 3).



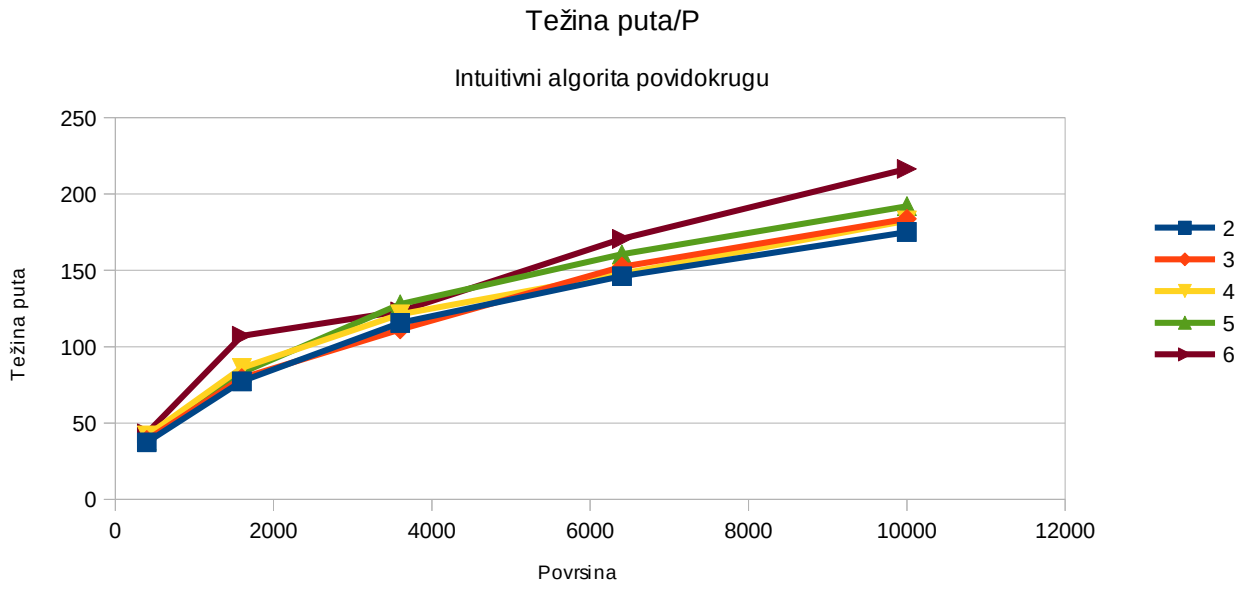


Grafik 2 – veliki vidokrug



Grafik 3 – mali vidokrug

Primetno je da je za veće poluprečnike grafik intuitivnog algoritma pravilniji, dok se sa manjim vidokruzima pojavljuju velika neočekivana odstupanja (Grafik 4).



Grafik 4 – Efikasnost intuitivnog algoritma u zavisnosti od polprečnika vidokruga
Poluprečnik je predstavljan kao dužina matrice / x

6. Zaključak

U radu je ispitivana efikasnost težećeg i intuitivnog algoritama za kretanje u lavirintu, u odnosu na optimalno rešenje i rešenje u najgorem slučaju. Pokazano je da za lavirinte male površine intuitivni algoritam ima veoma blisku efikanost optimalnom rešenju, dok je težeći algoritam nešto lošiji.

Pretpostavka je da intuitivni algoritam dobija najveću prednost u potezu kada cilj ulazi u vidokrug agenta. Tada više nije potrebno da se primenjuje ceo algoritam, već se koristi A* da bi se izračunala preostala udaljenost od cilja.

Takođe se pretpostavlja da su male razlike između efikasnosti intuitivnog algoritma za različite dužine vidokruga upravo posledica toga da je korist od vidokruga najviše izražena na kraju simulacije.

Predpostavlja se da grafik intuitivnog algoritma gubi pravilan oblik zbog previše malog broja ponavljanja eksperimenta, što nije bilo nemoguće rešiti zbog trajanja izračunavanja.

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

U narednim fazama istraživanja, eksperiment je moguće proširiti uzimajući u obzir:

- lavirinte većih površina,
- lavirinte različitih odosa dužina stranica,
- više algoritama za pretragu.

Literatura:

(1) Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". *Introduction to Algorithms* (Second ed.). MIT Press and McGraw–Hill. pp. 595–601. ISBN 0-262-03293-7

(2) Dechter, Rina; Judea Pearl (1985). "Generalized best-first search strategies and the optimality of

A*". *Journal of the ACM* **32** (3): 505–536. doi:10.1145/3828.3830

(3) R. C. Prim: *Shortest connection networks and some generalizations*. In: *Bell System Technical Journal*, 36 (1957), pp. 1389–1401

Comparing efficiency of algorithms for navigating through the labyrinth

This paper is about comparing several methods of dynamic labyrinth walk-through methods. Namely, we designed two algorithms which assume that the moving agent knows the position of the goal, but does not know the whole configuration of the maze, i.e. the agent learns the configuration as it explores the maze. The first algorithm, based on the combination of a greedy heuristic and the A* algorithm, mimics intuitive, human-like exploring behavior. 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 closed to the optimal.