The 10th Conference on Mathematics in Engineering: Theory and Applications
Novi Sad, December 12-14th 2025

UPOTREBA RAČUNARSKE ANIMACIJE KAO METODE OBUKE IZ OBLASTI ALGORITAMA PRETRAGE[1]

Dunja Španović [2], Daria Varga[2], Bojan Banjac[2] i Lidija Krstanović[2]

https://doi.org/10.24867/META.2025.17 Stručni članak

Sažetak. Ovaj rad prikazuje implementaciju i interaktivnu vizualizaciju pretrage mreže pomoću različitih algoritama za pronalaženje putanje od polazne do ciljne tačke. Kreirani interfejs omogućava definisanje dimenzija mreže unosom broja vrsta i kolona, određivanje pozicija prepreka, kao i izbor polazne i ciljne tačke. Nakon inicijalizacije mreže, implementirani algoritmi pronalaze putanju, pri čemu je svaki korak izvršavanja praćen vizuelnim prikazom i ispisom ključnih informacija o toku pretrage. Cilj rada je da se ilustruju razlike između algoritama pretrage uz jednostavan interfejs koji omogućava njihovu analizu na korisnički definisanim mrežama, u obrazovne i istraživačke svrhe.

AMS klasifikacija (2020): 68T20, 68U05

Ključne reči: algoritmi pretrage, korisnički interfejs, vizualizacija, pretraga u širinu, pretraga u dubinu, Dijkstrin algoritam, A*

1. Uvod

Planiranje putanje predstavlja jedan od ključnih zadataka u oblasti robotike, sistema koji se autonomno kreću, veštačke inteligencije, kao i u razvoju video igara. Njihov osnovni cilj je pronalaženje optimalnog puta između početne i ciljne tačke u zadatom okruženju koje može sadržati statične ili dinamične prepreke. U zavisnosti od konteksta primene, kriterijum optimalnosti može biti najkraće rastojanje, najmanji broj posećenih ćelija mreže, izbegavanje prepreka ili kombinacija ovih faktora.

Prema količini dostupnih informacija tokom izvršavanja, algoritmi za pretragu mogu se klasifikovati na neinformisane i informisane. Neinformisani algoritmi, kao što su pretraga u širinu (eng. *Breadth-First Search, BFS*) i pretraga u dubinu (eng. *Depth-First Search, DFS*), pretražuju mrežu sistematično, bez znanja o poziciji cilja. Nasuprot njima, informisane pretrage, kao što je A*,

[1]Ovaj rad predstavlja unapređenu i proširenu verziju funkcionalnosti razvijenih u okviru master rada „Primena modernih algoritama i metoda u sistemima autonomnih vozila“, autorke Dunje Španović pod mentorstvom vanr. prof. dr Lidije Krstanović.

[2]Departman za opšte discipline u tehnici, Fakultet tehničkih nauka, Univerzitet u Novom Sadu, e-mail: duspanovic@uns.ac.rs — dariavarga@uns.ac.rs — bojan.banjac@uns.ac.rs — lidijakrstanovic@uns.ac.rs

koriste heurističke funkcije kako bi usmerili pretragu ka cilju, sa namerom da se smanji broj posećenih ćelija mreže i ubrza proces pronalaženja putanje.

Za potrebe ovog rada, mreža se definiše kao diskretna podela dvodimenzionalnog prostora na konačan broj vrsta i kolona. Svaka ćelija mreže može se posmatrati kao čvor neusmerenog grafa sa granama jedinične težine ka ćelijama levo, desno, iznad i ispod. Putanja između polazne i ciljne tačke predstavlja niz susednih ćelija mreže, odnosno povezanih čvorova koji ne sadrže prepreke. U ovakvim grafovima, najkraća putanja je ona koja sadrži najmanji broj čvorova između polazne i ciljne tačke [1].

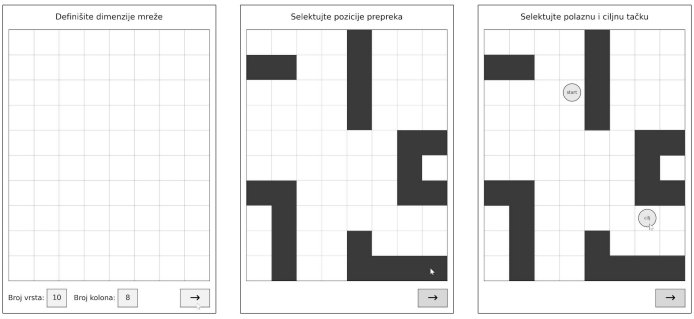
U cilju boljeg razumevanja i poređenja pomenutih pristupa, u ovom radu je implementiran interaktivni korisnički interfejs koji omogućava intuitivnu vizualizaciju procesa pretrage i analizu rezultata na korisnički definisanim mrežama. Na ovaj način korisnik može neposredno da prati tok pretrage, uporedi različite pristupe i uoči razlike u brzini, efikasnosti i karakteristikama svakog algoritma.

2. Implementacija

Implementacija funkcionalnosti izvedena je u programskom jeziku *Python*, uz korišćenje biblioteke *NumPy* za rad sa brojevnim strukturama podataka i *Matplotlib* za vizualizaciju i interakciju sa korisnikom [2]. Time je omogućen prikaz svakog koraka pretrage u realnom vremenu, uz prethodno podešavanje parametara mreže na kojoj se pretraga izvršava.

2.1. Korisnički interfejs

Korisnički interfejs omogućava korisniku da u tri koraka definiše sve parametre potrebne za pokretanje algoritama pretrage (vidi sliku 1). U prvom koraku definiše se veličina mreže unosom broja vrsta i kolona, nakon čega se prikazuje prazna mreža zadatih dimenzija. Zatim, naredni korak obuhvata određivanje pozicija prepreka. Selekcijom pomoću miša prohodna ćelija postaje prepreka i obrnuto, pri čemu se svaka promena istovremeno vizualizuje na mreži. U završnoj fazi biraju se polazna i ciljna tačka pretrage, uz ograničenja koja sprečavaju njihovo preklapanje i selektovanje ćelija koje su u prethodnom koraku postale prepreke. Kada su svi parametri definisani, mreža je spremna za pokretanje algoritama pretrage i uporedni prikaz njihovog izvršavanja u zadatim uslovima.



Slika 1: Kreiran korisnički interfejs za inicijalizaciju mreže

2.2. Implementirani algoritmi i heuristike za pretragu

Algoritmi pretrage implementirani su kao zasebne funkcije koje koriste istu reprezentaciju mreže i principe vizualizacije [3]. Prateće metrike, poput ukupnog broja posećenih ćelija, broja otvorenih i zatvorenih ćelija, kao i dužina pronađene putanje, omogućavaju direktno poređenje njihove efikasnosti.

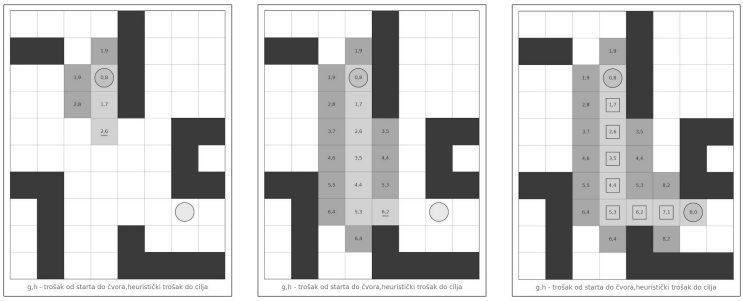
Pretraga u širinu je jedan od osnovnih algoritama za traženje najkraće putanje. Ovaj algoritam istražuje ćelije mreže po nivou udaljenosti od početne pozicije, tako što prvo obrađuje sve ćelije na udaljenosti od jednog koraka, zatim dva, i tako redom [4]. U našoj implementaciji opisani postupak je realizovan pomoću strukture podataka red, koja omogućava obradu ćelija po principu *First In, First Out*. Iterativnim iscrtavanjem mreže, uz bojenje i ispis udaljenosti ćelije od polazne tačke, korisnik može da uoči osnovne osobine ovog algoritma, uključujući širenje pretrage u svim pravcima. Glavna prednost *BFS*-a jeste to što uvek pronalazi najkraći put. Međutim, s obzirom na to da tokom izvršavanja ne koristi informacije o položaju cilja, ovaj pristup može dovesti do velikog broja posećenih ćelija, čime se povećava i vreme procesiranja.

Pretraga u dubinu ima suprotan pristup u odnosu na *BFS*: ona istražuje jedno grananje što je dublje moguće pre povratka i prelaska na sledeće. Implementirani *DFS* se oslanja na strukturu podataka stek kako bi se izbegla upotreba rekurzije i samim tim potencijalno preopterećenje sistema. Pri različitim konfiguracijama mreže, korisnik može zaključiti da *DFS* ne garantuje da je pronađena putanja najkraća i da njen oblik zavisi od redosleda poteza po kojem se obrađuju susedne ćelije (u našem slučaju: gore, dole, levo, desno).

U sistemima gde se određuje putanja u težinskim grafovima sa nenegativnim težinama grana primenjuju se algoritmi koji tokom pretrage prate ukupni trošak pređenog puta $g(n)$, pri čemu n označava ćeliju koja se trenutno obrađuje [5]. Jedan od najpoznatijih među njima jeste Dijkstrin algoritam, koji predstavlja osnovu mnogih savremenih rešenja za planiranje putanje. Za razliku od *BFS*-a koji u grafu sa jednakim težinama pronalazi najkraću putanju od polazne do ciljne tačke samo na osnovu broja čvorova koji čine putanju, Dijkstrin algoritam određuje putanju pomoću prioritetnog reda tako što u svakom koraku bira čvor sa najmanjim poznatim troškom puta. Tokom izvršavanja algoritma prate se dve grupe ćelija: otvorene i zatvorene. Otvorena lista obuhvata sve trenutno dostupne kandidate za dalju ekspanziju pretrage. U njoj se nalaze ćelije koje su otkrivene i za koje je izračunat trenutni trošak puta, na osnovu čega prioritetni red određuje koja ćelija će biti izabrana za narednu obradu. Zatvorenu listu čine ćelije za koje je već potvrđeno da imaju najmanji mogući trošak u određenom trenutku izvršavanja algoritma, pa se zbog toga više ne razmatraju pri daljem širenju pretrage. U vizualizaciji implementiranoj u ovom radu otvorene ćelije prikazane su tamnosivom, dok su zatvorene obeležene svetlijom nijansom sive, čime se omogućava praćenje procesa izbora ćelija tokom pretrage.

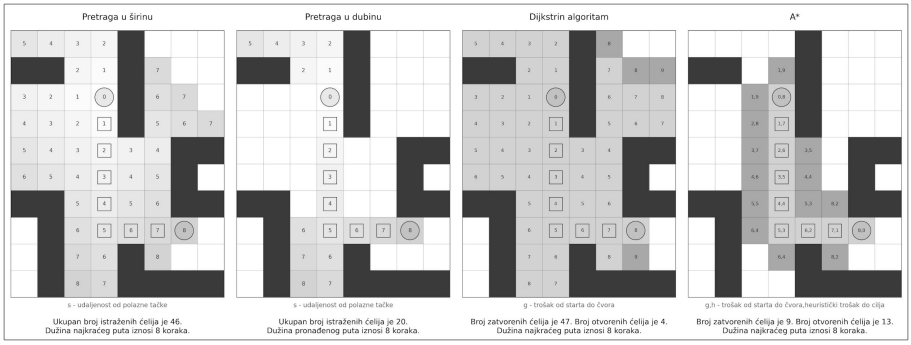
Na mrežama gde sve grane imaju istu težinu, Dijkstrin algoritam daje iste rezultate kao *BFS*, ali uz veće vreme procesiranja usled upotrebe prioritetnog reda. Ipak, njegovo uključivanje u analizu omogućava jasnije poređenje sa A* koji predstavlja proširenje Dijkstrinog pristupa uvođenjem heuristike troška puta do cilja [6].

A* kombinuje stvarni trošak puta od polazne tačke do trenutne ćelije *g(n)* sa heuristikom troška preostalog puta do cilja *h(n)*, formirajući vrednost funkcije evaluacije *f(n) = g(n) + h(n)*, gde *n* označava trenutni čvor. U našoj implementaciji, heuristika se zasniva na Menhetn udaljenosti koja je pogodna u slučajevima kada se kretanje po mreži vrši u četiri smera, dok se ćelije obrađuju pomoću strukture podataka prioritetni red koji omogućava efikasan izbor ćelija sa najmanjom vrednošću funkcije evaluacije. U svakom koraku se vizualizuju obrađeni čvorovi sa istaknutim otvorenim i zatvorenim ćelijama, kao i ispisom njihove *g* i *h* vrednosti (vidi sliku 2). Na ovaj način, korisnik može da prati kako funkcija evaluacije utiče na tok pretrage i da uoči glavnu prednost A* algoritma koja podrazumeva pronalaženje najkraće putanje uz značajno manji broj obrađenih ćelija [7, 8].



Slika 2: Prikaz nekoliko koraka pretrage pomoću A*

Da bi se olakšalo poređenje, koraci izvršavanja sva četiri algoritma vizualizovani su istovremeno, jedan pored drugog, u okviru istog prozora (vidi sliku 3). Ovakav prikaz, uz ispis ključnih metrika poput ukupnog broja posećenih ćelija, broja otvorenih i zatvorenih ćelija, kao i dužina putanje, omogućava korisniku da lakše sagleda prednosti i ograničenja sva četiri pristupa u identičnim uslovima.



Slika 3: Uporedni prikaz rezultata pretrage pomoću *BFS*-a, *DFS*-a, Dijkstrinog algoritma i A*

3. Zaključak

U radu je realizovan uporedni prikaz pretrage mreže u širinu, u dubinu, Dijkstrinog algoritma i A*. Implementirane funkcionalnosti obuhvataju korisnički interfejs za definisanje dimenzija mreže, prepreka, kao i polazne i ciljne tačke, uz prikaz toka pretrage sa odgovarajućim metrikama. Interaktivna priroda kreiranog rešenja čini ga pogodnim za obrazovne svrhe, jer korisniku pruža mogućnost da inicijalizuje okruženje u kojem se pretraga odvija i da istovremeno prati svaki korak izvršavanja sva četiri algoritma. Moguća unapređenja uključuju nove pristupe za traženje putanje, primenu dodatnih heuristika, omogućavanje dijagonalnih poteza prilikom obilaska mreže i uvođenje dinamičnih prepreka.

Literatura

[1] Ivica Bošnjak, Dragan Mašulović, Vojislav Petrović i Ratko Tošić, *Zbirka zadataka iz teorije grafova*, PMF, Departman za matematiku i informatiku, Novi Sad, 2005.

[2] Matplotlib Development Team, *Event handling and picking*, `https://matplotlib.org/stable/users/explain/figure/event_handling.html`, 2025.

[3] Silvester Dian Handy Permana, Ketut Bayu Yogha Bintoro, Budi Arifitama, and Ade Syahputra, *Comparative Analysis of Pathfinding Algorithms A*, Dijkstra, and BFS on Maze Runner Game*, IJISTECH (International Journal of Information System & Technology), vol. 1, no. 2, pp. 1, 2018. DOI: 10.30645/ijistech.v1i2.7

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms, Fourth Edition*, The MIT Press, 2022.

[5] Steven S. Skiena, *The Algorithm Design Manual, Third Edition*, Springer, 2020.

[6] Sharmad Rajnish Lawande, Graceline Jasmine, Jani Anbarasi, and Lila Iznita Izhar, *A Systematic Review and Analysis of Intelligence-Based Pathfinding Algorithms in the Field of Video Games*, Applied Sciences, vol. 12, no. 11, pp. 5499, 2022. DOI: 10.3390/app12115499

[7] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael, *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, 1968. DOI: 10.1109/TSSC.1968.300136

[8] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach, Third Edition*, Pearson, 2010.