Efikasno pretraživanje grafova korišćenjem algoritma A*

Efficient graph searching using A* algorithm

dr Marko Marković, M.Sc Ivan Pantelić, M.Sc Jelena Kaljević, dr Biljana Tešić,
Poslovni fakultet Valjevo, Univerzitet Singdunum

***Sažetak*—Algoritmi pretraživanja omogućavaju da se u skupu rezultata pronađe željeni podatak na osnovu određenog identifikatora. Kao posebno zanimljiva klasa algoritama pretraživanja ističe se A* koji je poznat po preciznosti i performansama koje ostvaruje. Iako su razvijeni algoritmi koji u određenim primenama postižu bolju efikasnost, A* predstavlja jedan od najviše korišćenih. Ovakvi grafovski algoritmi se mogu predstavljati i grafički, tako da se za njihovo razumevanje u savremenom obrazovanju često koriste i interaktivni vizuelni simulatori, a pregled najčešće korišćenih će biti napravljen u ovom radu.**

Ključne riječi – algoritmi pretraživanja, grafovski algoritmi, a*, softverski simulatori

***Abstract* –** Search algorithms make possible to find the desired result in the dataset, based on certain identifiers. As a particularly interesting class of these algorithms, A* is emphasized by its precission and performance. Although more efficient algorihms have been developed for certain applications, A* is one of the most used ones. In order to facilitate understanding, it's possible to graphicaly represent these algorithms. In this way, users have an overview of the activities that take place in an algorithm. Therefore, there is a large number of visual simulators whose overview is also given in this paper.

Keywords – search algorithms, graph algorithms, a*, software simulations

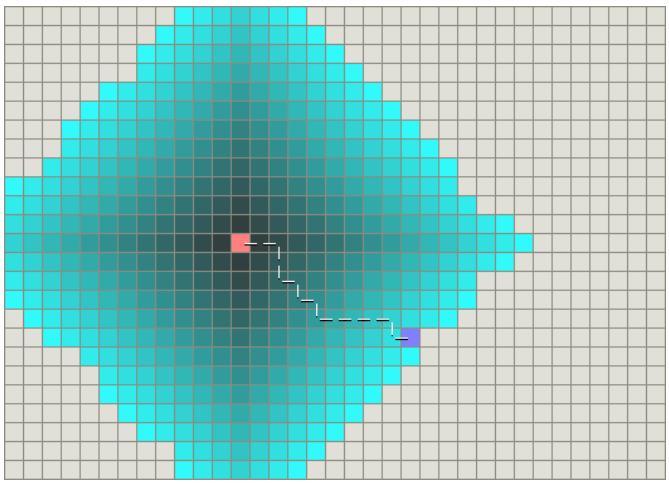
I. UVOD

A* pretraživanje spada u najpoznatiji oblik pretraživanja prvo najbolji (*best-first search*) sa kojim u osnovi deli istu proceduru rada. Spada u najpopularnije algoritme za probleme nalaženje puta jer je fleksibilan i može se koristiti u velikom broju situacija, uključujući i rad sa ogromnim prostorima stanja. Prvi put je predstavljen 1968. godine, a predstavljao je nadogradnju Dijkstrinog[1] algoritma iz 1959. godine (Sl. 1.). [1] Dobre performanse su postignute korišćenjem heurističkih funkcija za procenu minimalnih troškova obilaska grafa.

[1] *Edsger W. Dijkstra* (1930-2002.), holandski naučnik, tvorac poznatog algoritma za nalaženje najkraćih puteva u okviru grafa.

Idealna heuristička funkcija će veoma brzo naći najkraću putanju. Ukoliko nije izabrana najpogodnija heuristička funkcija, algoritam će i dalje nalaziti najkraće putanje, ali će mu trebati više vremena ili će raditi veoma brzo, ali neće uvek nalaziti najbolja rešenja. Dakle, dobra heuristička funkcija mora napraviti kompromis između brzine i preciznosti rada.

Sl. 1. Grafički prikaz rada Dijkstrinog algoritma. Izvor: [8]



II. NAČIN RADA ALGORITMA A*

Algoritam A* u osnovi koristi funkciju f(n) kako bi odredio redosled u kom će se obilaziti graf. Funkcija f(n) predstavlja sumu druge dve funkcije: [2]

$$f(n) = g(n) + h(n).$$

Funkcija g(n) predstavlja funkciju stvarnog troška, od početnog do trenutnog čvora, dok funkcija h(n) predstavlja heurističku funkciju - funkciju koja procenjuje preostalo rastojanje do cilja. Važno je da h(n) bude prihvatljiva heuristika koja nikada neće preceniti trošak stizanja do cilja.[3]

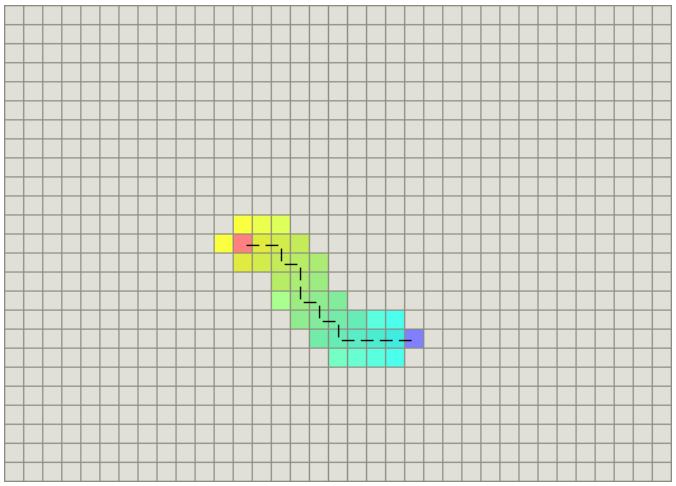
Algoritam A* održava listu čvorova koji se razmatraju za otvaranje (otvorena lista) i listu čvorova koji su već otvarani

DOI: 10.7251/ZRSNG1501051M

(zatvorena lista). U svakom koraku, najbolji čvor iz otvorene liste se pomera u zatvorenu listu, potom se proširuje i njegovi sledbenici se dodaju u otvorenu listu. Ta procedura se nastavlja sve dok se ne dođe do rešenja. Uz konzistentnu heurističku funkciju, čvor koji je proširen i premešten u zatvorenu listu, nikada ne bi trebao da bude ponovo vraćen u otvorenu listu. [4] Sâm algoritam ima sledeću strukturu: [5]

1. Formirati listu parcijalnih putanja. Inicijalno lista sadrži samo jednu putanju nulte dužine koja koja sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste putanja koja dostiže ciljni čvor.
   2.1. Ako je prva putanja dostigla ciljni čvor, ne raditi ništa.
   2.2. Ako prva putanja nije dostigla ciljni čvor, uraditi sledeće:
      2.2.1. Ukloniti prvu putanju iz liste.
      2.2.2. Za svaki sledbenik poslednjeg čvora na uklonjenoj putanji formirati po jednu novu putanju produžujući sledbenikom uklonjenu putanju.
      2.2.3. Za svaku od novodobijenih putanja izračunati ukupnu (kumulativnu) cenu koštanja c kao zbir cena koštanja operatora na toj putanji; za poslednji čvor na putanji izračunati heurističku funkciju h. Funkciju procene f za svaku od novih putanja izračunati kao zbir heurističke funkcije h i cene koštanja putanje c (f = h + c).
      2.2.4. Dodati nove putanje u listu parcijalnih putanja.
      2.2.5. Sortirati listu putanja po rastućim vrednostima funkcije procene f.
      2.2.6. Ako dve ili više putanja iz liste imaju isti poslednji čvor, ukloniti iz liste sve takve putanje osim jedne koja ima najmanju cenu koštanja (princip dinamičkog programiranja).
3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

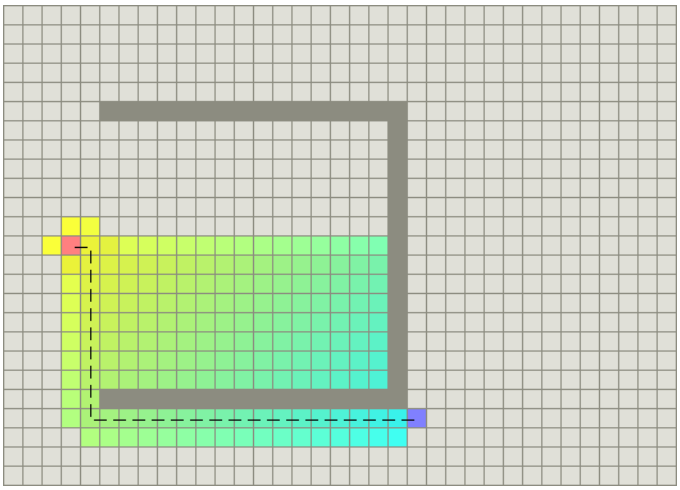
Sl. 2. Grafički prikaz rada algoritma A*. Izvor: [8]



Slika 2. prikazuje rad algoritma A* na istom skupu podataka koji prikazuje i slika 1. Jasno je da je u oba slučaja pronađeno isto rešenje, ali A* to radi sa mnogo manje koraka. Za razliku od algoritma koji je predložio Dijkstra, A* je po efikasnosti mnogo sličniji pohlepnim algoritmima kao što je *best-first.*

Zanimlivo je i da se A* odlično snalazi i u situacijama kada postoji i određena "prepreka". U tim slučajevima će naći prilično efikasan način da je zaobiđe i dođe do tačnog rešenja (Sl. 3.). To se postiže kombinovanjem pristupa koji koristi Dijkstrin algoritam (favorizovanje čvorova bližih početnoj tački) i pohlepni *best-first* (favorizovanje čvorova bližih odredištu).

Sl. 3. Grafički prikaz obilaženja prepreke pomoću algoritma A*. Izvor: [8]



A* je kompletan algoritam koji će pronaći rešenje problema ukoliko ono postoji. Ukoliko ne uspe da pronađe rešenje, tada se može garantovati da ono i ne postoji. A* će naći rešenje uz najmanji mogući trošak, a to će veoma zavisiti od definisane heurističke funkcije i njene procene. [6] U slučaju kada je heuristička funkcija nekonzistentna, algoritam će raditi veoma loše jer će se određeni čvorovi koji su već otvarani otvarati ponovo veliki broj puta. To u najgorem slučaju može rezultovati složenošću od O(2N), gde je N broj različitih čvorova koji su otvarani. [7]

III. PRIKAZ POSTOJEĆIH VIZUELNIH SIMULATORA

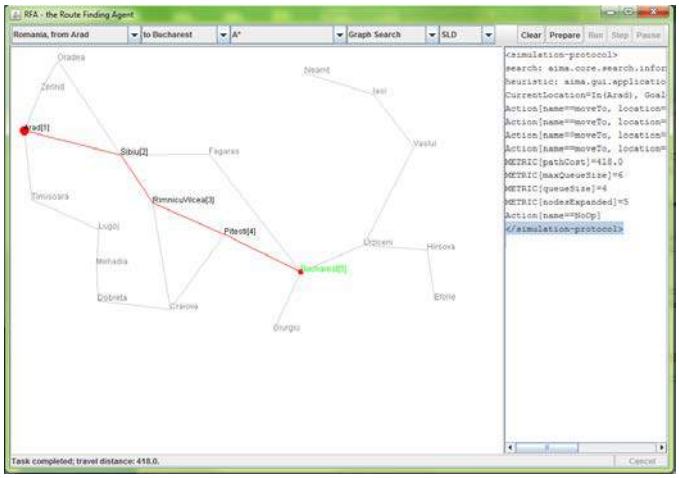
U cilju boljeg razumevanja načina rada algoritma A*, na velikom broju univerziteta širom sveta se koriste softverski vizuelni simulatori. Njihovi načini realizacije i pristup objašnjavanju su izuzetno raznovrsni. U ovoj sekciji će biti prikazana najpoznatija rešenja uz predstavljanje njihovih najvažnijih karakteristika.

Simulator *AIMA3e* (*Artificial Inteligence - A Modern Approach 3rd Edition*) (Sl. 4.) [9] predstavlja dopunu udžbenika *Veštačka inteligencija* koji su napisali *Stuart Russell* i *Peter Norvig*. Najvažniji koncepti iz ovog udžbenika

DOI: 10.7251/ZRSNG1501051M

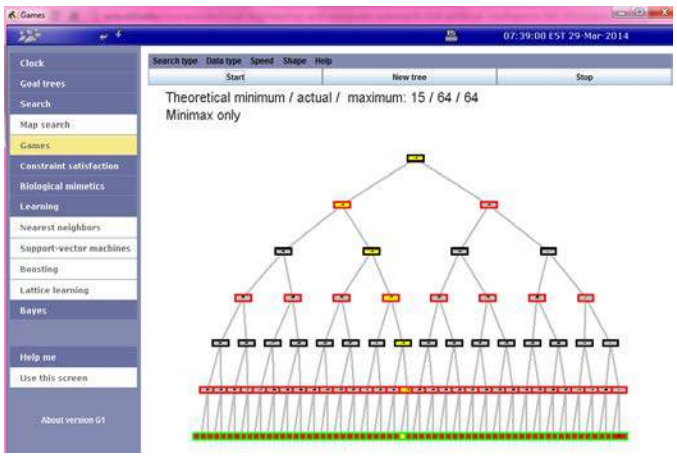
su predstavljeni pomoću simulatora, kako bi se olakšalo njihovo razumevanje. Obrađeni su mnogobrojni algoritmi pretraživanja, a korisnički interfejs je veoma preglednan i omogućava lako upoznavanje i korišćenje. U okviru modula A* nije moguće učitati sopstveni skup podataka, već se koristi samo ugrađeni primer koji predstavlja obilazak gradova u Rumuniji. Iako se ne može koristiti za druge primene, korišćenje samo ovog primera je razumljivo obzirom da prati zadatke iz udžbenika koji upravo njega koriste kao osnovu i sa njim predstavljaju kompletnu kombinaciju koja omogućava relativno lako razumevanje osnovnih koncepata.

Sl. 4. Izgled ekrana simulatora AIMA3e.



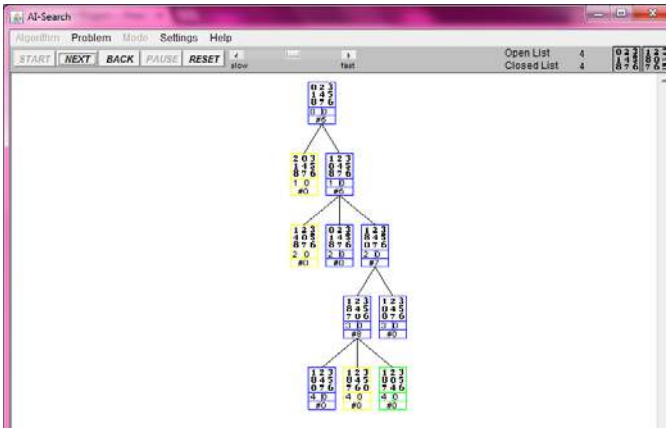
Simulator *Open CourseWare* [10] je simulator koji se koristi na MIT-u za potrebe demonstriranja algoritama pretraživanja (Sl. 5.). Obuhvaćeni su najpoznatiji algoritmi pretrage (po dubini, po širini, planinarenje, pretraga po snopu, grananje sa ograničavanjem, A*), i razne mini-igre za *mini-max* pretragu, *alpha-beta* odsecanje itd. Iako ovaj simulator obuhvata veliki skup mogućnosti, probleme u njegovom korišćenju može stvoriti činjenica da se ne može upravljati algoritmom na precizan način brojem koraka koji odgovaraju korisniku, kao ni vratiti na neki prethodni korak.

Sl. 5. Izgled ekrana simulatora MIT *Open CourseWare*.



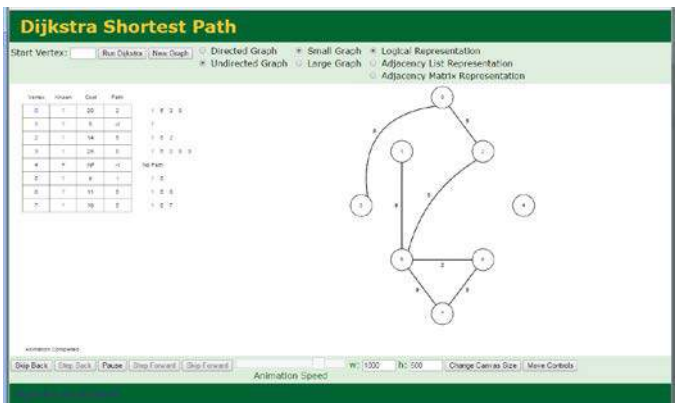
Simulator *AI-Search* (Sl. 6.) [11] razvijen je na RMIT Univerzitetu u Australiji. Na *8-puzzle* primeru se demonstriraju najpoznatiji algoritmi pretrage kao što su pretraga po dubini i širini, A*, pohlepni algoritmi ili iterativno produbljivanje. U simulatoru je moguće prolaziti kroz problem korak po korak. Zanimljivo je što je za demonstraciju pretraživanja upotrebljena igra *8-puzzle*. Iako je sistem u upotrebi oko 15 godina i za današnje pojmove ima prilično zastareo intefejs koji može da bude problematičan za korišćenje i razumevanje, može se veoma uspešno koristiti u nastavi.

Sl. 6. Izgled ekrana simulatora AI-Search.



Data Structure Visualizations (Sl. 7.) predstavlja zanimljivu kolekciju simulatora koja je razvijena na *University of San Francisco* [12]. Obuhvaćeni su mnogobrojni grafovski algoritmi (uključujući i Dijkstrin), algoritmi sortiranja i indeksiranja itd. Bitno je naglasiti da je ceo sistem napravljen pomoću *JavaScript*-a što mu omogućava da radi na velikom broju uređaja bez instaliranja bilo kakvih dodatnih komponenti. Interfejs je pregledno organizovan i ima lako dostupne opcije. Svaki algoritam se u bilo kom trenutku može pauzirati, a program ima jasno prikazan izlaz, tako da korisnici mogu jasno videti u kojoj se fazi algoritam nalazi u svakom trenutku.

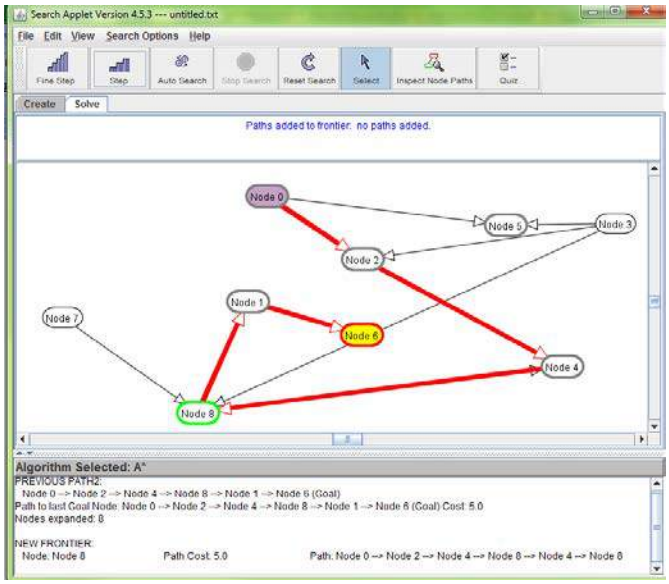
Sl. 7. Izgled ekrana simulatora *Data Structure Visualisations*.



DOI: 10.7251/ZRSNG1501051M

AI Space simulator (Sl. 8.) [13] se koristi za učenje i istraživanje koncepata u oblasti veštačke inteligencije [14]. Razvijen je i koristi se na *University of British Columbia*. Obuhvata veliki broj algoritama i tehnika pretraživanja grafova, obrađuju se i stabla odlučivanja, neuronske i Bajesove mreže itd. I pored velike raznolikosti obuhvaćenih tehnika, uložen je veliki trud da interfejs između svih modula bude ujednačen, kako bi se korisnici što brže prilagođavali. Kroz rešavanje problema se može proći odjednom, ili postepeno korak po korak. Jedini potencijalni nedostatak može predstavljati delimično konfuzan način ispisivanja poruka koje prate rad algoritama, kao i grafički aspekt izgleda simulatora, mada to ne umanjuje opšti utisak upotrebljivosti.

Sl. 8. Izgled ekrana simulatora *AI Space*.



IV. ZAKLJUČAK

Nijedan drugi algoritam koji pretražuje od početnog čvora uz korišćenje iste heurističke funkcije neće proširiti manji broj čvorova od A*. Sa druge strane, održavanje liste neotvaranih čvorova može veoma brzo potrošiti dosta memorijskih resursa. Prostorna složenost je linearna i zavisi od dubine na kojoj se nalazi optimalno rešenje u okviru grafa. Vremenska složenost zavisi od heurističke funkcije koja se koristi.

I pored određenih specifičnih problema koji se mogu javiti, A* predstavlja izuzetno korišćeno rešenje u praksi. Upravo se zbog toga i izučava na velikom broju univerziteta širom sveta, što je doprinelo razvoju velikog broja edukativnih softverskih simulatora u ovoj oblasti. Pored prikaza rada algoritma A*, ovaj rad sadrži i prikaz najpopularnijih vizuelnih simulatora uz navođenje nekih od njihovih najvažniji karakteristika

ZAHVALNICE

Ovaj rad je delimično finansiran od strane Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije (TR32054).

LITERATURA

[1] Hart P. E., Nilsson N. J., Raphael B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics, 1968.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] Nosrati M., Karimi R., Hasanvand H. A., Investigation of the * (Star) Search Algorithms: Characteristics, Methods and Approaches, World Applied Programming, Vol (2), No (4), 2012, str. 251-256.

[3] Cui X., Shi H., A*-based Pathfinding in Modern Computer Games, IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, January 2011, str. 125-130.

[4] Zhang Z., Sturtevant N. R., Holte R., Schaeffer J., Felner A., A* search with inconsistent heuristics, Proceedings of the 21st international joint conference on Artifical intelligence (IJCAI'09), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, str. 634-639.

[5] Nikolić B., Ekspertski sistemi, Elektrotehnički fakultet, Beograd, 2011.

[6] Grant S., Williams D., Using the A-Star Path-Finding Algorithm for Solving General and Constrained Inverse Kinematics Problems, Orion Transfer, 2008.

[7] Martelli A., On the Complexity of Admissible Search Algorithms, Artificial Intelligence, 8(1):1–13, 1977.

[8] http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html, datum pristupa: 3.10.2015.

[9] AIMA3e, https://code.google.com/p/aima-java/, datum pristupa: 8.10.2015.

[10] MIT Open CourseWare, http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/demonstrations/, datum pristupa: 8.10.2015.

[11] AI-Search, http://www.cs.rmit.edu.au/AI-Search/Product/, datum pristupa: 5.10.2015.

[12] Data Structure Visualisations, https://www.cs.usfca.edu/~galles/visualization/, datum pristupa: 8.10.2015.

[13] AISpace, http://www.aispace.org/, datum pristupa: 30.9.2015.

[14] Knoll B., Kisynski J., Carenini G., Conati C., Mackworth A., Poole D. L., AIspace: Interactive Tools for Learning Articial Intelligence, Proceedings of the AAAI AI Education Colloquium, Chicago, IL, 2008.

DOI: 10.7251/ZRSNG1501051M