SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Hana Tomich

RAZVOJ ALGORITMA ZA OPTIMIZACIJU RUTE KROZ LABIRINT

SEMINARSKI RAD

UVOD U UMJETNU INTELIGENCIJU

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Hana Tomich

Matični broj: 0016147359

Studij: Informacijski i poslovni sustavi

RAZVOJ ALGORITMA ZA OPTIMIZACIJU RUTE KROZ LABIRINT

SEMINARSKI RAD

Mentor:

dr. sc. Bogdan Okreša Đurić

Varaždin, siječanj 2024.

Hana Tomich

Izjava o izvornosti

Izjavljujem da je ovaj SEMINARSKI RAD izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Ovaj rad bavi se razvojem i implementacijom algoritma A* za optimizaciju rute kroz labirint, s praktičnom primjenom na igru NetHack. Algoritam A* predstavlja standard u teorijsko-metodološkom pristupu rješavanja problema navigacije, spajajući efikasnost i optimalnost putem heurističkog pretraživanja. U radu se detaljno razmatraju teorijske osnove algoritma, izvedivost i način na koji heuristika utječe na performanse pretraživanja. Središnji dio rada usredotočuje se na programsku implementaciju algoritma A*, koja je prilagođena za interaktivno rješavanje labirinta u posebnoj tekstualnoj datoteci veličine 5x5. Korisnik ima mogućnost unosa početne i završne točke, te prepreka unutar labirinta, nakon čega algoritam izračunava najefikasniju rutu izlaska. Ova prilagodba algoritma omogućuje razumijevanje njegovih osnovnih principa te pruža jasan uvid u njegovu praktičnu primjenost. Rezultati pokazuju da algoritam A* uspješno rješava postavljeni problem, demonstrirajući svoju učinkovitost i prilagodljivost u različitim scenarijima pretraživanja. Rad pridonosi daljnjem razumijevanju i popularizaciji algoritma A* u području razvoja računalnih igara i umjetne inteligencije.

Ključne riječi: NetHack, A*algoritam, pretraživanje, najkraći put, optimizacija, umjetna inteligencija

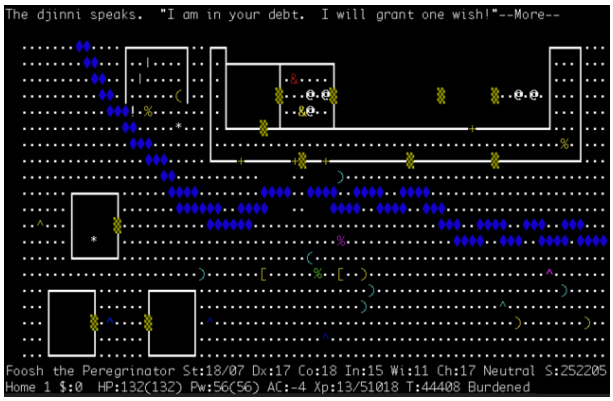
Sadržaj

1. Uvod

Najveći broj igara s primijenjenim metodama proceduralno generiranog sadržaja su pripadale takozvanom RPG (engl.*roleplaying game*) žanru, odnosno preciznije podžanru RPG igara kojeg su igrači neslužbeno nazvali engl.*Roguelike* [1]. NetHack je poznat po svojim nasumično generiranim labirintima, što igračima pruža jedinstven i nepredvidiv izazov svaki put kada igraju. Ovaj rad fokusira se na razvoj i implementaciju algoritama koji bi mogli omogućiti efikasno pronalaženje puta u takvim labirintima, koristeći pristup A* algoritam. Primarni cilj ovog rada je istražiti i razviti algoritme koji bi efikasno rješavali problem pronalaženja puta kroz labirint u igri NetHack. Uz to, cilj je također razumjeti prednosti i ograničenja svakog algoritma te kako se oni mogu prilagoditi specifičnostima i izazovima koje nudi NetHack. Posebna pažnja posvetit će se A* algoritmu zbog njegove široke primjene i efikasnosti, kao i algoritmu pretrage prvo u dubinu, koji nudi drugačiji pristup problemu.

Interes za ovu temu proizašao je iz želje da se istraži kako klasični algoritmi pretraživanja, poput A*, mogu biti primijenjeni u realnim i interaktivnim scenarijima. NetHack, ig ra koja je svojim labirintima i procedurama generiranja sadržaja izazivala i zabavljala generacije igrača, poslužila je kao savršeno polje za primjenu i testiranje algoritma. Tema je privukla pažnju zbog svoje interaktivnosti i mogućnosti za korisnika da izravno sudjeluje u igri, što povećava angažman i pruža neposrednu povratnu informaciju o performansama i efikasnosti algoritma. NetHack nije samo igra, već dio opće kulture, što daje dodatnu vrijednost i motivaciju za detaljno razumijevanje njenih mehanizama i mogućnosti koje pruža za inovacije u području algoritama.

Razvoj efikasnih algoritama za navigaciju kroz labirint ne samo da može unaprijediti igračko iskustvo u igrama poput NetHack-a, već ima i šire primjene u robotici, automobilima koji se sami voze, i drugim područjima gdje je potrebno efikasno navigiranje kroz složene prostore.



Slika 1: Igra NetHack. Izvor: `https://en.wikipedia.org/wiki/NetHack`

2. A* algoritam i NetHack

A* Algoritam: A* algoritam je napredna tehnika pretraživanja za pronalaženje optimalne rute u grafu, kombinirajući stvarne troškove putanja i heurističke procjene [2]–[4].

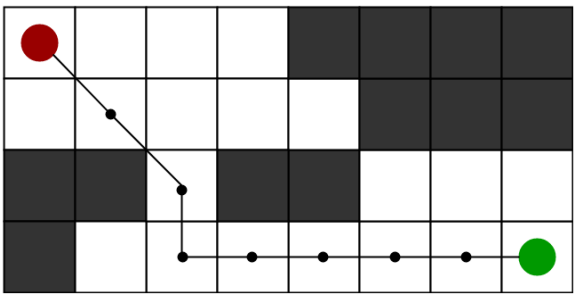
Pretraživanje Prvo u Dubinu: Metoda koja istražuje dublje u grafu prije nego što ispituje alternativne rute [5].

NetHack: NetHack je kompleksna roguelike video igra, poznata po svojoj dubini i generativnoj prirodi [6]

Optimizacija: Proces pronalaženja najefikasnijeg ili najefektivnijeg načina za obavljanje zadatka ili rješavanje problema, ključan u algoritmima kao što je A* [7], [8].

Heuristika: Heuristika u računalnim znanostima je metoda ili pristup rješavanju problema koji pruža dovoljno dobra rješenja u razumnom vremenskom okviru. Ova metoda možda neće pronaći najbolje moguće rješenje, ali je vrijedna zbog svoje efikasnosti i sposobnosti da pruža brze procjene troškova. Heuristike se često koriste u kombinaciji s algoritmima optimizacije kako bi se poboljšala njihovu učinkovitost. U kontekstu algoritama poput A*, heuristike omogućuju brzo i učinkovito pronalaženje puteva u grafovima, smanjujući broj čvorova koje je potrebno obraditi. Heuristike mogu biti admissible, što znači da nikada ne preprocjenjuju stvarni trošak do cilja,Primjeri admissible heuristika uključuju Hamming i Manhattan udaljenosti, koje se koriste u različitim problemima pretrage i optimizacije [9].

U kontekstu igre NetHack, važno je razumjeti kako se ovi algoritmi mogu primijeniti na nasumično generirane labirinte. A* algoritam može biti efikasan u pronalaženju najkraćeg puta do cilja, ali zahtijeva dobru heuristiku koja može procijeniti udaljenost u složenom i promjenjivom okruženju. Razumijevanje ovog algoritama i njegovih karakteristika ključno je za razvoj efikasnih metoda navigacije kroz labirint u igri NetHack. Osim toga, primjena ovog algoritama u složenom okruženju igre pruža vrijedne uvide u njihovu praktičnu primjenjivost i ograničenja, što je od velike važnosti za daljnje istraživanje u području umjetne inteligencije.



Slika 2: A* algoritam pretraživanja. Izvor: `https://www.geeksforgeeks.org/a-search-algorithm/`

3. Kritički osvrt

U ovom poglavlju fokusiramo se na praktičnu izvedivost i primjenu algoritama pretraživanja u kontekstu optimizacije rute kroz labirint, posebno u igri NetHack. Iako teorijska osnova i algoritamska efikasnost mogu biti uvjerljive, praktična primjena često predstavlja značajne izazove. Ovi izazovi uključuju adaptaciju algoritama na dinamička i nepredvidiva okruženja, efikasnost izvršavanja, i relevantnost u stvarnim aplikacijama.

NetHack je poznat po svojim složenim i nasumično generiranim labirintima, što predstavlja izazov za algoritme kao što je A*. U mojoj implementaciji algoritma zbog velike složenosti igrice, korisnik unosi labirint u posebnu tekstualnu datoteku, veličine 5 redova i 5 stupaca.Jedan od ključnih aspekata je adaptacija algoritama na promjenjive uvjete, gdje prethodno izračunati putevi mogu postati zastarjeli ili neefikasni zbog dinamičkih promjena u labirintu. To zahtjeva razvoj algoritama koji su sposobni brzo reagirati na promjene i ažurirati svoje putanje. U kontekstu igara poput NetHack-a, važno je da se algoritmi mogu izvoditi u stvarnom vremenu bez značajnog usporavanja igre. A* algoritam, iako efikasan u teoriji, može zahtjevati znatne resurse prilikom obrade velikih labirinata. Jedan od ključnih izazova je prilagodba ovih algoritama specifičnostima NetHack-a. To može uključivati razvoj prilagođenih heuristika za A* algoritam koje bolje odražavaju stvarne uvjete labirinta.

Izbor heurističke funkcije ključan je za performanse A* algoritma. Neadekvatna heuristika može dovesti do neefikasnog pretraživanja ili čak propusta optimalnog puta. Također, prilagođavanje algoritma za specifične aspekte igre, poput izbjegavanja neprijatelja ili optimizacije resursa, može biti složeno A* algoritam nudi snažan alat za navigaciju i optimizaciju puteva u igricama poput NetHacka, ali njegova primjena zahtijeva pažljivo razmatranje igračkog iskustva, dinamičnosti okruženja i optimizacije performa. Kritički osvrt na praktičnu izvedivost i primjenu algoritama A* i u igri NetHack pokazuje da, iako su teorijski zvučni, njegova stvarna primjena zahtjeva značajnu prilagodbu i optimizaciju. Balansiranje između teorijske efikasnosti i praktične izvedivosti ključno je za uspješnu implementaciju ovog algoritama u dinamičkim i nepredvidivim okruženjima, ne samo u igrama, već i u širem spektru primjena umjetne inteligencije.

4. Opis implementacije

Implementacija algoritma A* za optimizaciju rute kroz labirint uključuje nekoliko ključnih komponenata koje zajedno omogućuju efikasno pronalaženje puta.

Ključna komponenta ove implementacije je klasa Cvor, koja služi kao osnovna strukturalna jedinica u labirintu. Svaki Cvor sadrži informacije o svojoj poziciji unutar labirinta, povezanost s roditeljskim čvorom, te troškove putovanja od početnog do trenutnog čvora (g), procijenjenu udaljenost do cilja (h), i ukupni trošak (f). Ova klasa je također opremljena metodom za usporedbu čvorova, što je ključno za održavanje prioritetnog reda u A* algoritmu.

U A* algoritmu je heuristička funkcija, u ovom slučaju implementirana kao Manhattanska udaljenost. Ova funkcija procjenjuje minimalni broj koraka potreban za kretanje između dvije točke u labirintu, uzimajući u obzir samo horizontalna i vertikalna kretanja. To omogućuje algoritmu da efikasno procijeni potencijalne rute i usmjerava pretragu prema cilju.

Algoritam A* implementiran je kroz funkciju astar, koja koristi gore navedene komponente za navigaciju kroz labirint. Algoritam počinje inicijalizacijom otvorenog i zatvorenog skupa čvorova (otvoreni_skup i zatvoreni_skup). Otvoreni skup, implementiran korištenjem heapq modula za efikasno upravljanje prioritetnim redom, sadrži čvorove koji čekaju na obradu, dok zatvoreni skup sadrži one čvorove koji su već obrađeni. Kroz iterativni proces, algoritam istražuje susjedne čvorove trenutnog čvora, procjenjuje njihove ukupne troškove, i ažurira otvoreni skup. Kada se dođe do ciljnog čvora, put se rekonstruira i vraća kao rezultat.

Jedna od praktičnih komponenti implementacije je funkcija ucitaj_dokument, koja omogućava učitavanje labirinta iz vanjske tekstualne datoteke. Ova funkcionalnost pruža fleksibilnost u definiranju labirinta, omogućujući lako mijenjanje i prilagođavanje labirinta bez potrebe za izmjenom samog koda. Funkcija čita datoteku liniju po liniju, pretvara znakove u odgovarajuće vrijednosti unutar labirinta, i određuje početne i krajnje točke.

5. Prikaz rada aplikacije

Prikaz rada aplikacije za navigaciju kroz labirint temelji se na implementaciji algoritma A* unutar okruženja igre poput NetHack. Ovaj prikaz detaljno objašnjava kako se aplikacija ponaša od trenutka učitavanja labirinta do pronalaska i prikaza optimalne rute.

Kada se aplikacija pokrene, prvi korak je učitavanje labirinta iz tekstualne datoteke koristeći funkciju ucitaj_dokument. Ova funkcija čita strukturu labirinta, liniju po liniju, iz vanjske datoteke i pretvara je u matricu koja predstavlja labirint. Zidovi labirinta označeni su kao 'X', hodnici kao prazni prostori, a početna i krajnja točka kao 'S' (Start) i 'E' (End).

Nakon uspješnog učitavanja, aplikacija inicijalizira algoritam A* postavljanjem početnih i krajnjih točaka, te stvaranjem početnog i krajnjeg čvora. Otvoreni i zatvoreni skupovi (otvoreni_skup i zatvoreni_skup) inicijalizirani su za daljnju obradu.

Algoritam A* započinje svoje izvršavanje. U svakoj iteraciji, algoritam iz otvorenog skupa (koji je prioritetni red) izvlači čvor s najmanjim ukupnim troškom (f vrijednost) i analizira njegove susjedne čvorove. Za svakog susjeda, izračunavaju se g, h i f vrijednosti. Ako je susjedni čvor bolji (tj. nije u zatvorenom skupu i predstavlja potencijalno bolji put), dodaje se u otvoreni skup.

Kada algoritam dosegne krajnji čvor, put se rekonstruira praćenjem roditeljskih veza od krajnjeg do početnog čvora. Ovaj put potom se prilagođava za prikaz, pretvarajući koordinate iz 0-baziranih u 1-bazirane, i prikazuje se korisniku.

Korisnik može promatrati proces pronalaska puta kroz labirint, što pruža uvid u složenost i efikasnost algoritma A*. U slučaju da put nije pronađen ili ako labirint nije ispravan, aplikacija informira korisnika o problemu.

6. Zaključak

U svijetu računalnih igara, složenost virtualnih okruženja neprestano raste, postavljajući nove izazove u području umjetne inteligencije. Jedan od klasičnih problema u AI-a je navigacija kroz labirint, što je posebno važno u igrama poput NetHack-a [10]. Ovaj seminarski rad pružio je detaljan uvid u razvoj algoritma za optimizaciju rute kroz labirint, s posebnim fokusom na igru NetHack i primjenu A* algoritma. Rad je obuhvatio sve od teorijskih osnova, preko kritičkog osvrta na praktičnu izvedivost, do detaljnog opisa implementacije i prikaza rada aplikacije.

Implementacija algoritma A* pokazuje kako se složeni algoritmi mogu primijeniti u stvaranju dinamičkih i interaktivnih igračkih okruženja. Kroz kombinaciju strukturiranih klasa, efikasnih algoritamskih tehnika i fleksibilnog učitavanja podataka, stvara se sustav koji ne samo da efikasno rješava problem navigacije kroz labirinte, već i pruža temelj za daljnji razvoj i unapređenje igračkih mehanika.

Literatura

[1] M. Panić, *Algoritmi za proceduralno generiranje sadržaja u računalnim igrama*, Dostupno na: https://zir.nsk.hr/islandora/object/algebra:286, 2015.

[2] Wikipedia. „A* search algorithm.” Dostupno na: https://en.wikipedia.org/wiki/A*_search_algorithm. (2022.), adresa: https://en.wikipedia.org/wiki/A*_search_algorithm.

[3] Brilliant. „A* Search.” Dostupno na: https://brilliant.org/wiki/a-star-search/. (2022.), adresa: https://brilliant.org/wiki/a-star-search/.

[4] Codecademy. „A* Search Algorithm.” Dostupno na: https://www.codecademy.com. (2022.), adresa: https://www.codecademy.com.

[5] S. University. „Introduction to A*.” Dostupno na: https://theory.stanford.edu. (2022.), adresa: https://theory.stanford.edu.

[6] Wikipedia. „NetHack.” Dostupno na: https://en.wikipedia.org/wiki/NetHack. (2022.), adresa: https://en.wikipedia.org/wiki/NetHack.

[7] AlmaBetter. „A* Algorithm in AI (Artificial Intelligence).” Dostupno na: https://www.almabetter.com. (2022.), adresa: https://www.almabetter.com.

[8] HandWiki. „A* search algorithm.” Dostupno na: https://handwiki.org. (2022.), adresa: https://handwiki.org.

[9] Wikipedia, *Heuristic (computer science)*, Dostupno na: https://en.wikipedia.org/wiki/Heuristic_(computer_science), 2022.

[10] A. Patel, *A* Search Algorithm*, Dostupno na: https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html, 2020.

Korišteni alati:

- **Visual Studio Code**
- **Chat GPT** (https://chat.openai.com)
- **Jupyter Notebook**