# Vizualizacija širjenja informacije po omrežju

## Graph Algorithm Visualizer
## Tehnična dokumentacija

Matic Adamič 89202071
Vid Jerovšek 89202051

Koda je dostopna na githubu.

June 2021

# 1 Uvod

## 1.1 Porazdeljeni algoritmi na grafih

Algoritme na grafih lahko razdelimo v dve večji skupini. Prva skupina, klasični algoritmi na grafih, so skupina algoritmov, ki jih dobro poznamo. Taki algoritmi ponavadi iščejo neko lastnost grafa ali najkrajšo pot med dvemi vozlišči, izmed drugimi. Najbolj znani algoritmi so recimo: iskanje v širino, iskanje v globino, iskanje vpetih dreves, itd. Lastnost teh algoritmov je, da kot vhod prejmejo celoten graf, katerega obdelujejo.

Druga skupina, katera je fokus te projektne naloge, se imenuje porazdeljeni algoritmi na grafih , kjer je cilj algoritma ponavadi barvanje grafa, izmed drugih. V kontrastu s klasičnimi algoritmi, ki kot vhod dobijo celoten graf, porazdeljeni algoritmi kot vhod prejmejo eno vozlišče, na katerem se algoritem izvaja. Poznamo dva osnovna modela izvajanja algoritma: sinhrono in asinhrono. Pri asinhronem načinu izvajanja, se algoritem izvaja neodvisno od izvajanja na drugih vozliščih grafa, in komunikacija med vozlišči je asinhrona. Pri sinhronem načinu izvajanja, se algoritem izvaja po korakih: vsako vozlišče izvede algoritem, kjer lahko komunicira s sosedi. Ko se izvajanje algoritma konča v vsakem vozlišču, se korak zaključi in začne se nova korak, kjer se stvar ponovi.

V tej nalogi je izvajanje porazdeljenega algoritma implementirana v sinhronem načinu.

## 1.2 Definicija problema

Ideja programa je, da uporabniku omogoča poganjanje poljubnega algoritma, ki ga napiše uporabnik v svojem izbranem IDE-ju. Uporabnik bo kot zunanjo knjižnico klical program, ki bo kot argument prejel uporabnikov algoritem, nato pa bo odprl okno za vizualizacijo grafa in algoritma.

Program bo imel več funkcionalnosti, kot so izris grafa, uvoz grafa, sprehajanje po zgodovina izvajanja algoritma in nekaj nastavitev za izris grafa. Pri uvozu grafa, uporabnik lahko izbere med:
1) Prazen graf.
2) Uvoz grafa iz datoteke (kjer je format grafa sparse6 ali graph6), uporabnik poda število začetnih informiranih vozlišč.
3) Uvoz naključnega algoritma, kjer uporabnik poda število vozlišč, verjetnost povezave med dvema poljubnima vozliščema in število začetnih informiranih vozlišč.
4) Uvoz klike, kjer uporabnik poda število vozlišč in število začetnih informiranih vozlišč.

Pri uvozu grafov si uporabnik lahko specificira začetno postavitev grafa, izbira lahko med:
1) Krožna postavitev
2) Naključna postavitev
3) Postavitev FR (Fruchterman and Reingold Force-Directed Layout)
4) Biparticijska postavitev (če je možna, ni vsak graf biparticijski)

Program bo beležil stanje vseh vozlišč ob v določenem trenutku izvajanja algoritma. Uporabnik lahko tako po koncu izvajanja algoritma obišče katero koli stanje grafa. Stanja grafa se med seboj razlikujejo po rundi algoritma. Vsaka nova runda algoritma predstavlja novo stanje grafa, ki se zabeleži v zgodovino. Med sprehajanjem po zgodovini stanj, se stanje vseh vozlišč interaktivno spreminja. Uporabnik ima možnost spreminjanje stanj poljubnega vozlišča, vendar se v tem primeru prihodnja zgodovina izbriše. Poleg spreminjanje stanj vozlišč, uporabnik lahko v graf dodaja ali briše dodatna vozlišča, v katerem primeru se prav tako izbriše prihodnja zgodovina.

# 2 Aplikacija

## 2.1 Načrtovanje

Program je v osnovi sestavljen iz dveh okenj:
1) Glavno okno, ki prikazuje simulacijo algoritma na grafu.
2) Okno za vnos novega grafa.

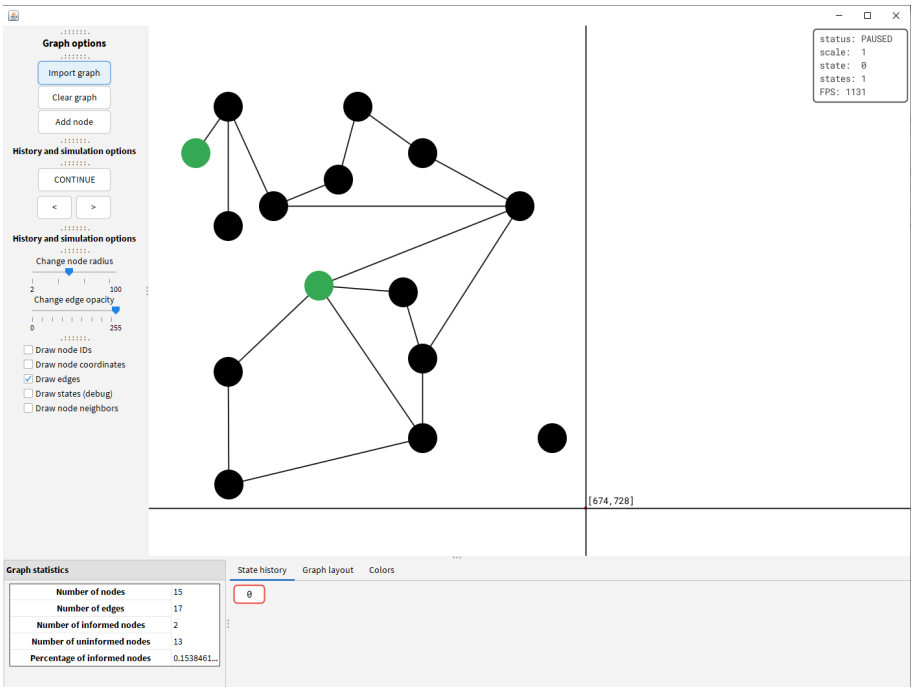


Figure 1: Zaslonska slika glavnega okna z uvoženim grafom.

## 2.2 Glavno okno

Glavno okno je razdeljeno na 4 komponente:
1) Panela, kjer je izrisan graf.
2) Meni na levi strani okna, ki ponuja funkcionalnosti za uvoz grafa, dodajanje vozlišč, spreminjanje velikosti vozlišč in transparentnost povezav.
3) Spodnja panela, ki vsebuje zavihka za spremljanje zgodovine, spreminjanje barv grafa: barvo informiranih in neinformiranih vozlišč, barvo povezav, in zavihek za spreminjanje postavitev grafa, kjer imamo enake možnosti kot pri uvozu grafa.
4) Tabela v spodnjem levem kotu, kjer se beleži osnovna statistika grafa: število vozlišč in povezav, število informiranih in neinformiranih vozlišč.

## 2.3 Okno za uvoz grafa

Okno za uvoz grafa se odpre, ko uporabnik pritisne na gumb "Import graph" gumb v menuju glavnega okna. V tem oknu uporabnik izbira med možnostmi, ki so opisane v definiciji problema; torej lahko bere iz datoteke ali pa ustvari poljubno kliko ali naključen graf. Izbere lahko tudi začetno postavitev. S pritiskom na gumb "Import" se graf uvozi, kjer ga uporabnik lahko vidi v glavni paneli. Če je pred uvozom že obstajal graf, potem se stari graf izbriše, prav

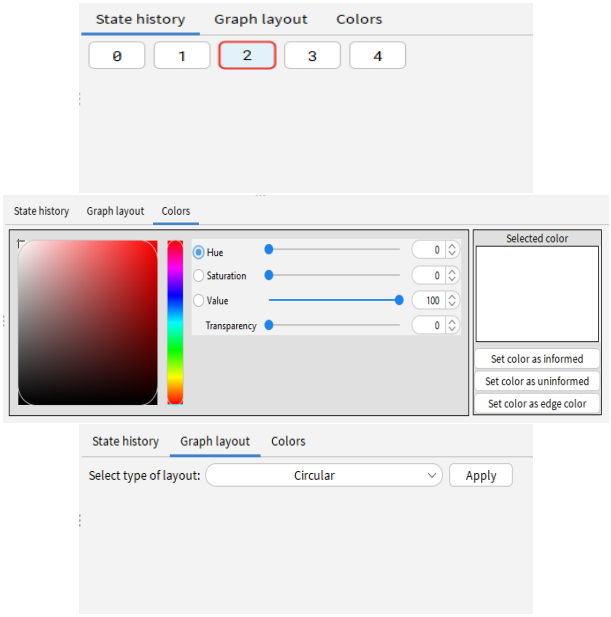


Figure 2: Zaslonske slike zavihkov spodnje panele: zavihek zgodovine stanj grafa, kjer je trenutno izbrano stanje 2, zavihek za izbiranje barv grafa, zavihek za spreminjanje postavitve grafa.

tako se izbriše prejšnja zgodovina, če je obstajala.

## 2.4 Izvajanje algoritma

Drugi ključni del programa je izvajanje algoritma na grafu. Simulacijo algoritma bo več nitna, za bolj učinkovito in hitrejše izvajanje algortma na večjih grafih. Kontrolna nit mora imeti pod seboj več delovnih niti - niti, ki bodo izvajale algoritem. Vsaki delovni niti se pripiše delež vozlišč grafa. Kontroler poskrbi za vzporedno izvajanje niti in sinhronizacijo med njimi.

## 2.5 Implementacija

Program je implementiran v programskem jeziku Java. Uporabniški vmesnik je narejen s pomočjo knjižnice za ustvarjanje vmesnikov Swing, ki je del JDK razvojnega paketa.
Za beleženje strukture je uporabljena knjižnica JGraphT, ki ponuja branje grafov iz datotek in različne postavitve grafov. Kot nabor uporabnik funkcij glede nizov in podatkovnih struktur je uporabljena knjižnica Apache Commons. Za izgled uporabniškega vmesnika je uporabljena knjižnica FlatLaf, ki nastavi "Look and Feel" uporabniškemu vmesniku, ki nastavi izgled gumbom, barvam in ostalim komponentam uporabniškega vmesnika.

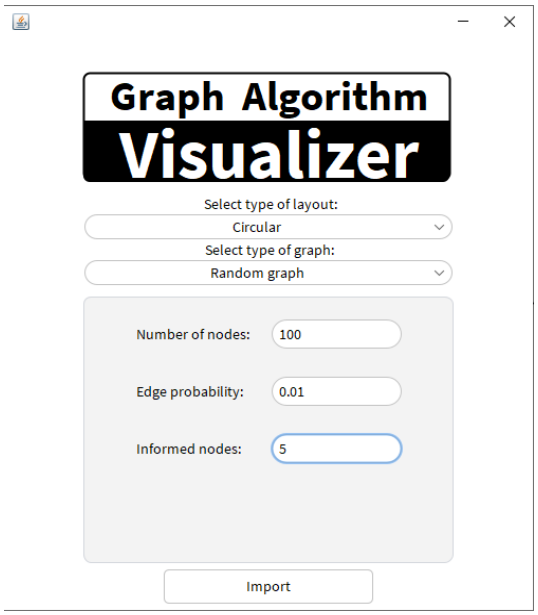


Figure 3: Zaslonska slika okna za uvožanje grafa. Izbrana je možnost naključnega grafa s podanimi parametri: število vozlišč: 100, verjetnost povezave: 0.1, število začetno informiranih vozlišč: 5, začetna postavitev je nastavljena na krožno.

Sam program je implementiran z idejo, da bo uporabljen kot eksterna knjižnica v uporabniškem programu. Uporabnik bo uvozil naš program kot knjižnico in enostavno ustvari glavni razred, ki bo inicializiral naš program in mu podal svoj algoritem. Uporabnik bo na ta način lahko implementiral algoritem v svojem razvojem okolju. Algoritem je definiran kot vmesnik, s samo eno metodo, ki prejme kot parameter vozlišče. Na ta način lahko uporabnik hitro implementira poljuben program kot lambdo, ali pa ustvari bolj kompleksen algoritem v svojem razredu, ki implementira vmesnik "Algorithm".

### 2.5.1 Struktura programa

Program je razdeljen na dve komponenti, kot je opisano v načrtovanju. Vsaka komponenta vsebuje svoje podkomponente. Podkomponente glavnega okna so gumbi, zavihki in meni. Podkomponente okna za uvoz grafa sta menija za izbiro načina uvoza grafa in izbira začetne postavitve grafa, in panela, ki se interaktivno spreminja glede na izrani tip uvoza grafa (datoteka, naključen graf, ...) in vsebuje nastavitve za število vozlišč, število informiranih vozlišč, itd. Vse glavne komponente so implementirane v svojem razredu, podkomponente pa so polja, ki jih vsebuje razred.

### 2.6 Levi meni

Levi meni vsebuje tri skupine kontrolnih gumbov za:
1) Gumba za uvoz in izbris grafa, ter gumb za ustvarjanje novega vozlišča.
2) Izvajanje algoritma, kot je neprekinjeno izvajanje algoritma, gumb za inkrementalno izvajanje algoritma, in gum za premik nazaj v zgodovini.
3) Drsniki za nastavljanje svetlosti povezav (uporabno, ko je uvožen velik graf z veliko povezavami) in potrditveni polji, ki omogočata izris identifikacijskih številk vozlišč in izris med vozlišči.

### 2.7 Spodnja panela

Spodnja panela je sestavljena iz dveh komponent: tabela na levi strani in panela z zavihki na desni strani. Tabela na levi strani prikazuje statistiko trenutnega uvoženega grafa in zajema: število vozlišč, število povezav, število informiranih in neinformiranih vozlišč.
Panela z zavihki vsebuje tri zavihke:
1) Zgodovino stanj, ki vsebuje gumb za vsako stanje grafa, ki ga je generiral algoritem. Uporabnik se lahko sprehaja skozi zgodovino s pritiskom željenega gumba, ki interaktivno posodobi stanje informiranih vozlišč v grafu.
2) Postavitev grafa, ki ponuja meni, v katerem izberemo novo postavitev grafa in imamo iste možnosti, kot so na voljo v oknu za uvoz grafa.
3) Barvanje grafa, kjer uporabnik lahko interaktivno izbere barvo in jo proglasi za izbrano barvo: informiranih ali neinformiranih vozlišč, ali povezav.

## 3 Simulacija algoritma

Izvajanje algoritma je implementirano ločeno od grafičnega vmesnika in se izvaja v svoji niti. Ob uvozu novega grafa, program inicializira delovne niti, ki izvajajo uporabnikov algoritem. Vsaka nit je zadolžena za izvajanje algoritma na nekem številu vozlišč, to število je približno enako med vsemi nitmi. Ob dodajanju novega vozlišča, se vozlišče poda naključni delovni niti.

Izvajanje algoritma nadzira kontroler, ki poskrbi za pravilno izvajanje delovnih niti. Ob vsaki rundi izvajanja algoritma, kontroler poskrbi za sinhronizacijo delovnih niti: najprej zbudi vse niti, in počaka, da se izvajanje vseh niti konča. Ko vse delovne niti končajo, potem kontroler zaključi rundo. V vsaki rundi vsaka delovna nit poskrbi za izvajanje algoritem nad vozlišči, za katere je nit zadolžena. Na koncu vsake runde se ustvari novo stanje zgodovine in posodobi se statistika. VKontroler v ozadju uporablja ExecutorService in CyclicBarrier, ki skrbita za pravilno sinhronizirano izvajanje niti po rundah.

# 4 Zaključek

Ob ustvarjanju tega programa sva se oba veliko naučila. Sam program je dosegel cilje, ki sva si jih na začetku zadala. Uporabniški vmesnik bi se še vedno lahko izboljšal, kjer bi se dodalo več funkcionalnosti in odpravilo nekaj hroščev. Program je omejen, saj lahko barvno vizualizira le širjenje informacije v grafu. Program bi se v prihodnosti lahko razširil tako, da bi vizualizacija grafa bila bolj splošna. Ob enem, bi lahko določitev barv posameznega vozlišča prepustila uporabniku, kar bi zelo povečalo uporabnost, zmožnost in fleksibilnost programa, hkrati pa bi lahko testiral širši nabor algoritmov, recimo algoritme za barvanje grafa.
Druga večja nadgradnja bi bila širše beleženje statistike o grafu in o samem izvajanju algoritma. Te podatki bi prišli uporabniku prav, saj veliko porazdeljenih programov uporablja komponente naključnosti.