Vladimir Novosel

Ispitivanje izomorfnosti grafova

Ispitivanje izomorfnosti grafova ima veliki praktični značaj. U radu je predložen algoritam sa osnovnom idejom da se ispitivanje izomorfnosti ne vrši za originalne grafove, već za grafove koji su izvedeni iz njih i za koje je provera izomorfnosti jednostavnija. Utvrđeni su opšti kriterijumi koje bi jedna implementacija algoritma trebalo da zadovolji. Među nekoliko razmatranih načina implementacije, jedan je odabran za implementaciju algoritma koja je predstavljena.

***Ključne reči**. teorija grafova, izomorfnost grafova, algoritam, karakteristično stablo grafa*

Uvod

Lakoća opisivanja fizičkih sistema grafovima, efikasnost metoda koje rešavaju veliku klasu problema, kao i pojava računara koja je omogućila široku primenu ovih metoda za rešavanje realnih problema, učinili su da teorija grafova stekne veliku popularnost i svoju primenu nađe u brojnim naučnim disciplinama, čija se lista stalno proširuje. Mada postoji veliki broj metoda i algoritama koji se efikasno mogu iskoristiti za rešavanje specifičnih problema, očit je nedostatak efikasnih metoda koje se bave fundamentalnim pojmovima teorije grafova.

Ispitivanje izomorfnosti grafova je jedan od takvih problema. Pri traganju za najboljim rešenjem, prirodno se nameću dva osnovna kriterijuma: efikasnost rešenja i njegova opštost (u smislu primenljivosti na različite tipove grafova). Postojeći algoritmi se mogu podeliti u dve grupe. U jednu grupu spadaju rešenja koja žrtvuju efikasnost algoritma zarad njegove opštosti, a u drugu rešenja koja sebi kao osnovni kriterijum postavljaju upravo efikasnost, specijalizujući sa samo za određene klase grafova.

Detaljno izložen formalizam teorije grafova, kao i poduži spisak publikacija vezanih za ovu temu može se naći u knjizi *Teorija grafova i njene primene* (Cvetković 1986).

Vladimir Novosel (1978), Smederevo, Radoja Domanovića 56, student 1. godine Elektrotehničkog fakulteta u Beogradu

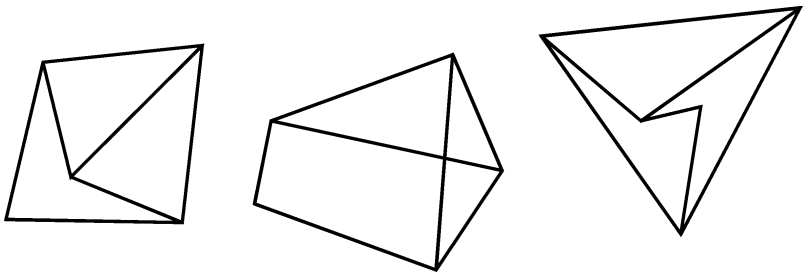
Algoritam

Pojam izomorfnosti

Izomorfizam grafova se definiše kao relacija koja preslikava skup čvorova i grana jednog grafa u skup čvorova i grana drugog, tako da održava osobinu susednosti čvorova (Cvetković 1997). To znači da, ako su dva čvora jednog grafa povezana granom, onda su granom povezani i odgovarajući čvorovi drugog grafa. Ako zamislimo čvorove i grane jednog grafa u prostoru, pomeranjem čvorova i deformisanjem odgovarajućih grana, možemo ga dovesti do poklapanja sa njegovim izomorfnim oblikom. Važno je da pri ovoj transformaciji ne narušimo susednost čvorova.

Težina provere izomorfnosti grafova je u tome što jedan graf može imati veoma veliki broj međusobno izomorfnih oblika. Kako se u primenama grafovi često sastoje od velikog broja čvorova i grana, problem postaje još teži. Priroda složenosti problema ilustrovana je slikom 1, sa tri međusobno izomorfna grafa. Čak ni u ovom jednostavnom slučaju, nije potpuno očigledno da se radi o istom grafu.

U cilju objašnjenja osnovne ideje kojom se vodi algoritam, podsetimo

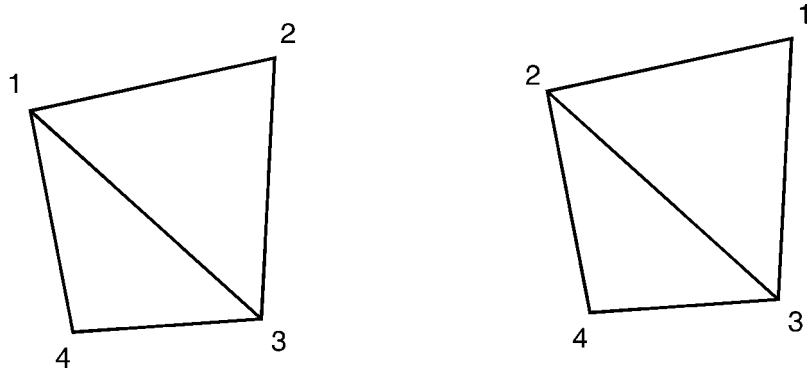


Slika 1.
Međusobno izomorfni grafovi.

Figure 1.
Isomorphic graphs.

se jedne klase grafova, označenih grafova, koju karakteriše specifična, znatno uprošćena provera izomorfnosti. Dva označena grafa se smatraju izomorfnim samo ako se oznake svakog čvora i grane jednog, tačno poklapaju sa oznakama drugog grafa (Cvetković 1997). Sa tako definisanom relacijom, dva grafa prikazana na slici 2 bi bila neizomorfna, iako se očigledno radi o grafovima identične morfologije.

Istina, ovako definisana relacija izomorfnosti izuzetno pojednostavljuje rad sa grafovima, ali istovremeno, klasa problema koji se mogu tretirati ovom vrstom grafova je znatno umanjena, kao i njena praktična primenljivost. Najbolje bi bilo kada bi smo mogli ispitivati izomorfnost grafova sa onom lakoćom koja se sreće kod označenih grafova, a da u isto vreme ne budemo ograničeni samo na tu klasu grafova.



Slika 2.
Neizomorfni označeni grafovi.

Figure 2.
Nonisomorphic signed graphs.

Osnovni koncept našeg algoritma je sledeći: umesto da se izomorfizam proverava za date grafove, upoređuju se morfološki jednostavniji grafovi koji su generisani nad njima. Takvi grafovi moraju biti morfološki karakteristični za graf nad kojim su generisani. Preostaje da se odrede pravila za generisanje ovih grafova, tako da oni budu minimalni grafovi sa željenom osobinom. Time se izbegava negativan efekat na efikasnost algoritma.

Karakteristično stablo grafa

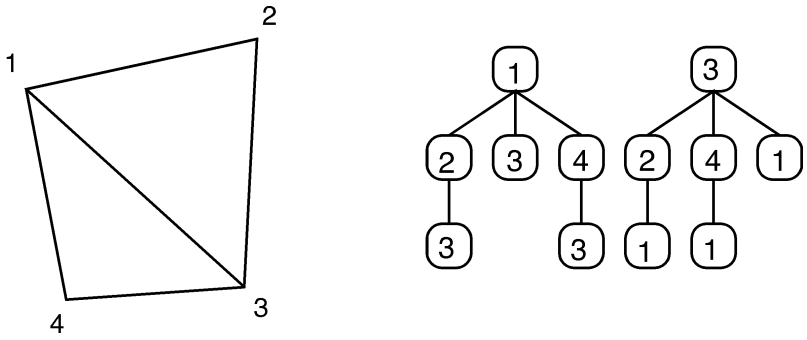
Uvedimo pojam *karakterističnog stabla grafa*. On se odnosi na stabla generisana nad grafovima za koje se utvrđuje izomorfnost, tako da predstavljaju njihovu morfološku invarijantu. Zatim se proverava njihova izomorfnost, kako bi se utvrdila izomorfnost polaznih grafova. Zahvaljujući specifičnoj morfologiji provera izomorfnosti stabala je jednostavna, a prolaz kroz grafove ove vrste se može vršiti po nivoima, što je naročito zgodno za naše potrebe.

Ograničimo se, u početku, samo na orijentisane povezane grafove. Generisanje karakterističnog stabla se vrši prema sledećim pravilima:

1. nađi čvorove sa najvećim izlaznim stepenom, i sve grane koje polaze od tih čvorova ubaci u stablo
2. za svaki krajnji čvor grana prethodnog nivoa, ubaci u stablo sve grane grafa koje polaze iz tih čvorova, a nisu već ubačene u stablo
3. ponavljaj korak 2. dok god ima grana za ubacivanje

Primer generisanja karakterističnog stabla za jedan jednostavan graf prikazan je na slici 3.

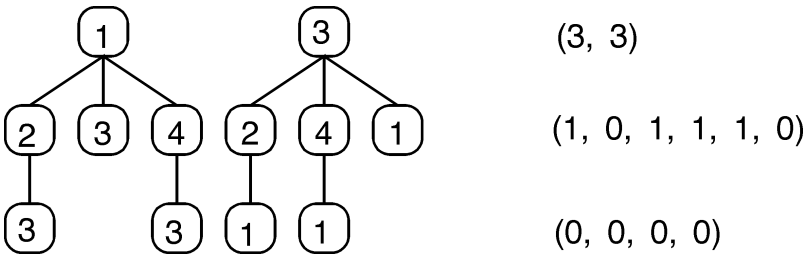
Kada imamo karakteristična stabla grafova, potrebno je proveriti njihovu izomorfnost. Provera se vrši u koracima, prolazeći kroz stablo od korena ka listovima (top-down) jedan po jedan nivo, upoređujući izlazne stepene krajnjih čvorova grana koje se nalaze na trenutnom nivou zane-



Slika 3.
Generisanje karakterističnog stabla grafa.

Figure 3.
Generation of charcteristic graph tree.

marujući njihov raspored. Pošto upoređivanje vršimo po nivoima, proces ispitivanja izomorfnosti možemo prekinuti čim naiđemo na neslaganje, jer to znači da grafovi nisu izomorfni. Pošto se u procesu provere koriste samo izlazni stepeni čvorova stabla, tu informaciju možemo predstaviti skupovima celih brojeva koji predstavljaju izlazne stepene ovih čvorova. Upoređivanje stabala sada vršimo poređenjem odgovarajućih skupova. Slika 4 na primeru demonstrira kako se na osnovu karakterističnih stabala formiraju ovi skupovi.



Slika 4.
Skupovna notacoja karakterističnog stabla.

Figure 4.
Numerical notation of the characteristic graph tree.

Ograničenje da graf koji se prosleđuje algoritmu mora biti povezan, lako se može prevazići tako što bi smo našli komplement tog grafa i nad njim generisali karakteristično stablo grafa. Ovde koristimo osobinu grafova da su povezani ako su nastali komplementiranjem nepovezanih grafova i obrnuto (Cvetković 1997). Drugo ograničenje, da grane moraju biti neorijentisane, predstavlja teži problem. Pošto algoritam tretira neorijentisane grane kao dvostruke suprotno orijentisane grane, to nije, samo po sebi, naročito velik problem, već činjenica da se kod grafova sa orijentisanim granama može javiti više vrsta povezanosti. Ispravno funkcionisanje algoritma se može garantovati samo za jako povezane grafove, dok bi korektnost rezultata u proveri jednostrano i slabo povezanih grafova zavisila od konkretnog slučaja (ibid.).

Karakteristično stablo grafa predstavlja neku vrstu ličnog potpisa morfologije jednog grafa. Ono implicitno sadrži informacije koje se mogu iskoristiti za rešavanje drugih problema, kao što su nalaženje najkraćeg puta, određivanje dijametra grafa itd. U planu je detaljno ispitivanje ovih mogućnosti, kako bi napor utrošen na generisanje karakterističnog stabla grafa bio višestruko isplativ.

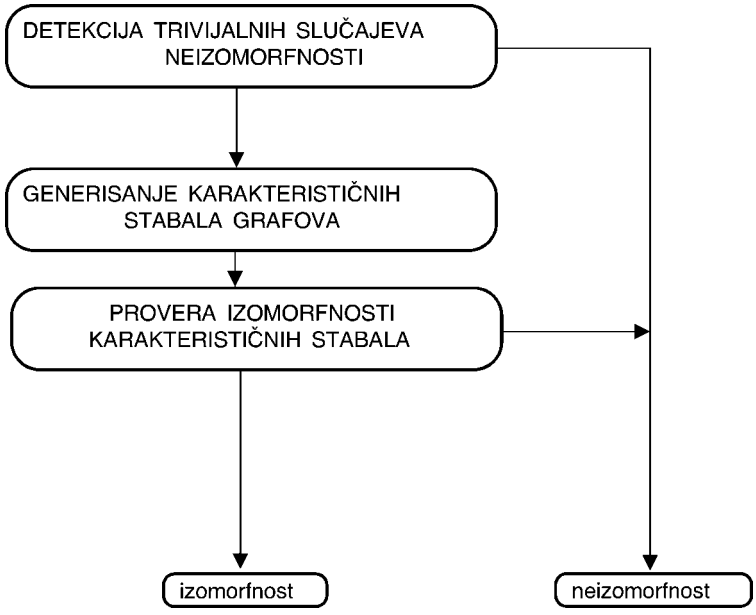
Detekcija trivijalnih slučajeva neizomorfnosti

Pre nego što se generiše karakteristično stablo grafa, vrši se odbacivanje trivijalnih slučajeva neizomorfnosti. Postupak je sledeći:

1. uporedi grafove po broju čvorova i grana
2. uporedi grafove i njihove komplemente po broju komponenata povezanosti
3. uporedi grafove i njihove komplemente po stepenima čvorova
4. ponavljati korake 2. i 3. rekurzivno za svaku komponentu povezanosti dok god one ili njihovi komplementi sadrže više od jedne komponente

U slučaju da se utvrdi neizomorfnost, takvi grafovi se ne šalju glavnom algoritmu. Ako grafovi prođu ovu fazu provere, to ne znači da su oni izomorfni, već samo da nisu trivijalno neizomorfni i da se moraju proslediti osnovnom algoritmu koji će doneti konačnu odluku.

Osnovna šema algoritma data je na slici 5.



Slika 5.
Šematski prikaz algoritma.

Figure 5.
The algorythm.

Implementacija

Veliki problem na koji se nailazi pri implementaciji grafova i algoritama koji rade sa njima, je dizajniranje strukture podataka koja bi bila dovoljno intuitivna za rad, kao i dovoljno efikasna. Implementacija koja koristi matričnu notaciju grafova ima neke svoje prednosti, naročito ako aplikacija ima za zadatak da rešava probleme nalaženja najkraćeg puta između dva čvora (Dijkstra algoritam) ili povezanost (Warshall-ov algoritam), koji su podrobno opisani u (Wiitala 1987) gde se može naći još dosta informacija o matričnoj notaciji grafova i njenoj primeni. Ovakva notacija ima veliki nedostatak – statičnost. Veoma često je potrebno grafovima opisivati kompleksne sisteme ili dinamičke šeme za koje je matrična notacija veoma neprimerena.

Pošto graf nije samo prost skup čvorova i grana, on zahteva visok integritet svojih podataka, u skladu sa određenim ograničenjima koja slede iz definicije grafa i nekih specijalnih osobina kao što su povezanost, orijentisanost grana itd. Zato se čini gotovo prirodnim implementirati grafove u nekom minimalnom relacionom modelu. Ovakva implementacija bi omogućila upotrebu formalno matematičke (skupovne) notacije koja se koristi u teoriji grafova. Ona bi bila sasvim opšta i lako nadogradiva. Takvu implementaciju bi onda mogla koristiti i osoba koja nema programerskog iskustva, ali u dovoljnoj meri poznaje formalizam teorije grafova.

Postoji i opcija da se algoritam implementira koristeći se nekom dinamičkom strukturom podataka kao što je lista ili stablo. Ovaj poslednji pristup je primenjen u našoj implementaciji algoritma, gde imamo listu čvorova koji poseduju informacije samo o granama koje izlaze iz njih.

Na taj način je postignuto da su podaci o grafu lokalnog karaktera, što znači da se oni mogu brzo i lako menjati, bez potrebe da se menja ostatak grafa. Istovremeno, ovakvom notacijom moguće je predstaviti bilo koji tip grafa, prolaz kroz graf je elegantan a implementacija prirodna i laka. Našoj implementaciji smo postavili sledeće ciljeve:

1. održavanje integriteta i validnosti podataka (orijentisanost grana, povezanost)
2. opštost klase i mogućnost njene široke upotrebe
3. implementacija algoritma za ispitivanje izomorfnosti

Kao rezultat, razvijena je klasa GRAPH u programskom jeziku C++ koja zadovoljava ove uslove. Ona pruža mogućnost preciznog definisanja grafa sa kojim želimo da radimo. Bez obzira da li se radi o neorijentisanim, orijentisanim, povezanim ili nepovezanim grafovima i multigrafovima, klasa se brine o tome da intervencije korisnika budu u skladu sa tipom grafa koji je definisan. Sve akcije koje bi narušile strukturu grafa

implementacija ignoriše. Na taj način se sprečava da korisnik greškom ili iz neznanja poremeti integritet grafa.

Korisnik ima na raspolaganju osnovne metode za rad sa grafom: ubacivanje/izbacivanje čvorova i grana; komplementiranje grafa; promena prenosa grane; metode za upit o broju čvorova, grana, komponenti, stepenu nekog čvora. Mogu se dobiti detaljnije informacije o komponentama i prenosima grana. Naravno, tu je i metoda koja vrši proveru izomorfnosti i sadrži implementaciju algoritma. Prenos grane je podatak koji se može pridružiti grani i služi za lakše opisivanje konkretnog fizičkog sistema. Ako bi smo grafove koristili da predstavimo neko električno kolo, prenos grane bi mogao biti podatak o otporu, naponu ili jačini struje odgovarajuće grane kola.

Klasa GRAPH je razvijena kako bi se ispitao algoritam i proverilo uklapanje postavljenih kriterijuma u koncepte objektnog programiranja. Lako je nadogradiva jer dodavanje novih metoda u klasu nije nikakav problem i zavisi isključivo od potreba konkretne primene. Zavisno od toga kako će se upotrebljavati algoritam, može se javiti potreba za drugačijom implementacijom koja će više odgovarati konkretnoj situaciji. U tom slučaju se treba pridržavati istih kriterijuma, budući da su oni sasvim opšte prirode.

Zaključak

U radu se opisuje jedan pristup problemu izomorfnosti kao alternativa metodi 'grube sile'. Podela procesa ispitivanja na dve faze se pokazala kao prilično efikasna. Daljom podelom procesa provere na korake koji su uređeni prema težini, postiže se da grafovi kod kojih je lakše utvrditi izomorfnost ispadaju ranije iz procesa provere, dok grafovi kod kojih su razlike u morfologiji suptilnije izlaze is procesa provere kasnije. Algoritam je očigledno najneefikasniji kada su grafovi zaista izomorfni, jer mora proći kroz ceo proces provere kako bi bio potpuno siguran da se radi o izomorfnim grafovima.

Realizovanom implementacijom su postavljeni kriterijumi koje bi trebalo i svaka druga implementacija da zadovolji. Od nekoliko predloženih načina implementacije, jedan je i realizovan. Odabrana implementacija se pokazala kao veoma prirodna tokom razvoja zbog sličnosti sa stvarnom morfologijom grafova. Metode ugrađene u klasu omogućavaju osnovni nivo interakcije sa podacima klase uz zaštitu integriteta podataka.

Dalji razvoj podrazumeva ispitivanje osobenosti karakterističnog stabla grafa i njegove upotrebe pri rešavanju drugih problema. Upotreba klase GRAPH za rešavanje nekog konkretnog problema bi omogućila da se stečeno iskustvo iskoristiti za uviđanje eventualnih slabosti algoritma i

dalje usavršavanje implementacije. Primer gde bi se klasa mogla primeniti je projektovanje električnih kola. Ovaj problem zahteva da kolo ima isto radno stanje, samo je jedan način pakovanja na štampanu ploču pogodniji od drugog, zbog uštede u materijalu, na prostoru, itd. Električno kolo bi se moglo projektovati ne vodeći računa o ovim ograničenjima, a računar bi zatim sam tražio izomorfan oblik koji najviše odgovara postavljenim kriterijumima.

Literatura

Cvetković, D.M., Lacković I.B, Merkle, M.J., Radosavljević, Z.S., Simić, S.K., Vasić, K.S. 1986. *Matematika I – Algebra*. Izdavači su autori.

Cvetković, D.M. 1986. *Teorija grafova i njene primene*. Beograd: Naučna knjiga.

Wiitala, A.S. 1987. *Discrete mathematics – A Unified Approach*. McGraw-Hill.

Vladimir Novosel

Determining the Isomorphism of Graphs

It is a well-known fact that examining graph isomorphism has high practical relevance. In this paper the new algorithm is proposed. The main idea was to simplify the process of determining the isomorphism by generating new graphs from the original ones, since the isomorphism of the former is less complex to determine. First, the general criteria were determined that an implementation of algorithm should meet. Afterwards, several possible ways of implementation were examined. One among them was chosen to be presented herein.