PROBLEM MAKSIMALNOG PAKOVANJA TROUGLOVA

NIKOLA NIKOLIĆ

1 UVOD

Prvo ćemo definisati problem maksimalnog pakovanja skupova. Data je populacija P i familija njenih podskupova F. Potrebno je izabrati najveću moguću potfamiliju $R \subseteq F$ uzajamno disjunktnih skupova tj. $r \cap s = \varnothing$ za sve različite $r, s \in R$. U slučaju da svaki podskup iz F ima tačno k elemenata, reč je o problemu maksimalnog pakovanja k-skupova. Za $k = 2$, to je problem maksimalnog uparivanja u grafu, koji se može rešiti u polinomijalnom vremenu. Za $k > 2$, poznato je da je ovaj problem NP-kompletan[1].

U problemu maksimalnog pakovanja trouglova (eng. *Maximum triangle packing*, skraćeno *MTP*) dat je graf i potrebno je izabrati najveći mogući skup trouglova (tj. potpunih podgrafova na tri čvora) koji se međusobno ne dodiruju (nemaju zajedničke čvorove). Vidimo da je MTP ekvivalentan problemu maksimalnog pakovanja 3-skupova, gde je populacija jednaka skupu čvorova. Postoji i drugi problem poznat pod istim nazivom, gde se razmatra potpun težinski graf i traži se skup trouglova maksimalne težine. Neki teoretski rezultati vezani za aproksimaciju MTP dati su u [2] i [3]. Poznato je da je ovaj problem i dalje NP-težak čak i ako se maksimalni broj grana po čvoru ograniči na 4[4]. U radu [5] dat je veći broj algoritama za aproksimaciju MTP, kao i egzaktan algoritam koji koristi metodu grananja sa odsecanjem.

2 OPIS REŠENJA

Najpre je potrebno izdvojiti skup trouglova iz grafa. Za svaku granu (u, v) i svaki čvor w se proverava da li su grane (v, w) i (w, u) takođe prisutne. Ako jesu, (u, v, w) se dodaje u listu trouglova. Ovaj postupak je složenosti $O(nk)$, gde je n broj čvorova, a k broj grana u grafu.

Predstavićemo algoritam koji efikasno aproksimira MTP. Osnovna ideja je sledeća. Za datu listu trouglova, u svakom koraku biramo jedan trougao i uzimamo ga u rešenje. Tom prilikom je potrebno izbaciti iz liste taj trougao i sve one koji sa njim dele čvor. Kada početna lista ostane prazna, dobijeno je jedno rešenje tj. jedno pakovanje trouglova.

Definišimo sada neke vrednosti na osnovu kojih će se vršiti izbor trougla u svakom koraku (definicije preuzete iz [5]). Za čvor sa indeksom i, α_i predstavlja broj trouglova kojima taj čvor pripada. Za trougao sa indeksom i, β_i (koje ćemo nazvati *težina*) predstavlja zbir α_i njegova tri čvora. U svakom koraku osnovnog algoritma računamo α_i i β_i i uzimamo trougao sa minimalnom težinom.

Trougao koji ima veći broj suseda će imati veću težinu. Ideja je da se prvo izaberu trouglovi sa "periferije" grafa, čiji izbor neće isključiti mnoge druge trouglove. Ovaj pristup je sličan algoritmu za određivanje donje granice optimalnog rešenja *LB1* iz [5], s tim što se tamo za izbor trougla koristi vrednost λ_i, koja predstavlja stepen susedstva trougla (tj. broj trouglova koji su mu susedni). β_i i λ_i su slična merila. Naime, za dati trougao, svaki njegov sused povećava vrednost λ_i za 1, a β_i za 1 ili 2, u zavisnosti od broja čvorova koji im je zajednički.

Navedene vrednosti se mogu izračunati u $O(m)$, gde je m broj trouglova u listi. Prvo za sve i postavi $\alpha_i = 0$ i zatim se, prolazeći kroz listu, za trougao (a, b, c) vrednost $\alpha_a, \alpha_b, \alpha_c$ povećaju za jedan. Prilikom drugog prolaska kroz listu, za i-ti trougao (a,b,c) se postavlja $\beta_i = \alpha_a + \alpha_b + \alpha_c$.

3 EKSPERIMENTALNI REZULTATI

Radi testiranja algoritma, slučajno je generisan određen broj grafova. Grafovi su generisani uniformnom metodom, tako da se svaka moguća grana dodaje u graf sa verovatnoćom p. Svaki graf je zadat brojem čvorova n i očekivanim brojem trouglova t, na osnovu kojeg se određuje verovatnoća p. Za svaku vrednost n korišćene su tri različite vrednosti t: $n/2$, n i $2n$. Grafovi su dati u sledećoj tabeli:

Test primeri

Graf	n	t
R1	400	200
R2	400	400
R3	400	800
R4	800	400
R5	800	800
R6	800	1600
R7	1200	600
R8	1200	1200
R9	1200	2400
R10	1600	800
R11	1600	1600
R12	1600	3200
R13	2000	1000
R14	2000	2000
R15	2000	4000

Testiranje je vršeno na procesoru Intel i3-4130, sa 8 GB RAM-a. Korišćen je kompilator g++ na operativnom sistemu Windows 7. Vršeno je poređenje sa algoritmom *LB1*, koji se pokazao kao najbolji aproksimativni algoritam za ovaj problem u smislu tačnosti i brzine. U sledećoj tabeli dati su broj trouglova u nađenom rešenju i vreme izvršavanja u milisekundama za algoritam LB1 i opisani algoritam LBβ.

Rezultati

Graf	$n(LB1)$	$t(LB1)$	$n(LB\beta)$	$t(LB\beta)$
R1	68	78	69	1
R2	92	265	90	16
R3	116	1406	118	16
R4	145	499	144	16
R5	194	2278	189	15
R6	228	9705	226	47
R7	220	1772	219	31
R8	289	7765	290	36
R9	344	30947	343	87
R10	278	3431	276	31
R11	384	19766	383	63
R12	455	68070	453	154
R13	375	8630	375	59
R14	485	36845	488	109
R15	574	150050	574	234

Kao što smo pretpostavili, dva algoritma daju vrlo slične rezultate. Razlika je u složenosti izračunavanja koje se ponavlja u svakom koraku algoritma. Naime, α_i i β_i se izračunavaju u $O(m)$, gde je m broj trouglova u listi, a izračunavanje λ_i je u $O(m^2)$. Ovo objašnjava velike razlike u vremenu izvršavanja.

4 GENETSKI ALGORITAM

U ovom poglavlju predstavljamo genetski algoritam za rešavanje MTP. Svaka jedinka (izbor trouglova) je kodirana kao niz bitova, gde i-ti bit ima vrednost 1 ako je i-ti trougao izabran, a 0 inače. Pogodnost jedinke je broj jedinica u tom nizu. Prvo je generisana početna populacija od 100 validnih rešenja (ne sadrže trouglove koji se međusobno dodiruju) na slučajan način. Korišćena je ruletska selekcija (svaka jedinka se bira za roditelja sa verovatnoćom proporcionalnoj njenoj pogodnosti). Vršeno je jednopoziciono ukrštanje sa verovatnoćom mutacije 0.1 i dobijene jedinke su "popravljane" tako da predstavljaju validna rešenja. Primenjen je elitistički pristup: najboljih 30 jedinki iz generacije se zadržava u sledećoj. Rad algoritma se završava kada se ne postigne poboljšanje u 100 uzastopnih generacija.

Radi poređenja, implementiran je algoritam grube sile. U ovom algoritmu se obilazi stablo koje predstavlja sva moguća rešenja, a odsecaju se jedino grane koje predstavljaju nevalidna rešenja. Testiranje je vršeno na slučajnim grafovima kao u poglavlju 3, sa zadatim vrednostima $t = n$. Rezultati testiranja su dati u narednoj tabeli. Za poslednja tri grafa rad algoritma grube sile je prekinut posle 5 minuta, tako da njegovo rešenje nije nužno optimalno. Vidimo da genetski algoritam daje skoro iste rezultate kao egzaktni algoritam, a izvršava se daleko brže počevši od veličine ulaza oko 40-50.

Rezultati

$\|V\|$	$n(BF)$	$t(BF)$	$n(GA)$	$t(GA)$
10	3	0	3	124
20	5	15	5	156
30	8	76	8	117
40	9	208	9	143
50	11	2508	11	198
60	14	30761	13	256
70	15	294303	15	221
100	22	300000*	21	375
1000	204	300000*	203	18134
2000	411	300000*	412	66755

5 ZAKLJUČAK

Opisani algoritam daje približno isti kvalitet rešenja kao najbolji poznati algoritmi za aproksimaciju i efikasan je. Ovo ga čini pogodnim u slučajevima kada je vreme izvršavanja važno. Takođe bi trebalo da se koristi pri određivanju donje granice rešenja u pristupu grananja sa odsecanjem, jer u nekim slučajevima predstavlja malo poboljšanje u odnosu na ostale takve algoritme (što omogućava efikasnije potkresivanje stabla).

Pošto se ovaj osnovni pristup (odabir trougla i brisanje suseda iz liste) pokazao kao najbolji, dalje istraživanje bi moglo da se bavi izborom funkcije na osnovu koje se bira trougao, tj. možda se može koristiti neka funkcija koja je bolji prediktor "pogodnosti" trougla od λ_i i β_i.

LITERATURA

[1] M. R. Garey & D. S. Johnson. Computers and intractability. a guide to the theory of np-completeness. 1979.

[2] M. V. Ashley et al. On approximating four covering and packing problems. 2009.

[3] G. Manić & Y. Wakabayashi. Packing triangles in low degree graphs and indifference graphs. 2008.

[4] A. Caprara & R. Rizzi. Packing triangles in bounded degree graphs. information processing letters. 2002.

[5] Y. Abdelsadek et al. Branch-and-bound algorithm for the maximum triangle packing problem. 2015.