Rešavanje NP-kompletnih problema svođenjem

— Master teza —

Aleksandar Zeljić

10. oktobar 2011

mentor:
dr Predrag Janičić

članovi komisije:
dr Miodrag Živković
dr Filip Marić
dr Predrag Janičić

Matematički fakultet
Univerzitet u Beogradu
2011

Sadržaj

Glava 1

Uvod

NP-kompletni problemi zauzimaju posebno mesto u računarstvu. Oni predstavljaju teške probleme odlučivanja (za čije rešavanje nije poznat efikasan algoritam) i međusobno su svodivi u polinomijalnom vremenu.

U NP-kompletne probleme spadaju mnogi problemi od praktičnog značaja, kao što su razni problemi pokrivanja, particionisanja, planiranja i raspoređivanja. Veoma je bitno da se za njihovo rešavanje nađu što brža rešenja. Jedan od najizučavanijih NP-kompletnih problema — SAT problem tu igra vrlo značajnu ulogu, jer poslednjih decenija razvijene su veoma efikasne implementacije za njegovo rešavanje. Kako je SAT NP-kompletan problem, moguće je izvršiti njegovo svođenje na bilo koji drugi NP-kompletan problem u polinomijalnom vremenu, a i obratno. Na primer, problem klike, problem Hamiltonovog ciklusa i drugi se mogu svesti na SAT problem. Ovo omogućava da se SAT rešavači koriste kao univerzalni alat, npr. ukoliko ne postoji specijalizovani rešavač za neki problem, kao što je problem usmerenog Hamiltonovog cilklusa (u daljem tekstu — UHC). Praktično svi rešavači za problem Hamiltonovog ciklusa rešavaju neusmerenu varijantu problema (u daljem tekstu — NHC). Pored toga, moderni SAT rešavači imaju primene u mnogim oblastima i uspešno rešavaju i instance od nekoliko miliona klauza. Ipak, neke instance problema SAT ostaju preteške za SAT rešavače. Pitanje je da li neke od njih mogu efikasnije da se reše svođenjem na neki drugi NP-kompletan problem, upotrebom odgovarujećeg rešavača. Takođe, interesantno pitanje je da li moderni SAT rešavači mogu biti efikasniji za rešavanje nekih instanci drugih NP-kompletnih problema od specijalizovanih rešavača. Interesantno je ispitati i da li najteže instance jednog problema (u nekom skupu) ostaju najteže i nakon svođenja na neki drugi problem.

Glava 2

NP-kompletnost i polinomijalna svođenja

U rešavanju problema, u računarstvu se teži efikasnijim (u smislu vremena i memorijskog prostora) algoritmima za njihovo rešavanje. Međutim, za neke probleme nisu pronađena elegantna i efikasna rešenja, štaviše postoje razlozi za verovanje da efikasna rešenja i ne postoje za te probleme. U nastavku će prvo biti izložene klase složenosti, a zatim pojam NP-kompletnosti [7, 8].

2.1 Klase složenosti

Efikasnost algoritma se može meriti brojem koraka (vremenska složenost) i npr. brojem bitova (memorijska ili prostorna složenost) potrebnih za izvršenje algoritma u funkciji veličine ulaza. Međutim, kako izvršavanje algoritma može varirati u zavisnosti od instance do instance problema, potreban je nekakav način da mere efikasnosti budu nezavisne od specifičnosti instanci. Mogla bi se vršiti procena najboljeg, prosečnog i najgoreg slučaja. Ocena najboljeg slučaja retko kada predstavlja dobru meru efikasnosti algoritma, pa se nikada i ne koristi. Ocena prosečnog slučaja kao ocenu koristi prosečan broj koraka u algoritmu i u nekim situacijama je poželjna, međutim često zahteva vrlo složena izračunavanja, te se stoga retko primenjuje. Preostaje ocena najgoreg slučaja, koja predstavlja najveći broj koraka ili memorijski prostor potreban za neko izvršavanje algoritma. Ocena najgoreg slučaja daje gornju granicu kao meru složenosti algoritma i jednostavna je za izračunati (u odnosu na ocenu prosečnog slučaja). Za izražavanje asimptotskog ponašanja gornje granice složenosti koristi se tzv. "veliko O notacija". Ako postoji konstanta c i prirodni broj n_0 takvi da za pozitivne funkcije f i g nad domenom prirodnih brojeva važi: $f(n) \leq c \cdot g(n)$ za sve vrednosti n veće od n_0 piše se $f = O(g)$ i čita se "f je veliko O od g". "veliko O notacija" predstavlja klasu funkcija koje imaju istu gornju granicu. Za izražavanje asimptotskog ponašanja donje granice složenosti koristi se tzv. "omega" notacija. Ako postoji pozitivna konstanta c i prirodan broj n_o takva

da za funkcije f i g važi: $f(n) = c \cdot g(n)$, za svako $n > n_o$, onda se kaže da je funkcija $g(n)$ asimptotska donja granica funkcije $f(n)$ i piše se $f(n) = \Omega(g(n))$. Za funkcije f i g se kaže da imaju iste asimptotske brzine rasta, ukoliko istovremeno važi i $f = O(g(n))$ i $f = \Omega(g(n))$, što se označava $f(n) = \Theta(g(n))$.

Problem se naziva *rešivim*, ukoliko postoji *efikasan* algoritam za njegovo rešavanje. Za algoritam se kaže da je *efikasan* ili *polinomijalan*, ukoliko je njegova vremenska složenost $O(P(n))$, gde je $P(n)$ polinom od veličine problema. Klasa svih problema koji se mogu rešiti efikasnim (tj. polinomijalnim) algoritmom se označava sa P.

2.2 Polinomijalna svođenja

Problemi odlučivanja su problemi za čiju svaku instancu se očekuje odgovor DA ili NE. Većina problema se može relativno jednostavno izraziti pomoću problema odlučivanja. Svaki problem odlučivanja se može razmatrati kao *problem prepoznavanja jezika*. Neka je skup U skup svih mogućih ulaza za problem odlučivanja, a $L \subset U$ skup svih ulaza za koje je rešenje problema DA. Za L se tada kaže da je jezik koji odgovara problemu. Problem odlučivanja je onda problem utvrđivanja da li ulaz pripada jeziku L.

Neka su L_1 i L_2 dva jezika, takvi da $L_1 \subset U_1$ i $L_2 \subset U_2$. Kaže se da je L_1 *polinomijalno svodljiv* na L_2, ako postoji polinomijalni algoritam koji dati ulaz $u_1 \in U_1$ prevodi u ulaz $u_2 \in U_2$, tako da važi $u_1 \in L_1$ akko $u_2 \in L_2$ Ako je jezik L_1 polinomijalno svodljiv na jezik L_2 i ako postoji polinomijalni algoritam za prepoznavanje L_2, onda postoji i polinomijalni algoritam za prepoznavanje L_1. Algoritam se dobija primenom algoritma svođenja jezika L_1 na L_2 i primenom algoritma za prepoznavanje jezika L_2 na izlaz dobijen iz algoritma svođenja. Relacija polinomijalne svodljivosti nije simetrična, jer definicija polinomijalne svodljivosti zahteva samo da se proizvoljan ulaz za L_1 transformiše u ulaz za L_2, dok obrnuto ne mora da važi. Može se smatrati da je problem na koji se vrši svođenje teži ili iste težine. Takođe, relacija polinomijalne svodljivosti je tranzitivna.

Za dva jezika L_1 i L_2 se kaže da su *polinomijalno ekvivalentni*, ako je svaki od njih polinomijalno svodljiv na drugi. Relacija polinomijalne ekvivalencije je od velikog značaja za probleme koji nisu u klasi P, jer praktično označava probleme jednake težine.

2.3 Klasa NP

Klasa NP se neformalno definiše pomoću pojma *nedeterminističkog algoritma*, koji se sastoji iz dve faze:

1. Faza pogađanja — na neki način se bira struktura S, u odnosu na dati ulaz u.

2. Faza provere — za ulaz u i strukturu S, daje odgovor DA ili NE (ili se izavršava beskonačno dugo).

Za nedeterministički algoritam se kaže da "rešava" problem odlučivanja Π, ako su za proizvoljni ulaz $u \in U_\Pi$ ispunjena sledeća dva uslova:

1. Ako $u \in L_\Pi$, onda postoji takva struktura S, takva da faza provere za pogađenu strukturu S i ulaz u završava odgovorom DA.

2. Ako $u \notin L_\Pi$, onda ne postoji struktura S čijim bi se pogađanjem faza provere za strukturu S i ulaz u završila odgovorom DA.

Za nedeterministički algoritam se kaže da se "izvršava u polinomijalnom vremenu", ako postoji polinom p takav da za svaki ulaz $u \in L_\Pi$ postoji pogađanje strukture S takvo da se faza provere algoritma završava odgovorom DA za vreme $p(|u|)$, gde $|u|$ označava veličinu ulaza u. Klasa NP je klasa svih problema odlučivanja koji pri pogodnom kodiranju mogu biti rešeni nedeterminističkim algoritmom za polinomijalno vreme (eng. *NP — nondeterministic polynomial*). Čini se da su nedeterministički polinomijalni algoritmi moćniji od determinističkih polinomijalnih, međutim nikome do sad to nije pošlo za rukom da dokaže ili opovrgne. Da bi se to pokazalo potrebno je pronaći neki problem iz NP koji nije u P. Da bi se pokazalo da su klase P i NP jednake potrebno je pokazati se svaki problem iz klase NP može rešiti determinističkim algoritmom polinomijalne vremenske složenosti. Problem odnosa klasa P i NP je jedan od značajnijih u računarstvu i poznat je kao problem "$P = NP$?".

2.4 NP-kompletnost

Među problemima koji se nalaze u klasi NP mogu se izdvojiti "najteži" problemi. se kaže da je *NP-težak* ukoliko se svaki problem Y koji pripada klasi NP može svesti na X u polinomijalnom vremenu. Ako se za bilo koji NP-težak problem dokaže da je u P, to bi značilo da je $P = NP$. Za problem X se kaže da je *NP-kompletan* ako X pripada klasi NP i ako je X NP-težak. Definicija NP-kompletnosti ne daje jednostavan način za utvrđivanje da li problem pripada NP-kompletnim problemima. NP-težak dolazi se do oblika definicije koji je lakši za primenu, međutim zahteva da bude poznat bar jedan NP-kompletan problem. Problem zadovoljivosti iskazne formule — SAT (eng. *satisfiability*) je prvi problem za koji je dokazano da je NP-kompletan. Taj dokaz spada među najznačajnije rezultate teorijskog računarstva i prvi ga je izveo Kuk 1971. godine [1], a nezavisno od njega i Levin [4]. Ubrzo zatim je za veliki broj drugih problema dokazano da su NP-kompletni, npr. problem 3-SAT, problem Hamiltonovog ciklusa, problem klika, problem bojenja grafa, problem celobrojnog $(0-1)$ programiranja, problem pokrivanja čvorova i mnogi drugi [2].

Glava 3

SAT problem i rešavači za SAT problem

U okviru ove glave su ukratko izloženi osnovni pojmovi iskazne logike, SAT problem, kao i pregled različitih postojećih SAT rešavača i njihovih karakteristika.

3.1 Kratak uvod u iskaznu logiku

U iskaznoj logici korisite se promenljive da bi predstavile iskaze, koji se mogu kombinovati u složenije iskaze. Postoje tri aspekta iskazne logike: sintaksa, semantika i deduktivni sistemi. U nastavku će ukratko biti izložene samo sintaksa i semantika[7]. Naime, SAT problem je zadat u terminima semantike i u njegovom rešavanju se obično ne koriste deduktivne metode.

3.1.1 Sintaksa iskazne logike

Sintaksa iskazne logike opisuje jezik iskazne logike, tj. na koji način se formiraju sintaksno ispravne iskazne formule.

Azbuku iskazne logike Σ čine:

- prebrojiv skup iskaznih slova P;
- skup logičkih veznika $\{\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$, gde je $\neg$ unarni veznik, a ostali su binarni;
- skup logičkih konstanti $\{\top, \bot\}$;
- skup pomoćnih simbola $\{(,)\}$.

Skup iskaznih formula nad skupom iskaznih slova P je najmanji podskup skupa svih reči nad azbukom Σ takav da važi:

- iskazna slova i logičke konstante su iskazne formule;
- ako su A i B iskazne formule onda su to i: $(\neg A), (A \wedge B), (A \vee B)$, $(A \Rightarrow B), (A \Leftrightarrow B)$.

Iskazna slova se nazivaju i *iskaznim promenljivim*. Iskazna slova i logičke konstante se nazivaju *atomičkim iskaznim formulama*, a atomička formula ili njena negacija se naziva *literal*. Disjunkcija literala se naziva *klauza*.

Za iskaznu formulu se kaže da je u *konjunktivnoj normalnoj formi* (*KNF*) ako je oblika: $A_1 \wedge A_2 \wedge \ldots \wedge A_n$ gde je svaka od formula $A_i (1 \leq i \leq n)$ klauza. Slično, za iskaznu formulu se kaže da je u *disjunktivnoj normalnoj formi* (*DNF*) ako je oblika: $A_1 \vee A_2 \vee \ldots \vee A_n$ gde je svaka od formula A_i $(1 \leq i \leq n)$ konjunkcija literala. Za svaku iskaznu formula postoji logički ekvivalentna iskazna formula u KNF (odnosno u DNF). KNF je od posebnog značaja, jer većina modernih SAT rešavača prima kao ulaz formule u KNF.

3.1.2 Semantika iskazne logike

Semantika iskazne logike govori o značenju formula.

Funkcije $v : P \mapsto \{0, 1\}$ se nazivaju valuacijama, a skup $\{0, 1\}$ domenom valuacije. Kao domen valuacije se može upotrebiti i bilo koji drugi dvočlani skup. Svaka valuacija v definiše funkciju interpretacije I_v, koja preslikava skup iskaznih formula u skup $\{0, 1\}$. Funkcija interpretacije se definiše na sledeći način:

- $I_v(p) = v(p)$, za svako iskazno slovo $p \in P$;
- $I_v(\top) = 1$;
- $I_v(\bot) = 0$;
- $I_v(\neg A) = 1$ ako je $I_v(A) = 0$, i $I_v(\neg A) = 0$, ako je $I_v(A) = 1$;
- $I_v(A \wedge B) = 1$ ako je $I_v(A) = 1$ i $I_v(B) = 1$; $I_v(A \wedge B) = 0$ inače;
- $I_v(A \vee B) = 0$ ako je $I_v(A) = 0$ i $I_v(B) = 0$; $I_v(A \vee B) = 1$ inače;
- $I_v(A \Rightarrow B) = 0$ ako je $I_v(A) = 1$ i $I_v(B) = 0$; $I_v(A \Rightarrow B) = 1$ inače;
- $I_v(A \Leftrightarrow B) = 1$ ako je $I_v(A) = I_v(B)$; $I_v(A \Leftrightarrow B) = 0$ inače;

Za formulu A kažemo da je *tačna* u valuaciji v ukoliko je $I_v(A) = 1$, a za valuaciju v se tada kaže da je *zadovoljavajuća* za formulu A. Ukoliko je $I_v(A) = 0$, onda kažemo da je formula A *netačna* u valuaciji v.

Za formulu se kaže da je *zadovoljiva* ukoliko postoji valuacija koja je za nju zadovoljavajuća. Ukoliko je svaka valuacija zadovoljavajuća za iskaznu formulu onda se za tu formulu kaže da je *tautologija*. Formula je *kontradikcija* (ili *nezadovoljiva*) ukoliko za nju ne postoji zadovoljavajuća valuacija.

Primer 1. *Formula $A \vee \neg A$ je tautologija, formula $A \wedge \neg A$ je kontradikcija, a formula $A \Rightarrow B$ je zadovoljiva.*

3.2 SAT problem

Problem ispitivanja zadovoljivosti iskazne formule, se naziva SAT problem (od eng. *satisfiability*). SAT problem je veoma težak problem u pogledu izračunljivosti, jer nije poznat efikasan (tj. polinomijalan) algoritam za njegovo rešavanje. Štaviše, SAT problem je prvi problem za koji je dokazano da je NP-kompletan. To je dokazao Kuk 1971. godine [1]. Ruski naučnik Levin je nezavisno došao do istog otkrića razmatrajući univerzalne probleme pretrage u svom radu [4], pa se to tvrđenje još naziva i Kuk-Levinova teorema.

Pored velikog značaja za teorijsko računarstvo, SAT problem ima jako široku praktičnu primenu, zbog velikog broja efikasnih rešavača. Razni teški problemi se kodiraju u vidu iskaznih formula, koje se potom rešavaju SAT rešavačima. Ovaj pristup je posebno važan za probleme za koje ne postoje efikasna specijalizovana rešenja. Neke od primena SAT problema su verifikacija logičkih kola i rešavanje raznih problema planiranja.

3.3 SAT rešavači

SAT rešavači su implementacije raznih metoda i algoritama za rešavanje SAT problema. Postoje npr. DPLL zasnovani, stohastički, ne-KNF i drugi SAT rešavači. DPLL zasnovani rešavači se još nazivaju i CDCL (eng. *conflict driven clause learning*) rešavačima.

Kompletni SAT rešavači mogu utvrditi da li je formula zadovoljiva ili nezadovoljiva, dok stohastički često mogu brže utvrditi da je formula zadovoljiva, ali ne mogu dati odgovor ukoliko je ona nezadovoljiva. Stohastički rešavači upotrebljavaju pristupe kao što je gramziva lokalna pretraga. Ideja gramzive lokalne pretrage je da se bira slučajna valuacija kao trenutna i onda se razmatraju sve valuacije koje se za jednu promenljivu razlikuju od trenutne. Kao novu trenutnu valuaciju bira se ona koja zadovoljava najveći broj klauza.

Većina kompletnih SAT rešavača je zasnovana na DPLL (Dejvis-Patnam-Logman-Lovland) proceduri [13]. Klasična DPLL procedura kao ulaz prima formulu u KNF, u vidu multiskupa (tj. u praktičnoj implementaciji u vidu liste) klauza[1]. Kao izlaz daje odgovor DA ukoliko je formula zadovoljiva, a inače daje odgovor NE. Moderni CDCL rešavači koriste verziju DPLL procedure koja umesto da prosleđuje i menja formulu (tj. multiskup klauza), prosleđuje i menja valuaciju. Ova izmena omogućuje mnogo efikasniji rad procedure, naročito u slučaju iskaznih formula koje sadrže veliki broj klauza [18].

Moderni SAT rešavači koriste tzv. *trag* (eng. *trail*) — listu promenljivih koja predstavlja parcijalnu valuaciju formule (promenljive u listi imaju dodeljene vrednosti, dok preostale promenljive nemaju dodeljene vrednosti). Kada je svakoj promenljivoj formule dodeljena vrednost, za valuaciju se kaže da je totalna valuacija. Primena *split* pravila DPLL algoritma se naziva *odlukom*, jer se

[1] DIMACS format, koji se najčešće koristi za formule u KNF je opisan u okviru glave 9 — Dodatak

```
function  DPLL (v : valuacija):  (DA,NE)
begin
  if F je netačno u valuaciji v then return NE
  else if v je totalna valuacija formule F then return DA
  else if postoji jedinična klauza (l ∨ l_1 ∨ ... ∨ l_k u F t.d.
     l, l̄ ∉ v, l̄_1, ..., l̄_k ∈ v)  then return DPLL(v ∪ {l})
  else begin
     izaberi literal l t.d. l ∈ F, l, l̄ ∉ v
     if DPLL(v ∪ {l} )=DA then return DA
     else return DPLL(v ∪ {l̄})
  end
end
```

Slika 3.1: Pseudo kod modifikovane DPLL procedure koja prosleđuje valuaciju, a ne formulu

u tom trenutku bira nova promenljiva koja će biti dodata u listu, tj. biće joj dodeljena vrednost u parcijalnoj valuaciji. Osim izbora promenljive koja se dodaje parcijalnoj valuaciji, potrebno je odabrati i njenu vrednost tj. *polaritet*. Primenom pravila *unit propagation* se vrednost nekog literala dodeljuje na osnovu vrednosti koje se nalaze u valuaciji, pa se takav zaključak naziva *implikacijom*. Ukoliko se javi situacija da jedna promenljiva ima implicirane različite vrednosti onda ona proizvodi *konflikt*. Konflikt označava da je u nekom trenutku napravljena pogrešna odluka, onda se može naći razlog, u vidu klauze, zbog kog je do konflikta došlo i tom klauzom proširiti početni skup klauza kako bi se sprečilo ponovno ispitivanje tog potprostora pretrage. Takve klauze se nazivaju *naučene klauze*. Kada se detektuje konflikt, u rešavanju se vrši *skok unazad* (eng. *backjump*) do poslednje odluke koja je dodelila vrednost promenljivoj koja učestvuje u konfliktu. Ovo je moguće izvesti jer sve dodele vrednosti posle te odluke su implicirane upravo tom odlukom. Posle nekog vremena broj naučenih klauza može znatno usporiti rad rešavača, pa je poželjno da neke klauze (npr. one koje se dugo nisu koristile) budu obrisane. Ovaj postupak se naziva *zaboravljanje*. Rešavač može i da posle izvesnog vremena krene pretragu od početka (pri tome zadržavajući naučene klauze, kako se ne bi vratio u isti deo prostora pretrage) sa očekivanjem da se rešenje ne nalazi u delu prostora pretrage koji je ispitivao. Ovaj postupak se naziva *kretanje iznova* (eng. *restart*). Međutim, kako se ne bi narušila kompletnost pretrage, neki rešavači što duže rade sve ređe će se vršiti započinjanje iznova.
Informacije o trenutno najboljim SAT rešavačima se mogu pronaći na adresi oragnizacije za takmičenja SAT rešavača [2]. Među poznatijim rešavačima su zChaff [3], Minisat[4], SATzilla[5], clasp[6] i drugi.

[2] http://www.satcompetition.org/
[3] www.princeton.edu/~chaff/zchaff.html
[4] www.minisat.se/MiniSat.html
[5] www.cs.ubc.ca/labs/beta/Projects/SATzilla/
[6] www.cs.uni-potsdam.de/clasp/

Glava 4

Problem Klika i rešavači za problem Klika

U ovoj glavi biće ukratko izloženi osnovni pojmovi teorije grafova, zatim problem k-klike i rešavač za problem k-klike [8].

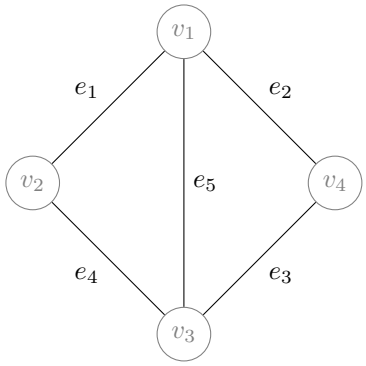
4.1 Osnovni pojmovi teorije grafova

Uređeni par $G = (V, E)$ se naziva *grafom*, gde je $V = \{v_1, v_2, ..., v_n\}$ skup *čvorova*, a $E \subseteq V \times V$ skup *grana*. Broj čvorova grafa se smatra veličinom grafa. Graf može biti *usmeren* ili *neusmeren*, u zavisnosti od toga da li je bitan redosled čvorova grana, odnosno da li važi $(v_i, v_j) \in E \iff (v_j, v_i) \in E$ za svaku granu $(v_i, v_j) \in E$, ukoliko važi — graf je neusmeren, a u suprotnom je usmeren. Čvorovi grafa se najčešće ilustruju krugovima, a grane linijama koje povezuju odgovarajuće čvorove. Usmerene grane imaju strelicu u pravcu čvora u koji ulaze.

Za granu $e = (v_i, v_j)$ se kaže da je *susedna* ili *incidentna* čvorovima v_i i v_j, a za čvorove v_i i v_j da su *susedni*. Broj grana susednih jednom čvoru se naziva *stepenom* tog čvora. Kod usmerenih grafova razlikuju se *ulazni* i *izlazni stepen* čvora, odnosno broj grana koje ulaze, tj. izlaze iz datog čvora.

Put u grafu je niz susednih čvorova povezanih granama. Za put kažemo da je *prost* ukoliko se nijedan čvor ne ponavlja duž njega. *Ciklus* čini prost put i grana koja povezuje poslednji i prvi čvor tog puta. Neusmereni graf je *kompletan* ili *kompletno povezan* ako postoji grana između svaka dva čvora u grafu. Za graf $G' = (V', E')$ kaže se da je podgraf grafa $G = (V, E)$, ako je $V' \subseteq V$ i $E' \subseteq E$. Za graf $G' = (V', E')$ se kaže da je *podgraf grafa G indukovan skupom čvorova V'* ukoliko je $V' \subset V$ i $E' = E \cap (V' \times V')$. Granama grafa se mogu pridružiti vrednosti koje se nazivaju *težine* (eng. *weights*), a takav graf se naziva *težinskim grafom* (eng. *weighted graph*). *Klika* je kompletno povezan podgraf, ili skup čvorova takav da su svi međusobno susedni. Klika veličine k se naziva *k-klika*.

Primer 2. *Neusmereni graf na slici 3.1 čini skup čvorova* $V = \{V_1, v_2, v_3, v_4\}$ *i skup grana* $E = \{e_1, e_2, e_3, e_4, e_5\}$. *Na primer, za granu* e_1 *se kaže da je susedna čvorovima* v_1 *i* v_2. *Za čvorove* v_1 *i* v_2 *se takođe kaže da su susedni. Niz čvorova i grana —* $< v_2, e_1, v_1, e_5, v_3, e_4, v_4 >$ *čini put u grafu. Dok zatvoreni put* $< v_2, e_1, v_1, e_2, v_4, e_3, v_3, e_4, v_2 >$ *čini ciklus u prikazanom grafu. U datom grafu se mogu videti dve klike veličine 3 i to su:* v_1, v_2, v_3 *i* v_1, v_3, v_4.

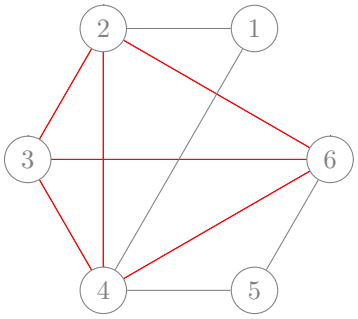


Slika 4.1: Jednostavan graf

4.2 Problem Klika

Problem klika je značajan grafovski problem. Neka je dat neusmereni graf G, problem klika ili problem k-klike je problem utvrđivanja da li u grafu G postoji kompletan podgraf G' veličine bar k. Dakle, problem klika je problem odlučivanja i pripada NP-kompletnim problemima [2]. Kako klika veličine k sadrži k klika veličine $k-1$, uvodi se pojam maksimalne klike. *Maksimalna klika* je ona koja nije deo nijedne druge klike. Problem maksimalne klike je NP-težak problem, jer ako se može rešiti problem maksimalne klike grafa, onda se zna i rešenje problema k-klike. Ukoliko je veličine maksimalne klike veća ili jednaka broju k, onda postoji k-klika u grafu, a u suprotnom ne postoji.

Primer 3. *Na narednoj slici prikazan je graf sa šest čvorova, crvenom bojom prikazane su grane klike veličine 4 u tom grafu. Možemo primetiti da se klika veličine 4 (koju čine čvorovi 2,3,4,6) sastoji od 4 klike veličine 3, ali i da ne postoji klika veličine 5 koja je sadrži te je ona ujedno i maksimalna klika u datom grafu, međutim ne i jedina. Klike veličine 3 koje čine čvorovi 1,4,2 i 4,5,6 su takođe maksimalne u ovom grafu.*



Slika 4.2: Graf koji sadrži 4-kliku.

4.3 Rešavači za problem Klika

Može se pronaći svega nekoliko rešavača za problem klike. Cliquer [1] koristi egzaktni algoritam grananja sa ograničavanjem (odsecanjem), za pronalaženje maksimalne klike i pojednostavljene verzije algoritma za lakše probleme, kao što je klika određene veličine [9]. Pretpostavlja se nekakav poredak među čvorovima, $V = v_1, v_2, \ldots, v_n$. Razmatraju se podgrafovi indukovani skupovima čvorova $S_i = v_1, v_2, \ldots, v_i$. Funkcija $c(i)$ se definiše kao veličina maksimalne klike u podgrafu indukovanom skupom S_i. Može se primetiti da je $c(i+1)$ jednako ili $c(i)$ ili $c(i)+1$, pri čemu $c(i+1) = c(i)+1$ ako i samo ako u podgrafu indukovanom sa S_{i+1} postoji maksimalna klika veličine $c(i)+1$ i njoj pripada čvor v_{i+1}. Dakle, kada se računa $c(i+1)$ i traži se maksimalna klika te veličine, kreće se od klike W koju čini samo čvor v_{i+1} i ona se proširuje čvorovima v_j gde je $j < i$, ukoliko su ispunjeni uslovi. Može se vršiti odsecanje prilikom pretrage ukoliko važi: $|W| + c(j) \leq c(i)$. Ako se j bira kao najveći nerazmotreni indeks koji se ne nalazi u W i uslov odescanja je ispunjen, to znači da ne postoji klika veličine $c(i)+1$. Pronalazak klike veličine $c(i)+1$ takođe vodi odsecanju. Rekonstrukcija jedne maksimalne klike u grafu je jednostavna kada je poznat niz vrednosti $c(1), c(2), \ldots, c(n)$. Kako čvor v_i pripada maksimalnoj kliki ukoliko $c(i) \neq c(i-1)$, onda se samo jednim prolazom kroz niz vrednosti može odrediti maksimalna klika. U slučaju da se traži klika veličine k, algoritam će prestati sa radom u onom trenutku kada za neko i bude važilo $c(i) = k$.
Cliquer prima ulaz u DIMACS formatu za grafove [2]. Rešava problem maksimalne klike, kao i problem pronalaženja klike čija veličina pripada nekom intervalu. Takođe, može da rešava analogne probleme na težinskim grafovima, pronalazeći najtežu maksimalnu kliku i sl. Za sve probleme ima opciju vraćanja svih mogućih rešenja. Kako vreme izvršavanja algoritma zavisi od redosleda čvorova, Cliquer nudi nekoliko strategija za definisanje poretka nad čvorovima kao što su slučajni poredak ili prema rastućem stepenu čvorova. Postoji još

[1] `http://users.tkk.fi/pat/cliquer.html`
[2] Opis DIMACS formata za iskazne formule i grafove se nalazi u okviru glave 9 — Dodatak

nekoliko rešavača, među kojima su Solver [3] i clique [4] koji su implementirani u programskom jeziku Java. Rešavač clique implementira isti algoritam koji koristi rešavač Cliquer. Rešavač Solver zapravo implementira algoritam za pronalaženje maksimalnog nezavisnog skupa (koji odgovara pronalaženju maksimalne klike u komplementarnom grafu, koji se dobija uklanjanjem postojećih grana i dodavanjem nepostojećih grana polaznog grafa) [16].

[3] www.vinnica.ua/~aplot/solver.html
[4] www.gitorious.org/mo-projects/clique

Glava 5

Problem Hamiltonovog ciklusa i rešavači za problem Hamiltonovog ciklusa

U ovoj glavi biće izloženi problem Hamiltonovog ciklusa i rešavači za problem Hamiltonovog ciklusa.

5.1 Problem Hamiltonovog ciklusa

Neka je dat graf G=(V,E). *Hamiltonov ciklus* predstavlja zatvoren put (ciklus) kroz graf takav da je svaki čvor grafa saržan tačno jednom. Postoje različite formulacije problema, u zavisnosti od toga da li je reč o usmerenom ili neusmerenom grafu. Postoji i optimizaciona varijanta ovog problema: ukoliko je zadat težinski graf, traži se najjeftiniji Hamiltonov ciklus i ovaj problem je poznat kao problem trgovačkog putnika. Za razliku od klasičnog problema Hamiltonovog ciklusa, kod problema trgovačkog putnika često se pretpostavlja da je graf potpun, tj. da postoji grana između svaka dva čvora grafa. Dodatno se razlikuju simetričan i asimetričan problem trgovačkog putnika, u zavisnosti od toga da li se težine grane (i,j) i (j,i) razlikuju ili ne za svaku granu (i,j) iz grafa. Problem Hamiltonovog ciklusa se trivijalno svodi na problem trgovačkog putnika, dodeljivanjem težine 0, ukoliko grana postoji u grafu ili težine 1 ukoliko grana ne postoji. U tekstu se skraćenicom NHC označava problem Hamiltonovog ciklusa u neusmerenom grafu, a problem Hamiltonovog ciklusa u usmerenom grafu se označava skraćenicom UHC.

5.2 Rešavači za problem Hamiltonovog ciklusa

Postoje mnogi rešavači za problem Hamiltonovog ciklusa, kao i za problem trgovačkog putnika. Postoji veliki broj raspoloživih implementacija[1] za rešavanje problema trgovačkog putnika, Hamiltonovog ciklusa i varijanti, među kojima se mogu naći egzaktni, heuristički i drugi rešavači.

HamCycles[2] je rešavač koji predstavlja implementaciju algoritma zasnovanog na *metodi više puteva* (eng. *multi-path method*) [5]. U nastavku sledi kratak opis ideje algoritma.

Neka je dat graf $G = (V, E)$, pri čemu je broj čvorova u grafu G, $n = |V|$, a broj grana $m = |E|$. Kako Hamiltonov ciklus mora da sadrži svaki čvor u grafu to znači da on mora sadržati n grana tog grafa. Dakle, metod grube sile za rešavanje problema Hamiltonovog ciklusa bi bio odabir n grana od postojećih m na sve moguće načine dok se ne nađe upravo Hamiltonov ciklus ili se iscrpi sve mogućnosti. Takav pristup bi se mogao izvesti pretragom sa vraćanjem (eng. *backtracking*). Međutim, može se primetiti da n grana Hamiltonovog ciklusa zapravo predstavljaju n parova susednih grana, tako da kada se odabere jedna grana da pripada ciklusu, za narednu je dovoljno razmatrati grane koje su incidentne sa njom. Ako se prostor pretrage razmatra kao stablo, ovim zapažanjem se broj grananja u svakom čvoru stabla smanjuje na stepen trenutno dostignutog čvora, umesto $m-k$ (gde je k broj već odabranih grana) koliko bi ih bilo u pristupu grube sile. Štaviše, umesto da se izvodi pretraga sa povratkom u prostoru grana, ona se može vršiti u prostoru čvorova. Čvorovi koji se biraju se nazivaju *tačkama spajanja* (eng. *anchor points*). Kada se odabere jedna tačka spajanja, naredna se bira među njoj susednim čvorovima. Kada se izvrši odabir n tačaka spajanja, potrebno je izvršiti proveru da li je zaista reč o Hamiltonovom ciklusu. Sledeće zapažanje koje vodi poboljšanju jeste da unutar ciklusa svaki čvor je stepena 2. Od svih grana koje su susedne čvorovima u ciklusu samo dve se smeju iskoristiti. Definiše se *segment* kao put u grafu, takav da se ne preklapa i ne deli krajnje čvorove sa drugim segmentima. Neka je data unija segmenata S u grafu G. Mogu se definisati dva skupa čvorova — $V_{in}(S)$ skup unutrašnjih čvorova (čiji stepen čvora je 2 unutar S) i $V_{ex}(S)$ skup krajnjih čvorova segmenata (čiji stepen čvora je 1 unutar S). Prilikom odabira narednog čvora dovoljno je razmatrati skup $V \setminus V_{in}$. Ideja je da se proširuju postojeći segmenti sve do trenutka kada moraju međusobno da se spoje, kako bi formirali jedinstven ciklus. Provera da li je reč o Hamiltonovom ciklusu je trivijalana i svodi se na proveru da li je dužina dobijenog ciklusa baš n, jer se ostala svojstva Hamiltonovog ciklusa induktivno održavaju.

Concorde TSP[3], koji se smatra najboljim rešavačem za problem trgovačkog putnika, koristi linearno programiranje za pronalaženje optimalne cene ciklusa, a zatim vrši pretragu kroz graf dok ne nađe ciklus sa baš tom težinom. Iako je lako prevesti problem Hamiltonovog ciklusa u problem trgovačkog putnika, problem je što je Concorde TSP rešavač za simetričan problem trgovačkog put-

[1] `www.or.deis.unibo.it/resesarch_pages/tspsoft.html`
[2] `http://bkocay.cs.umanitoba.ca/g&g/download.html`
[3] `http://www.tsp.gatech.edu/concorde.html`

nika. Postoji način da se asimetričan problem trgovačkog putnika prevede na simetričan, po cenu dupliranja broja čvorova [10]. Međutim, to ne rešava problem pretpostavke o postojanju svih grana u grafu.[4]

Zanimljivo je da postoji algoritam (čija efikasnost se može porediti sa već postojećim rešenjima) za rešanje UHC problema koji je implementiran pomoću SAT rešavača. Zasniva se na svođenju UHC problema na problem pridruživanja koji se potom rešava SAT rešavačem[6].

[4]Da bi se prevazišao problem, granama koje ne postoje u grafu potrebno je pridružiti težine nekoliko redova veličine veće u odnosu na težine pridružene postojećim granama. Međutim, tako dobijene instance Concorde TSP ne može egzaktno da reši, jer ne uspeva da pronađe bazno rešenje dualnog problema.

Glava 6

Svođenje problema

U narednoj glavi će biti izloženo nekoliko polinomijalnih svođenja, jednog NP-kompletnog na drugi.

6.1 Svođenje SAT problema na Klika problem

Algoritam opisan u nastavku izložio je Karp 1972. godine [2, 8]. Da bi se izvršilo svođenje SAT problema na Klika problem, za svako pojavljivanje literala u formuli čija zadovoljivost se ispituje, kreira se čvor grafa, označava se tim literalom i oznakom klauze u kojoj se nalazio. Granama se spajuju svi čvorovi koji ne pripadaju istoj klauzi i čiji literali ne predstavljaju negaciju onog drugog. Dobijeni graf ce sadržati kliku veličine broja klauza početne formule ako i samo ako je početna formula bila zadovoljiva. Da bismo dobili valuaciju koja zadovoljava početnu formulu, dovoljno je pročitati literale čvorova koji se nalaze u kliki veličine k. Sve promenljive koje se ne nalaze u literalima čvorova mogu imati prozvoljnu vrednost.

Teorema 1. *U grafu G dobijenom svođenjem iskazne formule F od k klauza nad n iskaznih promenljivih u KNF postoji k-klika ako i samo ako je početna formula zadovoljiva.*

Dokaz. Pretpostavimo da je v dobro formirana zadovoljavajuća valuacija iskazne formule F. Formula F ima oblik $\bigwedge_{1 \leq i \leq k} C_i$, gde su C_i klauze. Potrebno je pokazati da postoji klika veličine bar k u grafu G. U grafu G koji se dobija svođenjem formule F, svaki čvor je pridružen tačno jednom literalu formule. Svi čvorovi koji predstavljaju literale iz različitih klauza su susedni, osim ukoliko predstavljaju iskaznu promenljivu i njenu negaciju. Kako je v zadovoljavajuća valuacija formule F, ona zadovoljava svaku od klauza $C_i, 1 \leq i \leq k$, što znači da u svakoj klauzi postoji po bar jedan literal kome je dodeljena vrednost 1. Među tim literalima se ne mogu javiti iskazna promenljiva i njena negacija, jer je valuacija v dobro formirana, tj. dodeljuje tačno jednu vrednost svakoj iskaznoj promenljivoj. Dakle, k čvorova koji odgovaraju tim literalima su svaki

sa svakim međusobno susedni, jer pripadaju različitim klauzama, a ne predstavljaju iskaznu promenljivu i njenu negaciju. Dakle, u grafu G postoji klika veličine bar k.

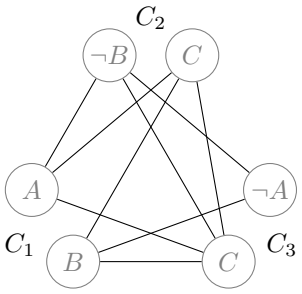
Pretpostavimo da u grafu G dobijenom svođenjem formule F postoji klika veličine k. Tih k čvorova klike mora predstavljati literale iz različitih klauza, jer između bilo koja dva čvora koji predstavljaju literale iste klauze ne postoji grana, pa ne mogu pripadati kliki bilo koje veličine. Ako se literalima koji su pridruženi čvorovima klike dodeli istinitosna vrednost 1, onda oni predstavljaju dobro formiranu valuaciju, jer se među njima ne mogu naći iskazna promenljiva i njena negacija. Kako nijedan par literala ne pripada istoj klauzi, zaključujemo da moraju pripadati k različitih klauza, pa svaka klauza formule sadrži literal koji ima vrednost 1, te je dobijena valuacija zadovoljavajuća za formulu F. Dakle, iskazna formula F je zadovoljiva.

□

Primer 4. *Razmotrimo opisano svođenje na primeru formule:*

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg A \vee C).$$

Na slici 5.1 se vidi kako izgleda graf dobijen svođenjem. Može se uočiti da on ima 3 klike veličine 3 i da svaka od njih odgovara bar jednoj valuaciji. Klika A,C,C daje dve zadovoljavajuće valuacije početne formule: A, B, C i $A, \neg B, C$.



Slika 6.1: Graf dobijen svođenjem polazne formule

6.2 Svođenje problema Klika na SAT problem

U nastavku će biti opisano svođenje čiji su autori Iwama i Miyazaki [3]. Neka je dat graf $G = (V, E)$, gde je $V = v_1, v_2, ...v_n$. Cilj je konstruisati formulu u KNF takvu da ona bude zadovoljiva akko u datom grafu postoji klika veličine k.

Neka su čvorovi grafa označeni brojevima od 0 do $n - 1$. Za binarni zapis indeksa svakog od čvorova potrebno je $N = \log n$ bita, odnosno sa N binarnih promenljivih može se predstaviti jedan čvor grafa. Za predstavljanje klike

veličine k potrebno je k čvorova, dakle $k \cdot N$ promenljivih (organizovanih npr. kao dvodimenziona matrica $X = [x_{i,j}]$, $i = 0, 1, ..., k-1$, $j = 0, 1, ..., N-1$). Sa X_i se označava niz od N binarnih promenljivih koje sadrže oznaku i-tog čvora klike, dok se sa $[X_i]$ predstavlja vrednost smeštena u niz X_i. Cilj svođenja je konstrukcija takve formule da se posle rešavanja u tih $k \cdot N$ promenljivih nalaze oznake k čvorova klike. Pošto neke od vrednosti koje se mogu zapisati sa N bita mogu biti veće od broja čvorova u grafu, neophodno je konstruisati klauze čija će vrednost biti 0 ukoliko je vrednost smeštena u N promenljivih veća od n (tj. $n-1$ jer će SAT rešavač vratiti vrednosti u intervalu od 0 do $n-1$). Kako se traži klika veličine k, onda je potrebno obezbediti da su sve vrednosti smeštene u vektore koje identifikuju čvorove različite (inače se dobija klika manja od k). Još je potrebno sprečiti da se u kliki (dakle u bilo kom paru vektora promenljivih) javi par identifikatora čvorova koji nisu susedni u grafu.
Svođenje čine sledeći koraci:

1. Za svako $i = 0, 1, ..., k-1$, konstruisati klauze čija vrednost postaje 0 ukoliko je $[X_i] > n-1$.

2. Za svako i,j td. $0 \leq i < j \leq k-1$, konstruisati klauze čija vrednost postaje 0 ukoliko je $[X_i] = [X_j]$.

3. Za svaka dva nesusedna čvora v_p, v_q, $0 \leq p < q \leq n-1$, i za svaki par i, j, $0 \leq i < j \leq k-1$ konstruisati klauze čija vrednost postaje 0 ukoliko važi $[X_i] = p \wedge [X_j] = q$, odnosno $[X_i] = q \wedge [X_j] = p$

Bitan element svođenja su klauze koje nameću gore opisana ograničenja. Neka $C = \bigvee_{0 \leq i \leq N-1} l_i$ je proizvoljna klauza od N literala . Klauza C se evaluira kao 0, ako svaki literal l_i, $(0 \leq i \leq N-1)$(u odnosu na valuaciju v) ima vrednost 0. Za izražavanje uslova algoritma, potrebne se klauze koje se evaluiraju kao 0 za neku određenu vrednost t predstavljenu nizom promenljivih $x_0, x_1, \ldots, x_{N-1}$ (radi jednostavnosti izostavlja se oznaka vrste). Neka je sa $t[i]$, $(0 \leq i \leq N-1)$ označena vrednost i-tog bita u binarnoj reprezentaciji vrednosti t, gde pozicija 0 označava bit najveće težine, a pozicija $N-1$ bit najmanje težine. Potrebno je za svakoj promenljivoj pridružiti literal tako klauza koju čine ti literali ima željena svojstva. Pridruživanje treba obaviti na sledeći način:

$$l_i = \begin{cases} x_i, & \text{ako} \quad t[i] = 0 \\ \neg x_i, & \text{ako} \quad t[i] = 1 \end{cases}, \; 0 \leq i \leq N-1.$$

Klauza označena sa $C([x_1, x_2, \ldots, x_{N-1}] = t)$ i koju čine literali l_i kojima je vrednost pridružena na opisani način, evaluira se kao nula ako niz promenljivih $x_0, x_1, \ldots, x_{N-1}$ predstavlja binarni zapis vrednosti t. Kada se zna konstrukcija ovih klauza sve ostale klauze koje se koriste u svođenju se mogu lako dobiti. Ograničenja koraka 1 nad nizom promenljivih $X_i = x_{i,0}, x_{i,1}, \ldots, x_{i,N-1}$, $(0 \leq i \leq k-1)$ mogu opisati nizom klauza (koje se vezuju u KNF). Za svaku vrednost t, takvu da je $n \leq t \leq 2^N - 1$, formuli se doda klauza $C([X_i] = t)$. Ograničenja iz koraka 2 nad dva niza promenljivih X_i i X_j, se generišu za svaku vrednost

t, $(0 \leq t \leq n-1)$. Formuli se dodaje klauza oblika:

$$(C([X_i] = t) \vee C([X_j] = t).$$

Ograničenja koraka 3 se generišu za oznake nesusednih čvorova p i q, nad dva niza promenljivih X_i i X_j. Formuli se dodaju ograničenja oblika:

$$(C([X_i] = p) \vee C([X_j] = q)).$$

Teorema 2. *Algoritam svođenja problema klike na SAT problem je korektan.*

Dokaz. Cilj je dokazati da je dobijena formula zadovoljiva ako i samo ako polazni graf sadrži k-kliku.

Ako pretpostavimo da u grafu postoji k-klika, to znači da postoji k različitih oznaka čvorova koje se mogu smestiti u vektore X_i, $(0 \leq i \leq n-1)$ pri tome zadovoljavajući sva ograničenja dobijene formule. Tih k oznaka bi predstavljale postojeće čvorove i bile bi međusobno različite. Kako one čine kliku ne bi bile podložne uslovima generisanim u koraku 3, dakle, dobijena formula bi bila zadovoljiva.

Ako pretpostavimo da u grafu ne postoji k-klika, najviše $k-1$ različitih oznaka čvorova se može smestiti u vektore X_i, $(0 \leq i \leq n-1)$ tako da ne narušavaju uslove formule. Ali k-ta vrednost bi morala da predstavlja ili nepostojeći čvor (narušavajući uslove iz koraka 1) ili neki od $k-1$ već odabranih čvorova (narušavajući uslove iz koraka 2) ili neki čvor koji sa preostalih $k-1$ čvorova ne čini k-kliku, što znači da bar sa jednim od $k-1$ čvorova nije susedan, pa postoji ograničenje dodato u koraku 3 koje je narušeno. Dakle, formula dobijena svođenjem bi bila nezadovoljiva.

Ako pretpostavimo da je dobijena formula zadovoljiva, to znači da postoji zadovoljavajuća valuacija koja predstavlja k oznaka postojećih čvorova grafa (prema ograničenjima iz koraka 1). Te oznake su sve različite (prema ograničenjima iz koraka 2) i svaki par oznaka predstavlja susedne čvorove (prema ograničenjima iz koraka 3). Dakle, tih k oznaka predstavljaju k-kliku polaznog grafa.

Ako pretpostavimo da formula nije zadovoljiva, znači da ne postoji k različitih oznaka koje predstavljaju postojeće čvorove grafa koji su svi međusobno susedni. Dakle, graf ne sadrži k-kliku.

Dakle, dobijena formula je zadovoljiva ako i samo ako polazni graf sadrži k-kliku.

□

U nastavku su izloženi neki detalji vezani za implementaciju pojedinačnih koraka.

Za implementaciju koraka 1 potreban je algoritam za konstrusanje klauza čija vrednost postaje 0 ukoliko je vrednost niza $\lceil \log_2 n \rceil$ promenljivih veća ili jednaka od n, koji je opisan u nastavku. Ukoliko je $n = 2^k$, za neko k, onda je $\lceil \log_2 n \rceil = \log_2 n$ i nema potrebe za ograničavajućim klauzama, jer su sve

vrednosti koje se mogu zapisati dozvoljene. Ideja je da se prođe kroz binaran zapis vrednosti n od bita najveće težine ka bitu najmanje težine i da se formuli dodaju klauze koje će se evaluirati kao nula za što veći raspon vrednosti većih ili jednakih od n. Kretanjem kroz binaran zapis fiksirane vrednosti n (počevši od najznačajnijeg bita i trenutne klauze, koja je na početku prazna), za razmatrani bit se dodaje novi literal - negirana promenljiva koja odgovara bitu koji se razmatra. Ukoliko je bit koji se razmatra jednak 1, prelazi se na sledeći najznačajniji bit. Ukoliko je bit koji se razmatra 0, zatvara se trenutna klauza i započinje nova i to tako što za sve već posmatrane bitove (uključujući i trenutni) dodaje literal i to ako je vrednost bita 1 dodaje se negirana promenljiva koja odgovara tom bitu, inače se dodaje promenljiva koja odgovara tom bitu (bez negiranja). Kada se obrade svi već razmatrani bitovi tek onda se prelazi na sledeći najznačajniji bit. Kada se obradi bit najmanje značajnosti zatvara se trenutna klauza.

Primer 5. *Neka je dat niz promenljivih $x_3x_2x_1x_0$, cilj je da formula bude nezadovoljiva ukoliko one predstavljaju vrednost veću ili jednaku* 9*. Algoritam daje sledeću formulu u KNF:*

$(\neg x_3 \vee \neg x_2)$ — *evaluira se kao nula za sve vrednosti veće od 11*

$(\neg x_3 \vee x_2 \vee x_1)$ — *evaluira se kao nula za vrednosti 10 i 11*

$(\neg x_3 \vee x_2 \vee x_3 \vee \neg x_0)$ —*evaluira se kao nula za vrednost 9*

Postoji više načina da se konstruišu klauze koje će učiniti formulu nezadovoljivom ukoliko dva niza promenljivih (dužine N) sadrže iste vrednosti. Jednostavan pristup je da se za svaku od mogućih vrednosti $0, 1, \ldots, 2^N - 1$ i za svaki par pozicija u kliki napravi klauza čija vrednost postaje 0 ukoliko oba niza promenljivih imaju baš tu vrednost. Ovo rešenje je eksponencijalne vremenske složenosti u odnosu na broj promenljivih u jednoj vrsti, ali na nivou celog algoritma svođenja vremenska složenost je $O(k^2 \cdot n)$. Može se primetiti da klauze iz koraka 1 dozvoljavaju samo n vrednosti (umesto $2^{\lceil \log_2 n \rceil}$). Broj generisanih klauza se može smanjiti na n ignorisanjem već zabranjenih vrednosti, tako da je ovaj korak svođenja polinomijalan[1]. Drugi način na koji bi se ovo moglo izvesti je upotrebom *Cajtin kodiranja* (eng. *Tseitin encoding*) koje se upotrebljava da se izbegne eksponencijalna eksplozija prilikom svođenja formula u KNF[14]. U nastavku će biti detaljnije razmotren ovaj pristup. Neka su data dva niza promenljivih $x_{N-1}, ..., x_1, x_0$ i $y_{N-1}, ..., y_1, y_0$. Potrebno je da vrednost klauze bude 0, ukoliko nizovi promenljivih sadrže iste vrednosti. Odnosno, potrebna je logička funkcija čiji je izlaz 0, ako odgovarajući parovi promenljivih (x_i, y_i), $(0 \leq i \leq N-1)$ sadrže iste vrednosti. Ako se razmatraju dva niza

[1]Složenost ovog koraka algoritma je $O(k^2 \cdot n)$ i bez tog zapažanja, jer je $O(k^2 \cdot 2^{\lceil \log_2 n \rceil})$ u najgorem slučaju $O(k^2 \cdot 2^{1+\log_2 n}) = O(2 \cdot k^2 \cdot 2^{\log_2 n})$, što je posle zanemarivanja multiplikativne konstante $O(k^2 \cdot n)$. Kako je $k \leq n$, $O(k^2 \cdot n) = O(n^3)$, pa je ovaj korak je polinomijalane vremenske složenosti u odnosu na veličinu ulaza svođenja

dužine 1, cilj je eksluzivna disjunkcija dva bita (odnosno XOR funkcija), koja se može predstaviti pomoću veznika konjunkcije i disjunkcije na sledeći način: $x_i \, xor \, y_i = (x_i \wedge \neg y_i) \vee (\neg x_i \wedge y_i)$.

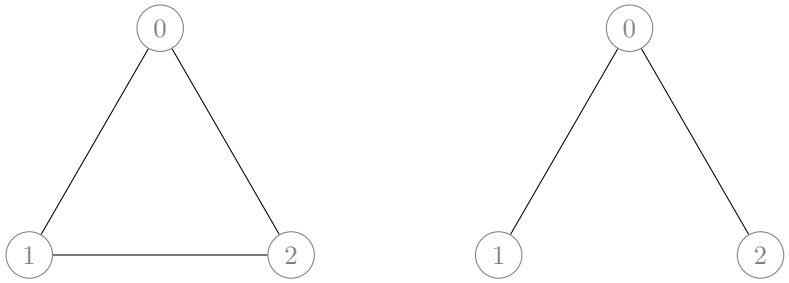
Ovde je problem u tome što formula treba da bude u KNF tako da svi $\wedge$ veznici moraju da budu "iznad" $\vee$ veznika. Rešenje je sledeće: ako se uvede nova promenljiva z sa sledećim ograničenjima: $(\neg z \vee x) \wedge (\neg z \vee y) \wedge (z \vee \neg x \vee \neg y)$ tada nova promenljiva z ima istu vrednost kao $x \wedge y$, a uvedena ograničenja su u KNF i mogu se jednostavno dodati u formulu. Formula $(\neg z \vee x) \wedge (\neg z \vee y) \wedge (z \vee \neg x \vee \neg y)$ je u stvari KNF formule $z \Leftrightarrow x \wedge y$.

U ovom pristupu, je dakle, za opisivanje XOR funkcije nad nizom bitova dužine N potrebno $2 \cdot N$ novih promenljivih i $3 \cdot 2 \cdot N + 1$ klauza. Sa $3 \cdot 2 \cdot N$ klauza se dodeljuju vrednosti novouvedenim promenljivim, još je na kraju potrebno napraviti jednu klauzu u kojoj se javljaju sve novouvedene promenljive, čime ona efektivno služi kao indikator da li su vektori koji se porede isti ili različiti .

Kako se u algoritmu ovaj korak izvršava za svaki par vrsta promenljivih, tj. $\frac{k \cdot (k-1)}{2}$ puta, onda je na nivou celog svođenja potrebno dodatnih $k \cdot (k-1) \cdot \lceil \log n \rceil$ promenljivih. Svođenje koje koristi jednostavnu verziju ovog koraka, generiše formulu nad $k \cdot \lceil \log n \rceil$ promenljivih, tako da modifikovano svođenje Cajtinovim kodiranjem daje formulu nad $k^2 \cdot \lceil \log n \rceil$ promenljivih. Nemodifikovano svođenje u drugom koraku generiše $O(k^2 \cdot n)$ klauza, dok se u istom koraku modifikovanog svođenju generiše $O(k^2 \cdot \log n)$ klauza. Za dovoljno velike instance moglo bi se uštedeti na broju klauza, po cenu dodatnih promenljivih.

Primer 6. *Razmatra se svođenje na primeru dva grafa veličine* 3 *u kojima se traži klika veličine* 3*. Jedan graf sadrži, a drugi ne sadrži 3-kliku, tako da bi se uočile sličnosti i razlike u formulama dobijenim svođenjem. Broj čvorova je* $n = 3$*, a broj promenljivih potreban za predstavljanje jednog čvora* $N = \lceil \log_2 n \rceil = 2$*. Kako se traži 3-klika, formula će imati* $k \cdot N$ *promenljivih, što je u ovom slučaju* 6*.*

Neka su te promenljive označene sa $x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32}$*, a sa* $X_i = (x_{i1}, x_{i2})$, $(i = 1, 2, 3)$. $X_3 = \overline{x_{31} x_{32}}$



Slika 6.2: Graf koji sadrži 3-kliku (levo) i graf koji ne sadrži 3-kliku (desno)

Prvo se generišu klauze koje će obezbediti da vrednosti smeštene u X_i, $(i = 1, 2, 3)$ *ne budu veće od* $n - 1 = 2$*. Dobijaju se sledeće tri klauze:* $(\neg x_{11} \vee \neg x_{12})$, $(\neg x_{21} \vee \neg x_{22})$ *i* $(\neg x_{31} \vee \neg x_{32})$*. Može se primetiti da svaka od klauza ima vrednost*

nula ukoliko je $[X_i] = 3$ *(za odgovarajuće* i*). Dalje se generišu klauze koje će obezbediti da je reč o 3 različita čvora:*

$(x_{11} \vee x_{12} \vee x_{21} \vee x_{22})$ — *dobija vrednost 0 kada su* $[X_1]$ *i* $[X_2]$ *jednake 0*

$(x_{11} \vee \neg x_{12} \vee x_{21} \vee \neg x_{22})$ — *dobija vrednost 0 kada su* $[X_1]$ *i* $[X_2]$ *jednake 1*

$(\neg x_{11} \vee x_{12} \vee \neg x_{21} \vee x_{22})$ — *dobija vrednost 0 kada su* $[X_1]$ *i* $[X_2]$ *jednake 2*

Slične klauze se generišu za ostale parove vektora: X_1, X_3 *i* X_2, X_3*. Može se primetiti da nije potrebno generisati klauze ukoliko vektori sadrže vrednost 3, jer nije moguće da se ona javi u bilo kom vektoru vrednosti* X_i*,* $(i = 1, 2, 3)$ *usled klauza generisanih u prvom koraku algoritma. U trećem koraku algoritma nastaju razlike u formulama dobijenim svođenjem grafova na slikama. Levi graf sa slike nema nesusednih čvorova, pa stoga neće imati dodatnih klauza u formuli. Za desni graf će biti generisane klauze sledećeg oblika:*

$(x_{11} \vee \neg x_{12} \vee \neg x_{21} \vee x_{22})$ — *dobija vrednost 0 kada je* $[X_1] = 1$ *i* $[X_2] = 2$

$(\neg x_{11} \vee x_{12} \vee x_{21} \vee \neg x_{22})$ — *dobija vrednost 0 kada je* $[X_1] = 2$ *i* $[X_2] = 1$

Dakle, prethodne dve klauze obezbeduju da je formula nezadovoljiva ako se nesusedni čvorovi 1 *i* 2 *jave u kliki na pozicijama* 1 *i* 2*. Klauze istog oblika se generišu i za preostale kombinacije pozicija u kliki:* $1, 3$ *i* $2, 3$*.*

Primer 7. *Klauze koje su generisane u drugom koraku algoritma svođenja u primeru 6 se mogu zameniti klauzama koje koriste Tseitin kodiranje.*

Definišu se sledeće promenljive: $l_{i,j,1} \Leftrightarrow x_{i,1} \wedge \neg x_{j,1}$*,* $d_{i,j,1} \Leftrightarrow \neg x_{i,1} \wedge x_{j,1}$*,* $l_{i,j,2} \Leftrightarrow x_{i,2} \wedge \neg x_{j,2}$*,* $d_{i,j,2} \Leftrightarrow \neg x_{i,2} \wedge x_{j,2}$*,*$(0 \leq i, j \leq k-1)$*. Sada se uslov* $x_{i,t}$ *xor* $x_{j,t}$ *može zapisati kao* $l_{i,j,t} \vee d_{i,j,t}$*, za* $1 \leq t \leq 2$*, a ceo uslov* $[X_i] \neq [X_j]$ *se može predstaviti kao* $l_{i,j,1} \vee d_{i,j,1} \vee l_{i,j,2} \vee d_{i,j,2}$*. Potrebno je još dodati klauze koje će definisanim promenljivama dodeliti baš vrednosti odgovarajućih konjunkcija. Dakle, skup klauza koji bi kodirao uslov* $[X_i] \neq [X_j]$*,* $(0 \leq i < j \leq 2)$ *je:*

$(\neg l_{i,j,1} \vee x_{i,1})$

$(\neg l_{i,j,1} \vee \neg x_{j,1})$

$(l_{i,j,1} \vee \neg x_{i,1} \vee x_{j,1})$ — *ove 3 klauze ograničavaju vrednost* $l_{i,j,1}$ *na* $(x_{1,1} \wedge \neg x_{2,1})$

$(\neg d_{i,j,1} \vee \neg x_{i,1})$

$(\neg d_{i,j,1} \vee x_{j,1})$

$(d_{i,j,1} \vee x_{i,1} \vee \neg x_{j,1})$ — *ove 3 klauze ograničavaju vrednost* $d_{i,j,1}$ *na* $(\neg x_{1,1} \wedge x_{2,1})$

$(\neg l_{i,j,2} \vee x_{i,2})$

$(\neg l_{i,j,2} \vee \neg x_{j,2})$

$(l_{i,j,2} \vee \neg x_{i,2} \vee x_{j,2})$ — *ove 3 klauze ograničavaju vrednost* $l_{i,j,2}$ *na* $(x_{i,1} \wedge \neg x_{j,1})$

$(\neg d_{i,j,2} \vee \neg x_{i,2})$

$(\neg d_{i,j,2} \vee x_{j,2})$

$(d_{i,j,2} \vee x_{i,2} \vee \neg x_{j,2})$ — *ove 3 klauze ograničavaju vrednost* $d_{i,j,2}$ *na* $(\neg x_{i,2} \wedge x_{j,2})$

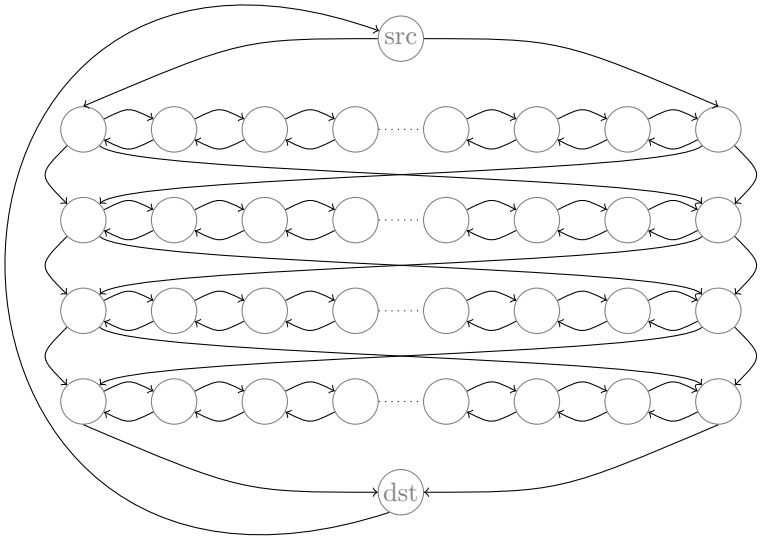
$(l_{i,j,1} \vee 2_{i,j,1} \vee l_{i,j,2} \vee d_{i,j,2})$ — *klauza ima vrednost* $(x_{i,1}\, xor\, x_{j,1}) \vee (x_{i,2}\, xor\, x_{j,2})$

Pri čemu, $l_{i,j,1}$, $l_{i,j,2}$, $d_{i,j,1}$, $d_{i,j,2}$, $(0 \leq i < j \leq 2)$ *predstavljaju novouvedene promenljive i njih bi u kompletnom primeru bilo* $\frac{k\cdot(k-1)}{2} \cdot 2 \cdot N = 12$. *Ovaj način kodiranja uslova koristi* $\frac{k\cdot(k-1)}{2} \cdot (2 \cdot 3 \cdot N + 1) = 37$ *klauza, dok jednostavan način koristi* $\frac{n\cdot k\cdot(k-1)}{2} = 9$ *klauza, tako da na ovako malom primeru nema svrhe koristiti Cajtin kodiranje. Međutim, u zavisnosti od veličine klike koja se traži Cajtin kodiranje može dati manji broj klauza, po cenu većeg broja promenljivih.*

6.3 Svođenje SAT problema na UHC problem

U ovom delu biće izložen algoritam svođenja na osnovu pristupa u knjizi "Algorithm Design"[11]. Ulaz algoritma je iskazna formula u KNF sa n promenljivih i m klauza, a izlaz je usmereni graf u kome će se tražiti Hamiltonov ciklus. Najpre je potrebno osmisliti grafovsku strukturu koja odgovara svim valuacijama, pa je potom ograničiti na takav način da samo zadovoljavajuće valuacije daju usmeren Hamiltonov ciklus. Svaka promenljiva u iskaznoj formuli može imati dve vrednosti, 1 i 0 (tačno i netačno). Svakoj promenljivoj će biti pridružen niz čvorova, povezanih u oba smera. Ideja je da ako se niz čvorova obilazi sa leva na desno promenljiva koja je predstavljena tim nizom ima dodeljenu vrednost 1, ili ako se obilazi sa desna na levo ima dodeljenu vrednost 0. Za svaku iskaznu promenljivu u formuli konstruiše se takav niz ulančanih čvorova. Kako su vrednosti dodeljene promenljivama međusobno nezavisne, krajevi lanaca se povezuju na sve moguće načine. Dovoljno je da svaki lanac (osim prvog i poslednjeg) budu povezani sa dva druga lanca. Lanac čvorova se povezuje sa narednim dodavanjem grana od krajnjih čvorova do krajnjih čvorova narednog lanca. Dodaju se grane koje povezuju iste krajeve lanaca, levi sa levim i desni sa desnim, ali i suprotne krajeve, levi sa desnim i desni sa levim. Struktura koja je do sada izložena ne može sadržati Hamiltonov ciklus i zato se dodaju jos dva čvora, izvor i odredište. Izvorni čvor se povezuje granama ka krajnjim čvorovima prvog lanca, dok se odredišni čvor povezuje granama od krajeva poslednjeg lanca. Sada u grafu postoji 2^n Hamiltonovih puteva, koji počinju u izvornom čvoru, a završavaju se u odredišnom čvoru. Svaki Hamiltonov put predstavlja jednu moguću valuaciju promenljivih. Da bi se dobili Hamiltonovi ciklusi dodaje se jedna grana od odredišta ka izvoru, čime se dobija grafovska struktura u kojoj svaki Hamiltonov ciklus odgovara jednoj valuaciji. Valuacije su određene smerom kojim Hamiltonov ciklus prolazi kroz svaki od lanaca čvorova koji predstavljaju promenljive.

Kako postoji osnova za graf, potrebno je razmotriti kako se može povezati zadovoljivost početne formule sa egzistencijom Hamiltonovog ciklusa u konstruisanom grafu. Formula u KNF je zadovoljiva ukoliko je svaka od njenih klauza



Slika 6.3: Grafovska struktura koja sadrži 2^n Hamiltonovih ciklusa

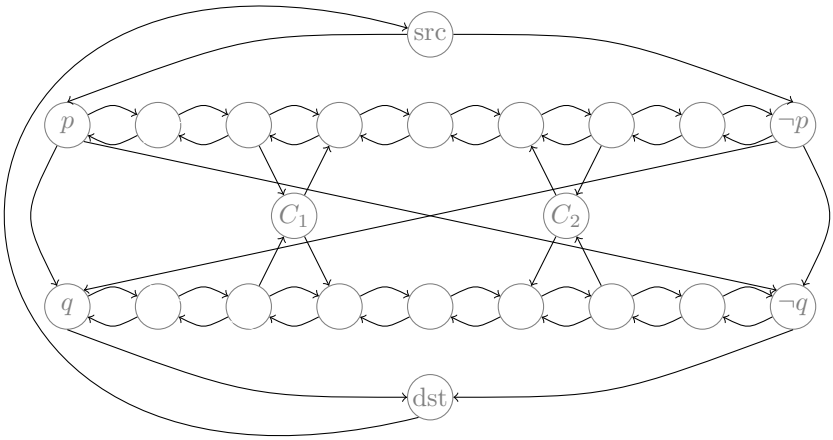
tačna u nekoj valuaciji, a klauza je tačna u nekoj valuaciji ako sadrži bar jedan literal čija je istinitosna vrednost tačno. Svakoj klauzi biće pridružen novi čvor u grafu. Ukoliko se čvorovi koji predstavljaju klauze ne povežu sa ostatkom grafovske strukture u grafu ne postoji Hamiltonov ciklus. Čvorove koji predstavljaju klauze će biti povezani sa lancima koji predstavljaju promenljive, ali samo sa onim lancima koji predstavljaju promenljive koje ta klauza sadrži. Povezivanje tih čvorova sa lancima koji predstavljaju promenljive se vrši na sledeći način: ako klauza sadrži promenljivu koja se u njoj javlja bez negacije, onda se ona ulančava u smeru koji odgovara dodeli vrednosti 1 toj promenljivoj ("u smeru tačno"), a ukoliko se promenljiva u klauzi javlja negirana, onda se ulančava u smeru koji odgovara dodeli vrednosti 0 toj promenljivoj ("u smeru netačno"). Da bi se lanac čvorova koji predstavlja promenljivu mogao povezati sa svakim čvorom koji predstavlja klauzu, potrebno je da se lanac sastoji od bar $m + 1$ čvorova, ne računajući krajnje čvorove koji povezuju lance. Međutim to nije dovoljno dobro, jer se može desiti da ciklus dođe do čvora koji predstavlja klauzu iz jednog lanca, a da nastavi kroz drugi lanac i da se dobije Hamiltonov ciklus. Da bi se to izbeglo, potrebno je da između svaka dva para čvorova koji treba da se povežu sa čvorom koji predstavlja klauzu postoji bar jedan čvor koji koji je povezan samo sa susednim čvorovima u tom lancu. Tako da će se svaki lanac sastojati od $3 \cdot m + 3$ čvora. Promenljivoj je dodeljena vrednost 1, ukoliko se lanac (ako je predstavljen u polozaju kao na slici 6.3) obilazi sa levog kraja ka desnom, a vrednost 0 ukoliko se obilazi sa desnog kraja ka levom. U skladu sa time, posmatrajući sa leva na desno čvorovi unutar lanca se označavaju brojevima $1, 2, \ldots, 3 \cdot m + 3$. Čvor koji predstavlja i-tu klauzu se može povezati samo sa čvorovima označenim brojevima $3 \cdot i$ i $3 \cdot i + 1$. Ukoliko se u i-toj klauzi

nalazi literal koji predstavlja iskazno slovo, u odgovarajućem lancu se dodaju usmerene grane od čvora $3\cdot$ do čvora koji predstavlja i-tu klauzu i od čvora koji predstavlja i-tu klauzu do čvora $3 \cdot i + 1$. Ukoliko se u i-toj klauzi nalazi literal koji predstavlja negirano iskazno slovo, u odgovarajućem lancu dodaju se usmerene grane od čvora $3 \cdot i + 1$ do čvora koji predstavlja i-tu klauzu i od čvora koji predstavlja i-tu klauzu do čvora $3 \cdot i$. U ovako dobijenom grafu postoji usmeren Hamiltonov ciklus ako i samo ako je polazna formula zadovoljiva. Zadovoljavajuća valuacija se može dobiti utvrđivanjem smera obilaska svakog od lanaca čvorova.

Teorema 3. *U grafu G dobijenom svođenjem početne formule F postoji Hamiltonov ciklus h ako i samo ako je formula F zadovoljiva*

Dokaz. Pretpostavimo da postoji zadovoljavajuća valuacija v početne formule F. Onda postoji ciklus h kroz graf G koji za svako iskazno slovo q formule F prolazi kroz lanac P_q (koji je pridruzen izkaznom slovu q). Lanac P_q se obilazi u smeru sa leva na desno ako promenljiva q ima vrednost 1 u valuaciji v, a u smeru sa desna na levo ukoliko promenljiva ima vrednost 0 u valuaciji v. Za svaku klauzu C_i koja je zadovoljena valuacijom v, postoji bar jedan lanac čvorova P_r iz kog postoji grana do čvora koja predstavlja klauzu C_i u pravcu kojim ciklus h prolazi kroz lanac P_r i postoji grana kojom ciklus h može nastaviti kretanje duž lanca P_r u ispravnom pravcu (ne preskočivši pri tome nijedan čvor lanca P_r). Kako je u valuaciji v svaka klauza formule F zadovoljena, to znači da ciklus h prolazi kroz sve čvorove koje predstavljaju klauze, a samim tim i sve čvorove grafa (jer je očigledno iz konstrukcije da svaka valuacija definise ciklus koji prolazi tačno jednom kroz sve čvorove koji ne predstavljaju klauze u grafu G). Dakle, može se zaključiti da je h Hamiltonov ciklus grafa G.

Pretpostavimo da postoji Hamiltonov ciklus h u grafu G. Ukoliko Hamiltonov ciklus h posećuje čvor koji predstavlja klauzu C_i prilikom obilaska lanca P_q u smeru sa leva na desno, onda čvor koji predstavlja klauzu C_i mora biti posećen granom iz čvora $3 \cdot i$ u lancu P_q i mora biti napušten granom ka čvoru $3 \cdot i + 1$ u lancu P_q. U suprotnom, čvor $3 \cdot i + 2$ u lancu P_q ima samo jednog neposećenog suseda, pa ciklus h ne može posetiti čvor $3 \cdot i + 2$, a da pritom ostane Hamiltonov ciklus. Slično, ukoliko Hamiltonov ciklus h posećuje čvor koji predstavlja klauzu C_i prilikom obilaska lanca P_q u smeru sa desna na levo, onda čvor koji predstavlja klauzu C_i mora biti posećen granom iz čvora $3 \cdot i + 1$ u lancu P_q i mora biti napušten granom ka čvoru $3 \cdot i$ u lancu P_q. U suprotnom, čvor $3 \cdot i - 1$ u lancu P_q ima samo jednog neposećenog suseda, pa ciklus h ne može posetiti čvor $3 \cdot i - 1$, a da pritom ostane Hamiltonov ciklus. Kako svaki čvor koji predstavlja klauzu u grafu G i koji se nalazi u Hamiltonovom ciklusu h, mora biti susedan paru međusobno susednih čvora na nekom lancu P_r, pa se Hamiltonov ciklus h može modifikovati na sledeći način. Neka je svaki čvor koji predstavlja klauzu C_i, $(1 \leq i \leq m)$, dosegnut u ciklusu h granom iz čvora l_i na nekom putu P_{q_i} i napušten granom ka čvoru k_i na putu P_{q_i}. Hamiltonov ciklus h se može skratiti, zamenom čvora koji predstavlja klauzu C_i, $(1 \leq i \leq m)$ i njemu susednih grana u ciklusu, usmerenom granom (l_i, k_i). Za svako $1 \leq i \leq m$, ta grana mora pos-



Slika 6.4: Graf koji sadrži samo dva Hamiltonova ciklusa

tojati u grafu G i pripadati putu P_{q_i}. Ovom transformacijom se dobija ciklus h' koji prolazi kroz osnovu grafa G, koja odgovara svim mogućim valuacijama n promenljivih. Utvrđivanjem smera obilaska svakog od puteva P_i, $1 \leq i \leq n$ ciklusom h' dobija se zadovoljavajuća valuacija formule F. □

Primer 8. *Na slici 5.4 se vidi graf dobijen svođenjem sledeće iskazne formule u KNF:* $(p \vee q) \wedge (\neg p \vee \neg q)$. *Sa* C_1 *označen je čvor koji predstavlja klauzu* $(p \vee q)$, *a sa* C_2 *čvor koji predstavlja klauzu* $(\neg p \vee \neg q)$. *Literali kojima su označeni čvorovi na krajevima lanaca (koji predstavljaju promenljive) predstavljaju literal koji dobija vrednost 1 ukoliko se u lanac ulazi preko tog čvora. U dobijenom grafu postoje samo dva Hamiltonova ciklusa:* $< src, p, \ldots, C_1, \ldots, \neg p, \neg q, \ldots, C_2, \ldots, q, dst, src >$ *i* $< src, \neg p, \ldots, C_2, \ldots, p, q, \ldots, C_1, \ldots, \neg q, dst, src >$

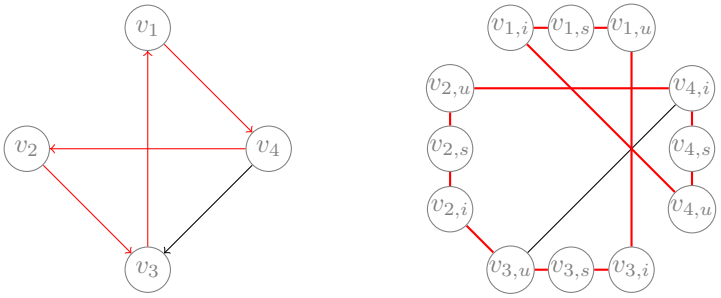
6.4 Svođenje UHC problema na NHC problem

Kako svi dostupni rešavači za problem Hamiltonovog ciklusa zapravo rešavaju neusmerenu varijantu problema, da bi se sprovelo rešavanje SAT problema svođenjem na problem Hamiltonovog ciklusa, neophodno je izvršiti svođenje UHC problema na NHC problem.

Ulaz algoritma je usmereni graf $G = (V, E)$, a izlaz je neusmereni graf $G' = (V', E')$ koji sadrži neusmereni Hamiltonov ciklus akko polazni graf sadrži usmereni Hamiltonov ciklus.

Svakom čvoru v polaznog grafa pridružuju se tri čvora u novom grafu, v_u, v_s, v_i, koji se nazivaju ulazni, središnji i izlazni. Središnji čvor v_s se povezuje samo sa svojim ulaznim i izlaznim čvorovima v_u i v_i. Ideja je da se sve usmerene grane koje izlaze iz čvora v preslikaju u neusmerene grane incidentne čvoru v_i, a sve usmerene grane koje ulaze u čvor v preslikaju u neusmerene grane incidentne čvoru v_u. Ovako konstruisan graf će sadržati Hamiltonov ciklus akko je polazni graf sadrži usmeren Hamiltonov ciklus. Središnji čvor u svakoj trojci služi da obezbedi poštovanje smera, odnosno da do svakog čvora ciklus mora doći usmerenom granom koja se u njemu završava, a nastaviti usmerenom granom koja u njemu počinje.

Primer 9. *Na narednoj slici crvenom bojom je označen Hamiltonov ciklus, u usmerenom grafu sa leve strane i konstruisanom neusmerenom grafu sa desne.*



Slika 6.5: Graf koji sadrži usmeren Hamiltonov ciklus (levo) i neusmereni graf dobijen njegovim svođenjem (desno)

6.5 Svođenje NHC problema na SAT problem

Algoritam opisan u nastavku su izložili Iwama i Miyazaki 1994. godine [3]. Ulaz algoritma je graf $G = (V, E)$, sa n čvorova, označenih brojevima od $0, 1, \ldots, n-1$, a izlaz je iskazna formula u KNF koja je zadovoljiva ako i samo ako početni graf sadrži Hamiltonov ciklus.

Hamiltonov ciklus predstavlja niz susednih čvorova grafa, takvih da se nijedan ne ponovi i svaki javi bar jednom. Kao kod svođenja problema k-klike, potrebno je $\lceil \log n \rceil$ binarnih promenljivih za predstavljanje jednog čvora, a za predstavljanje celog ciklusa $n \cdot \lceil \log n \rceil$. Potrebno je obezbediti da svaka vrednost zapisana u vektoru binarnih promenljivih koje predstavljaju oznaku čvora bude postojeća u grafu i da se nijedna oznaka čvora ne ponovi duž ciklusa. Dodatno, potrebno je obezbediti da ukoliko između dva čvora ne postoji grana onda se oni ne mogu nalaziti na susednim pozicijama u ciklusu (gde se smatra sa su prva i poslednja pozicija u ovoj reprezentaciji takođe susedne). Dakle, koristi se dvodimenzioni niz binarnih promenljivih, $X = [x_{i,j}]$, $i = 0, 1, ..., n-1$, $j = 0, 1, ..., N-1$), gde je $N = \lceil \log n \rceil$. U nastavku, sa X_i

označava se niz promenljivih koje sadrže identifikator i-tog čvora ciklusa, a sa $[X_i]$ vrednost koja je smeštena u nizu promenljivih X_i, za $i = 0, 1, ..., n-1$. Tehnike konstrukcija klauza su iste kao kod svođenja problema k-klike na SAT problem, pa im se ovde neće posevećivati dodatna pažnja.

Algoritam čine sledeći koraci:

1. Za svako $i = 0, 1, ..., n-1$ konstruisati klauze koje su nezadovoljive ukoliko je $[X_i] \geq n$
2. Za svaki par pozicija u ciklusu $0 \leq i < j \leq n-1$ konstruisati klauze koje su nezadovoljive ukoliko je $[X_i] = [X_j]$
3. Za svaki par čvorova $v_k, v_l \in V : (v_k, v_l) \notin E$ i za svaki par susednih pozicija i, j u ciklusu konstruisati klauze koje nisu zadovoljive ukoliko $[X_i] = k \wedge [X_j] = l$. Ukoliko je reč o problemu neusmerenog Hamiltonovog ciklusa onda se uz već navedene klauze dodaju i klauze koje nisu zadovoljive ukoliko $[X_i] = l \wedge [X_j] = k$.

Algoritam se može modifikovati kako bi se uštedelo na promenljivima potrebnim za zapis jednog čvora, tako što bi se odabrao neki čvor da bude poslednji u ciklusu (što ne škodi je svaki se mora naći u ciklusu), ali po cenu dodavanja skupa klauza koje obezbeđuju povezanost $[X_0]$ i $[X_{n-2}]$ čvora sa fiksiranim čvorom.

Teorema 4. *Algoritam svođenja problema Hamiltonovog ciklusa na SAT problem je korektan.*

Dokaz. Cilj je dokazati da je dobijena formula zadovoljiva ako i samo ako polazni graf sadrži Hamiltonov ciklus.

Ako pretpostavimo da u grafu postoji Hamiltonov ciklus, to znači da postoji k različitih oznaka čvorova koje se mogu smestiti u vektore X_i, $(0 \leq i \leq n-1)$ pri tome zadovoljavajući sva ograničenja dobijene formule. Tih k oznaka bi predstavljale postojeće čvorove i bile bi međusobno različite. Kako one čine Hamiltonov ciklus ne bi bile podložne uslovima generisanim u koraku 3, dakle, dobijena formula bi bila zadovoljiva.

Ako pretpostavimo da je dobijena formula zadovoljiva, to znači da postoji zadovoljavajuća valuacija koja predstavlja k oznaka postojećih čvorova grafa (prema ograničenjima iz koraka 1). Te oznake su sve različite (prema ograničenjima iz koraka 2) i svaki susedni par oznaka predstavlja susedne čvorove (prema ograničenjima iz koraka 3). Dakle, tih k oznaka predstavljaju Hamiltonov ciklus.

Dakle, dobijena formula je zadovoljiva ako i samo ako polazni graf sadrži Hamiltonov ciklus.

□

Primer 10. *Neka je dat graf kakav je prikazan na slici 6.6. Graf čini* $n = 4$ *čvora, za predstavljanje njihovih oznaka biće potrebno* $N = 2$ *promenljive.*

Promenljive ćemo organizovati u matricu $X = (x_{i,j}), (1 \leq i \leq 4, 1 \leq j \leq 2)$. *Koristiće se oznaka* X_i *za niz promenljivih* $(x_{i,1}, x_{i,2})$, *sa* $[X_i]$ *će biti označena vrednost smeštena u nizu* X_i *Može se primetiti se u koraku 1 algoritma neće generisati klauza koje ograničavaju vrednosti smeštene u nizove* X_i, *jer su potrebne sve vrednosti koje se mogu predstaviti sa 2 bita. U drugom koraku algoritma, za svaki par nizova* $X_i, X_j, (1 \leq i < j \leq 4)$ *se generišu klauze koje se evaluiraju kao 0, ukoliko je* $[X_i] = [X_j]$. *Na primer, za nizove* X_1 *i* X_2 *rezultujućoj formuli se dodaju sledeće klauze:*

$(x_{11} \vee x_{12} \vee x_{21} \vee x_{22})$ — *dobija vrednost 0 kada su* $[X_1]$ *i* $[X_2]$ *jednake 0*

$(x_{11} \vee \neg x_{12} \vee x_{21} \vee \neg x_{22})$ — *dobija vrednost 0 kada su* $[X_1]$ *i* $[X_2]$ *jednake 1*

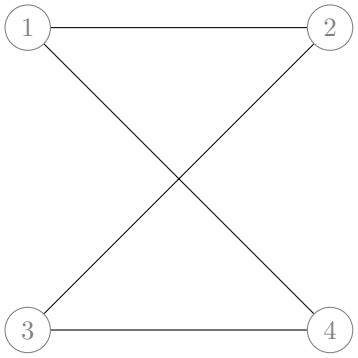
$(\neg x_{11} \vee x_{12} \vee \neg x_{21} \vee x_{22})$ — *dobija vrednost 0 kada su* $[X_1]$ *i* $[X_2]$ *jednake 2*

$(\neg x_{11} \vee \neg x_{12} \vee \neg x_{21} \vee \neg x_{22})$ — *dobija vrednost 0 kada su* $[X_1]$ *i* $[X_2]$ *jednake 3*

U trećem koraku algoritma, za parove nesusednih čvorova $(1, 3)$ *i* $(2, 4)$ *i za parove vektora koji predstavljaju susedne pozicije u ciklusu* $(X_1, X_2), (X_2, X_3), (X_3, X_4), (X_4, X_1)$, *generišu se klauze kao u trećem koraku algoritma svođenja problema klika na SAT problem. Konkretno za par čvorova* $(1, 3)$, *i za par susednih pozicija ciklusa* (X_4, X_1) *formuli se dodaju sledeće klauze:*

$(x_{41} \vee \neg x_{42} \vee \neg x_{11} \vee \neg x_{12})$ — *dobija vrednost 0 kada je* $[X_4] = 1$ *i* $[X_1] = 3$

$(\neg x_{41} \vee \neg x_{42} \vee x_{11} \vee \neg x_{12})$ — *dobija vrednost 0 kada je* $[X_4] = 3$ *i* $[X_1] = 1$



Slika 6.6: Graf u kome se traži Hamiltonov ciklus

Glava 7

Fazna promena

U nastojanju da se bolje razume NP-kompletnost i srž težine NP-kompetnih problema, eksperimentalno je istraživana težina različitih instanci. Njihovi rezultati su doveli do zapažanja da se težina instanci NP-kompetnih problema može proceniti na osnovu nekih parametara. Takav parametar zavisi od vrste problema. Na primer, za grafovske probleme je često srodan gustini grafa ili prosečnom stepenu čvorova grafa. Ispostavilo se da za sve NP-kompletne probleme postoji vrednost parametra koja razdvaja dve "faze" (eng. *phase*) problema. Na primer, logično je očekivati da bi u jako retkom grafu (gde je broj grana reda veličine broja čvorova) pronalazak k-klike ili Hamiltonovog ciklusa bio malo verovatan, dok vrlo gusti (skoro kompletni) grafovi gotovo sigurno sadrže k-kliku odnosno Hamiltonov ciklus. Fazna promena (eng. *phase transition*) predstavlja promenu specifične vrednosti parametra tako da za vrednosti pre promene veći deo instanci pripada jednoj "fazi", a za vrednosti posle promene veći deo instanci pripada drugoj "fazi".

Eksperimenti koji ispituju faznu promenu za razne vrednosti parametra obično se zasnivaju na generisanju slučajnih instanci, koje se zatim rešavaju i beleži se odgovor rešavača (DA ili NE) i vreme izvršavanja. U zavisnosti od algoritma koji je upotrebljen za rešavanje može se osmisliti neka mera koja treba da oslikava složenost rešavanja.

U nastavku će biti razmotrena fazna promena za probleme SAT, Hamiltonov ciklus i k-klike.

7.1 Fazna promena SAT problema

Prilikom analiziranja fazne promene SAT problema, razmatraju se formule u KNF koje u svakoj klauzi imaju k literala — k-SAT instance. Parametar koji karakteriše složenost k-SAT instanci je odnos broja klauza i broja promenljivih. Klauze u formuli dodaju ograničenja na moguće kombinacije vrednosti promenljivih. Može se očekivati da je formula sa malo klauza lako zadovoljiva (jer ima malo ograničenja), dok je formula sa puno klauza retko zadovoljiva (jer postoji

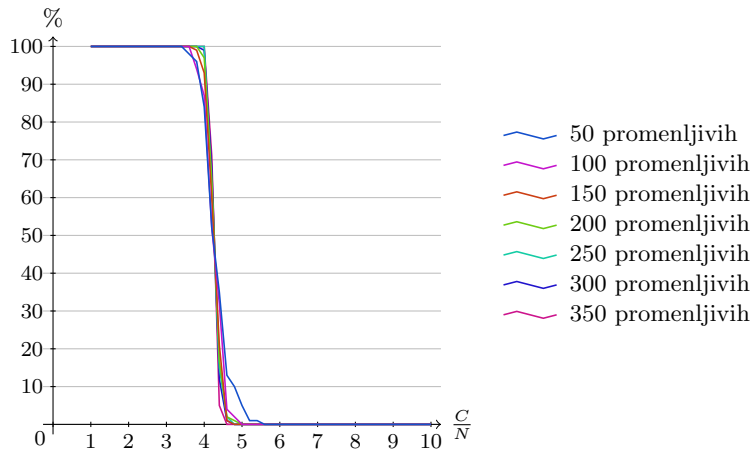
mnogo ograničenja, od kojih bi mnoga bila kontradiktorna). U oba slučaja to se relativno lako otkriva. Neka $M(N, L)$ označava skup iskaznih formula koje se sastoje od L klauza nad skupom od N iskaznih slova, i neka je *sat* funkcija zadovoljivosti skupa formula. Na primer, $sat(M(N, L))$ je procenat zadovoljivih formula u skupu $M(N, L)$. Smatra se da za različite tipove skupova $M(N, L)$ postoji kritična vrednost odnosa $\frac{L}{N} = c_0$, koja se naziva tačkom fazne promene i za koju važi:

$$\lim_{N\to\infty} sat(M(N, [c \cdot N])) = \begin{cases} 100\%, & ako\ je\ c < c_0 \\ 0\%, & ako\ je\ c > c_0 \end{cases}$$

Ako tačka fazne promene postoji, ona je jedinstvena za taj skup formula $M(N, L)$. Za različite vrednosti k različite su i vrednosti za tačku fazne promene k-SAT problema.

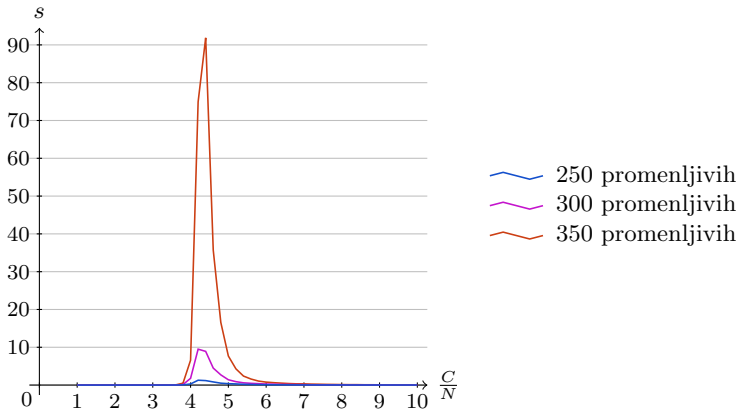
Za SAT problem je karakteristično da rešavanje instanci duž fazne promene postaje sve teže kako se instance približavaju kritičnoj tački a posle prolaska kroz nju težina opet opada.

Za 3-SAT formule eksperimentalno je utvrđeno da je vrednost tačke fazne promene približno 4.25 [15][1]. Na slici 7.1 se vidi tzv. "efekat zaoštravanja" koji je karakterističan za faznu promenu, sa porastom parametra N prelaz između dve "faze" problema je sve oštriji. Na slici 7.2 se vidi prosečno vreme rešavanja formula. Dakle, najteže formule su upravo okolini tačke fazne promene. Na slici 7.2 nisu prikazani rezultati rešavanja formula veličine do 250 promenljivih, jer se u proseku rešavaju za manje od 1 sekunde.



Slika 7.1: Udeo zadovoljivih instanci među 3-SAT formulama

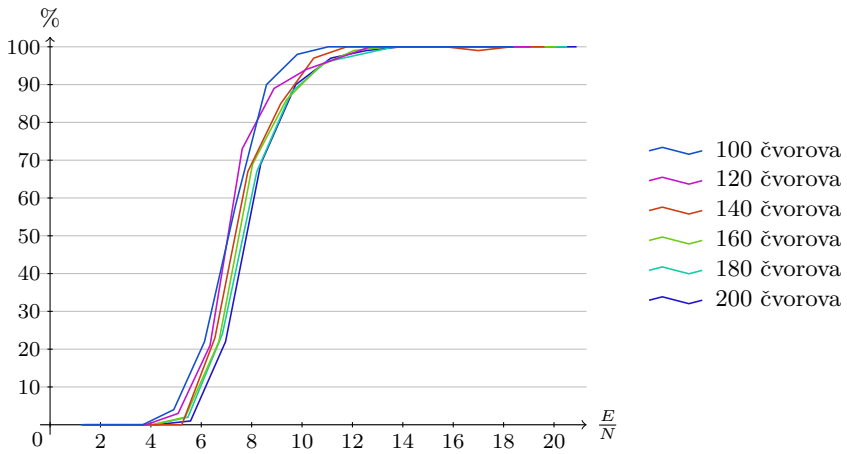
[1]Postoji više načina za slučajno generisanje k-SAT formula. Jedan od načina je slučajni k-SAT model. Ako je dat broj promenljivih N i broj klauza C, onda za svaku od C klauza se bira podskup od k elemenata iz skupa promenljivih koje će se naći u toj klauzi, i svakoj od odabranih promenljivih se dodaje negacija sa verovatnoćom 0.5.



Slika 7.2: Prosečno vreme rešavanja 3-SAT instanci rešavačem Minisat

7.2 Fazna promena NHC problema

Za NHC problem je utvrđeno da je parametar fazne promene prosečan stepen čvorova grafa. Ako je E broj grana, a N broj čvorova u grafu, fazna promena se analizira odnosa $\frac{E}{N \log N}$, kako bi se eliminisao uticaj odnosa parametra fazne promene i veličine grafa. Poznato je da ako graf sadrži čvor stepena manjeg ili jednakog 1, graf ne može sadržati Hamiltonov ciklus. Ako je prosečan stepen čvora mala vrednost, vrlo je verovatno da graf ne sadrži Hamiltonov ciklus. S druge strane, u kompletnom grafu (prosečni stepen čvorova u grafu je $n-1$) svaki prosti ciklus dužine n je Hamiltonov. Može se primetiti da instance koje pripadaju krajevima intervala parametra fazne promene mogu relativno lako rešiti. Čizman i drugi su eksperimentalno potvrdili teorijska očekivanja da do fazne promene kod problema Hamiltonovog ciklusa dolazi kada prosečni stepen čvora u grafu iznosi $\ln N + \ln \ln N$ [12]. Postoji više modela za generisanje slučajnih grafova. Dva najjednostavnija modela za generisanje slučajnih grafova su $G_{n,m}$ i $G_{n,p}$. $G_{n,m}$ model označava odabir m grana u grafu sa n čvorova. Model $G_{n,p}$ označava da svaka grana ima verovatnoću p da se javi u grafu. Fazna promena problema Hamiltonovog ciklusa je karakterisitična po tome što verovatnoća generisanja teške slučajne instance teži nuli kako veličina grafa n raste [17]. Zbog toga merenje složenosti rešavanja prilikom prolaska kroz faznu promenu ne znači mnogo, jer se uglavnom radi o "lakim" instancama. Na slici 7.3 je prikazan udeo instanci koje sadrže Hamiltonov ciklus prilikom prolaska kroz faznu promenu. Može se primetiti da efekat zaoštravanja nije očigledan kao kod SAT problema, jer tačka fazne promene zavisi od veličine instance. Na slici 7.4 prikazano je prosečno vreme rešavanja problema Hamiltonovog ciklusa prilikom prolaska kroz faznu promenu i koji prikazuje da je jako mali udeo teških instanci među velikim instancama problema Hamiltonovog ciklusa (oko 10 na generisanih 9000 hiljada slučajnih instanci). Velike razlike u vrednostima koji se mogu videti na grafikonu su posledica nekolicine instanci čije vreme je drastično

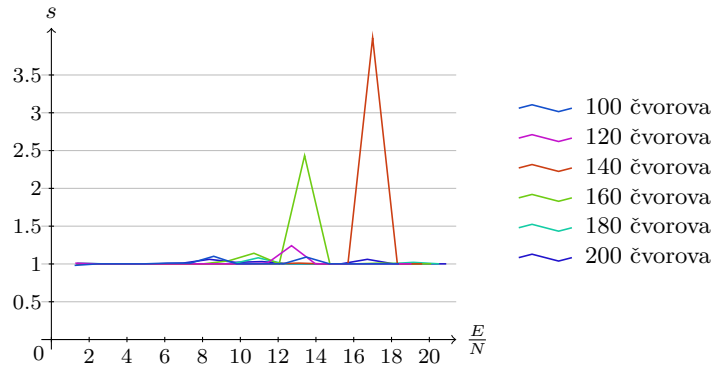


Slika 7.3: Udeo grafova koji sadrže Hamiltonov ciklus u odnosu na prosečni stepen čvora u grafu

odstupalo od prosečnog. Čini se da je reč o jako malom broju teških slučajnih instanci koje su generisane.

7.3 Fazna promena problema Klika

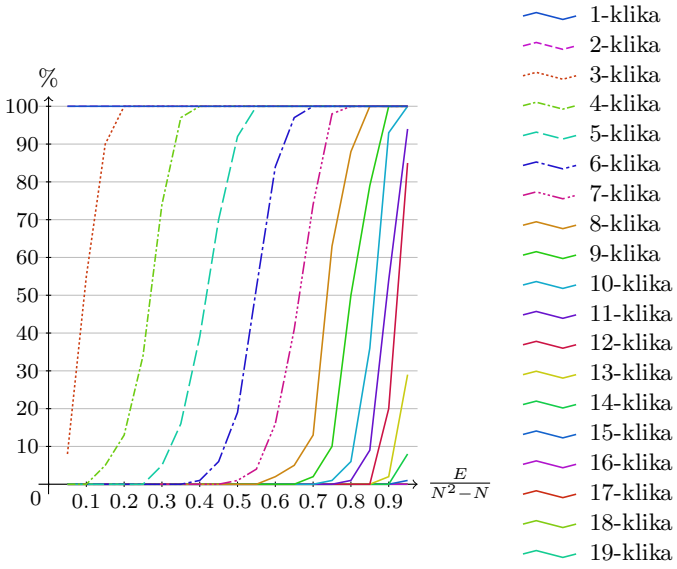
Fazna promena problema klika nije detaljno istražena. Deluje verovatno da ona zavisi od gustine grafa i veličine klike koja se traži. Vrlo je verovatno da se mala klika nađe u grafu (bez obzira da li je on gust ili redak). S druge strane, malo je verovatno da se velika klika pronađe u gustom grafu, a skoro je sigurno da se ne može naći u retkom grafu. Potrebno je videti kako odnos veličine klike i gustine grafa utiče na verovatnoću pronalaska klike u grafu. U nastavku su prikazani rezultati serije eksperimenata, gde je za grafove sa 20, 60, 100 i 120 čvorova prikazan udeo instanci koje sadrže kliku određene veličine. Veličine klika koje su tražene predstavljaju $5\%, 10\%, \ldots, 95\%$ veličine grafa. Klike su tražene u instancama čija gustina predstavlja $5\%, 10\%, \ldots, 95\%$ gustine kompletnog grafa. Na slikama 7.5, 7.6, 7.7 prikazan je udeo klika određene veličine u grafovima različitih gustina. Može se uočiti da i najmanji grafovi ne sadrže klike veće od 75% veličine grafa, a sa porastom veličine grafa veličina klike koja se može pronaći opada. Tokom eksperimenata je mereno vreme potrebno za rešavanje instanci, međutim vremena rešavanja su prilično ujednačena za grafove do 100 čvorova. Tek u grafovima sa 100 čvorova se javljaju neka uočljiva odstupanja. Na slici 7.8 se vidi da se male i velike klike nalaze u relativno kratkom roku, dok je za klike "srednje" veličine (npr. 20 do 50 čvorova) potrebno primetno više vremena. Na slici 7.9 je tabelarno prikazan udeo instanci koje sadrže kliku i prosečno vreme potrebno za njihovo nalaženje u grafovima sa 120 čvorova (isti podaci se nalaze na slikama 7.7 i 7.8). Najduže su rešavane instance koje pri-



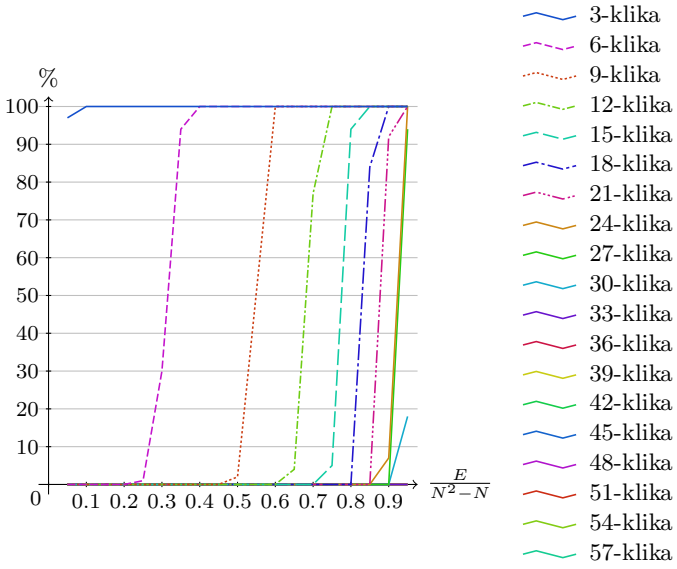
Slika 7.4: Prosečno vreme rešavanja instanci problema Hamiltonovog ciklusa u regionu fazne promene rešavačem HamCycles

padaju kategorijama koje su na faznom prelazu ili u njegovoj okolini. Visoko vreme rešavanja ne prati svuda fazni prelaz. Moguće objašnjenje je da u predelu koji je bio "teži" za rešavanje klike koje su tražene su bile dovoljno velike da je njihovo postojanje u grafu prilično verovatno, ali nedovoljno male da bi se lako pronašle. Na primer, klike do veličine 18 su dovoljno male da se na njih naleti prilično rano u toku pretrage. S druge strane, klike veće od 66 su toliko velike, da se rano u toku pretrage ustanovi uslovom odsecanja da one ne postoje u grafu. Klike između 18 i 66 nisu ni pronađene brzo ni odsečene brzo, što bi moglo objasniti uočeno ponašanje.

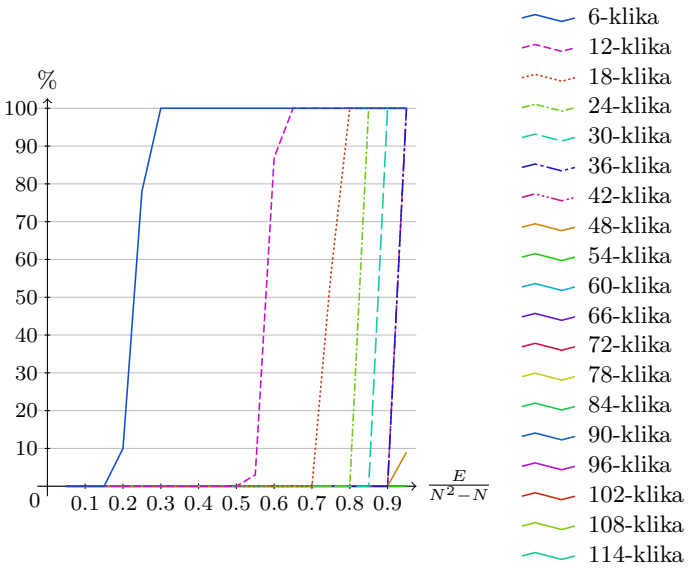
Zanimljivo je zapažanje da ako se zna veličina maksimalne klike u grafu, onda je jasno određena tačka fazne promene u odnosu na veličinu klike za taj graf. Ukoliko bi se mogao izraziti odnos veličine maksimalne klike u zavisnosti od gustine grafa i njegove veličine prišlo bi se korak bliže opsivanju fazne promene u proizvoljnom grafu. Na slici 7.10 prikazana je veličina maksimalne klike kao procenat veličine grafa u grafovima različitih gustina. Može se primetiti da se krive ponašaju slično, što sugeriše da bi zakonitost zaista mogla postojati.



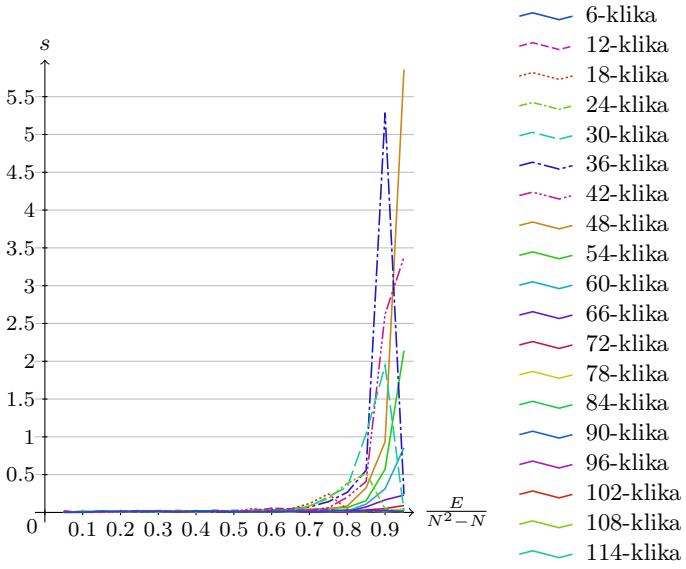
Slika 7.5: Udeo instanci koje sadrže k-kliku među grafovima sa 20 čvorova različite gustine



Slika 7.6: Udeo instanci koje sadrže k-kliku među grafovima sa 60 čvorova različite gustine



Slika 7.7: Udeo instanci koje sadrže k-kliku među grafovima sa 120 čvorova različite gustine



Slika 7.8: Prosečno vreme rešavanja problema k-klike rešavačem HamCycles među grafovima sa 120 čvorova različite gustine

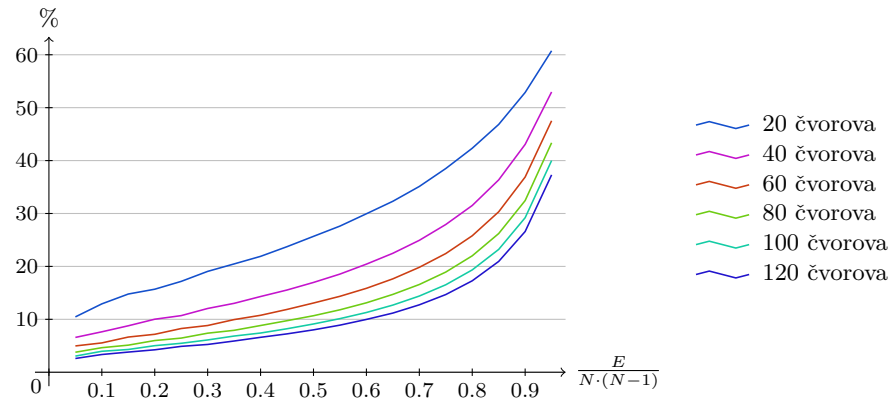
k

$\frac{E}{N^2-N}$	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114
0.95	100	100	100	100	100	100	100	9	0	0	0	0	0	0	0	0	0	0	0
0.9	100	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.85	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.8	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.75	100	100	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.7	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.65	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.6	100	87	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.55	100	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.5	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.45	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.4	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.35	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.3	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.25	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.2	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

k

$\frac{E}{N^2-N}$	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114
0.95	0.01	0.01	0.02	0.01	0.02	0.24	3.38	5.86	2.14	0.85	0.23	0.09	0.05	0.03	0.02	0.03	0.02	0.01	0.02
0.9	0.03	0.02	0.01	0.05	1.94	5.29	2.62	0.93	0.57	0.31	0.17	0.05	0.01	0.01	0.01	0.02	0.01	0.03	0.01
0.85	0.02	0.02	0.01	0.52	1.04	0.56	0.41	0.33	0.16	0.11	0.07	0.03	0.02	0.02	0.02	0.02	0.02	0.01	0.02
0.8	0.03	0.02	0.04	0.39	0.34	0.27	0.20	0.08	0.07	0.02	0.01	0.01	0.03	0.04	0.02	0.01	0.02	0.01	0.02
0.75	0.02	0.04	0.25	0.18	0.21	0.14	0.06	0.04	0.05	0.01	0.01	0.01	0.03	0.01	0.01	0.02	0.02	0.02	0.01
0.7	0.01	0.02	0.12	0.09	0.08	0.07	0.03	0.04	0.03	0.03	0.01	0.03	0.02	0.02	0.01	0.01	0.02	0.02	0.03
0.65	0.02	0.05	0.05	0.04	0.03	0.04	0.03	0.03	0.02	0.01	0.02	0.02	0.01	0.02	0.02	0.01	0.01	0.03	0.01
0.6	0.01	0.06	0.02	0.05	0.01	0.02	0.04	0.01	0.03	0.02	0.02	0.02	0.02	0.03	0.01	0.01	0.02	0.03	0.03
0.55	0.01	0.02	0.05	0.01	0.02	0.02	0.02	0.01	0.02	0.02	0.01	0.02	0.01	0.02	0.02	0.03	0.03	0.01	0.01
0.5	0.01	0.02	0.01	0.01	0.02	0.01	0.03	0.02	0.02	0.03	0.01	0.02	0.01	0.02	0.01	0.01	0.01	0.01	0.01
0.45	0.02	0.02	0.03	0.02	0.02	0.01	0.01	0.01	0.03	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.01	0.02	0.01
0.4	0.02	0.01	0.01	0.01	0.01	0.02	0.01	0.02	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01	0.02	0.01	0.02
0.35	0.01	0.02	0.02	0.02	0.02	0.02	0.01	0.02	0.03	0.02	0.01	0.02	0.02	0.01	0.01	0.02	0.01	0.00	0.01
0.3	0.01	0.02	0.03	0.01	0.02	0.01	0.01	0.01	0.00	0.01	0.02	0.01	0.00	0.01	0.01	0.01	0.00	0.00	0.00
0.25	0.01	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.01
0.2	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00
0.15	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01
0.1	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.00	0.00
0.05	0.00	0.01	0.01	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00

Slika 7.9: U gornjoj tabeli se može videti udeo instanci koji sadrže klike različitih veličina u grafovima sa 120 čvorova različitih gustina. Može se videti prilično izrazena razlika u udelu instanci koji sadrže kliku. Ova slika sugeriše da fazna promena problema klika ima dva paremetra. U donjoj tabeli su prikazana prosečna vremena rešavanja u sekundama istih instanci čija je udeo zadovoljivosti prikazan u gornjoj tabeli. Može se primetiti da insatnce za koje je bilo potrebno veće vreme rešavanja, se nalaze u okolini faznog prelaza.



Slika 7.10: Veličina maksimalne klike (izražena kao procenat veličine grafa) u odnosu na gustinu grafa za grafove različite veličine

Glava 8

Eksperimentalni rezultati

U ovoj glavi izloženi su rezultati eksperimenata za praćenje ponašanja rešavanja instanci duž fazne promene problema sa odgovarajućim rešavačem i svođenjem na druge probleme.

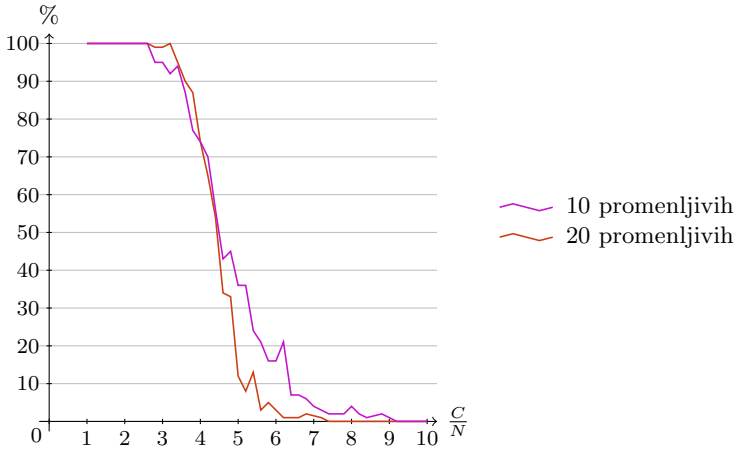
8.1 Rešavanje SAT problema svođenjem na NHC problem

Za potrebe eksperimenta generisane su slučajne 3-SAT instance tako da prolaze kroz region fazne promene. Za svaku vrednost odnosa broja klauza i promenljivih u intervalu od 1 do 10 sa korakom 0.2 generisano je 100 instanci. Svaka od instanci rešavana je prvo SAT rešavačem (Minisat), a potom je izvršeno rešavanje svođenjem na NHC problem (svođem SAT problema na UHC problem, pa svođenjem UHC problema na NHC problem) i upotrebom rešavača HamCycles. Eksperiment je izveden za formule sa 10 i 20 promenljivih.

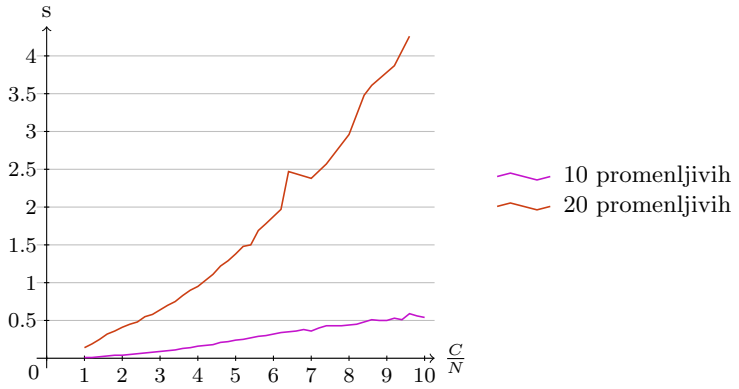
Smatra se da je instanca uspešno rešena ukoliko je za njeno rešavanje bilo potrebno manje od 5 minuta. Posle 5 minuta je prekidano rešavanje, i te instance se smatraju nerešenim. Sve instance SAT rešavač rešava za manje od 0.01 sekunde. Rešavač HamCycles nije uspešno rešio sve instance.
Na slici 8.1 se može videti udeo zadovoljivih instanci među generisanim formulama.

Na slici 8.2 je predstavljeno prosečno vreme potrebno za svođenje instanci, u zavisnosti od odnosa broja klauza i broja promenljivih. Može se videti da vreme raste sa porastom odnosa $\frac{C}{N}$, što je posledica prirode algoritma svođenja, jer za svaku klauzu broj čvorova u grafu svođenja raste za $O(n)$.

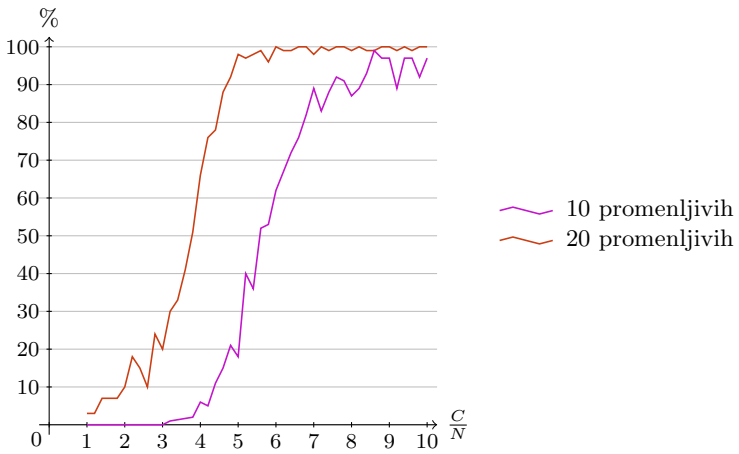
Posle svođenja, rešavač HamCycles ih je rešavao ili za vreme manje od 0.01 sekunde ili mu je trebalo više od predviđenih 5 minuta. Dakle, svođenjem su dobijene ili vrlo lake ili vrlo teške instance za rešavanje. Na slici 8.3 se može videti udeo nerešenih instanci. Rešivost ne zavisi od zadovoljivosti formule. Može se videti kako udeo nerešenih formula postojano raste sa porastom odnosa, a drastično ranije počinje sa rastom pri porastu broja promenljivih.



Slika 8.1: Udeo zadovoljivih instanci među formulama sa 10 i 20 promenljivih



Slika 8.2: Prosečno vreme svođenja SAT instanci sa 10 i 20 promenljivih na NHC problem



Slika 8.3: Udeo instanci koje HamCycles nije rešio u roku od 5 minuta, za formule sa 10 i 20 promenljivih

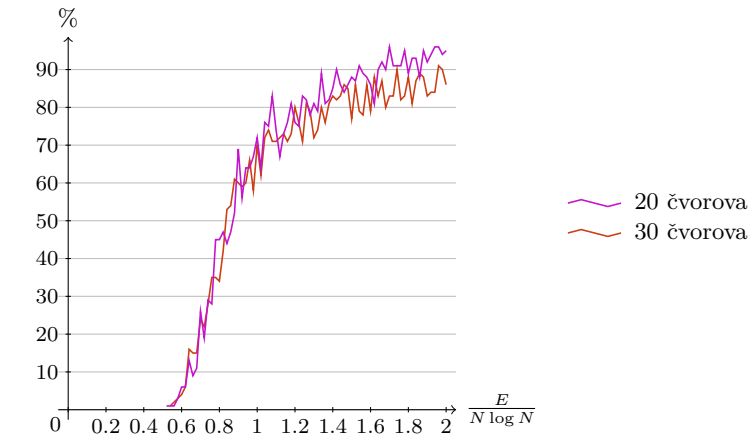
Efekat fazne promene se ne prenosi sa instanci SAT problema na instance NHC problema i čini se da svođenje koje je korišćeno nije dovoljno dobro, pre svega što se koristi kompozicija dva svođenja, SAT problema na UHC problem i UHC problema na NHC problem. Osim toga, samo svođenje SAT problema na UHC problem koristi previše čvorova koji ne služe ničemu. Na primer, ako se u formuli neka promenljiva javlja u svega nekoliko klauza, njen lanac čvorova i dalje sadrži $3 \cdot (C + 1)$ čvorova od kojih većina samo zbunjuje rešavač, a ne kodira nikakvu dodatnu informaciju. Takve situacije bi se mogle izbeći analizom konkretne instance koja se svodi i prema rezulatatima analize prilagoditi dužine lanaca koji predstavljaju promenljive. Takva prilagođavanja ne bi trebala da utiču na korektnost svođenja, a mogla bi poboljšati vreme rešavanja rešavačem za NHC problem.

8.2 Rešavanje NHC problema svođenjem na SAT problem

U narednom delu biće izloženi eksperimentalni rezultati dobijeni prolaskom kroz faznu promenu NHC problema. Parametar fazne promene NHC problema je prosečan stepen čvora u grafu. Međutim, ponekad se za prikazivanje fazne promene koristi i odnos $\frac{E}{N \log N}$. U eksperimentu koji je izveden, navedeni odnos uzima vrednosti iz intervala $[0.4, 2.0]$ sa korakom 0.02. Za svaku vrednost odnosa generisano je 100 slučajnih instanci po modelu $G_{n,m}$ i rešavano direktno rešavačem za problem Hamiltonovog ciklusa i svođenjem na SAT problem. Eksperiment je ponovljen za grafove sa 20 i 30 čvorova. Granica za uspešno rešavanje je 5 minuta. Ukoliko je rešavanje trajalo više od 5 minuta, ono je prekinuto i ta instanca se smatra nerešenom. Sve instance se rešavaju za manje

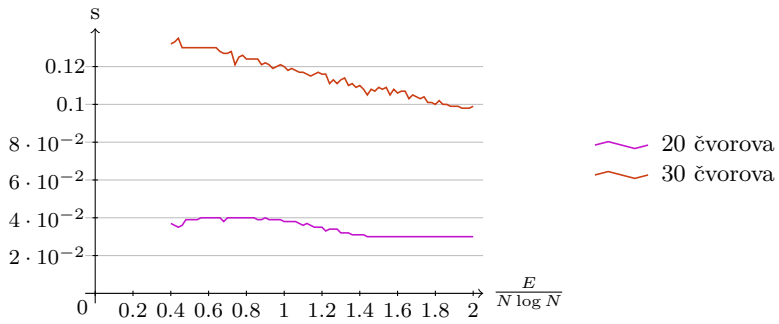
od 0.01 sekunde rešavačem HamCycles.

Na slici 8.4 je prikazan udeo instanci koje sadrže Hamiltonov ciklus prolazeći kroz faznu promenu problema. Udeo je iscrtan duž odnosa $\frac{E}{N \log N}$ za grafove sa 20 i 30 čvorova. Može se videti da je ponašanje slično za obe veličine grafova.



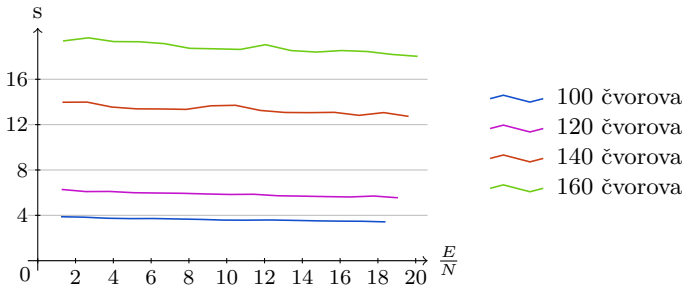
Slika 8.4: Udeo instanci sa Hamiltonovim ciklusom u grafovima sa 20 i 30 čvorova

Na slici 8.5 prikazano je prosečno vreme svođenja instanci problema Hamiltonovog ciklusa sa 20 i 30 čvorova na SAT problem. U oba slučaja vreme svođenja opada, zbog same prirode svođenja. Što je graf gušći to je manje parova čvorova za koje se moraju dodati klauze formuli, koje sprečavaju pojavu tih čvorova kao suseda u ciklusu. U nastavku je prikazano vreme i veličina svođenja za

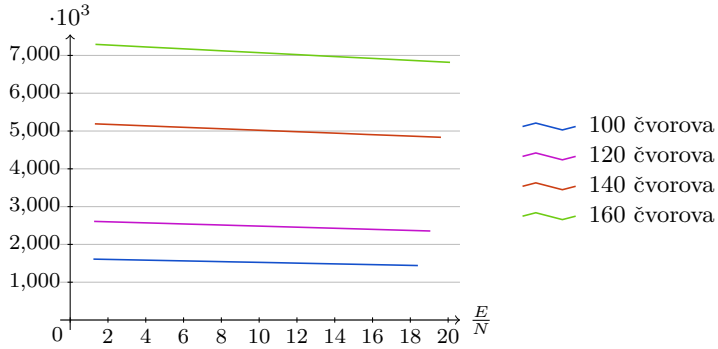


Slika 8.5: Prosečno vreme svođenja instanci NHC problema sa 20 i 30 čvorova

grafove sa 100, 120, 140 i 160 čvorova, ali samo iz regiona fazne promene (ne razmatraju se jako gusti grafovi koji su laki za NHC problem). Na slici 8.6 je prikazano prosečno vreme svođenja za instance iz regiona fazne promene NHC problema. Može primetiti da nedostaje ona zakrivljenost koja se mogla uočiti na



Slika 8.6: Prosečno vreme svođenja velikih instanci NHC problema koje leže u okolini tačke fazne promene na SAT problem

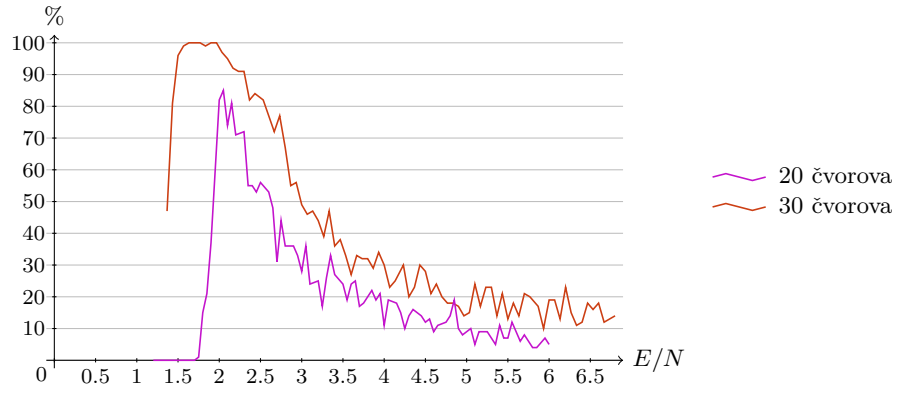


Slika 8.7: Prosečan broj klauza u formulama dobijenim svođenjem velikih instanci koje leže u okolini tačke fazne promene NHC problema na SAT problem

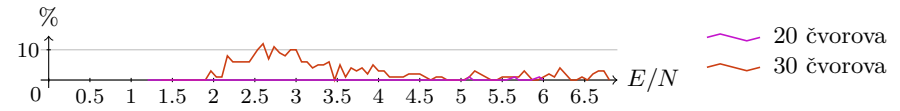
slici 8.5, jer su prikazane instance čija gustina ide do 20% maksimalne gustine grafa. Uprkos tome, može se videti da sa porastom gustine vreme svođenja polako opada. Na slici 8.7 je prikazan broj klauza formula dobijenih svođenjem velikih instanci grafova. Može se primetiti da je formula manja što je graf gušći, međutim razlika u veličini formula deluje zanemarljivo u odnosu na veličine dobijenih formula.

Formule dobijene svođenjem instanci NHC problema sa 20 i 30 čvorova su SAT rešavačem ili rešene za manje od 0.01 sekunde ili nisu rešene u roku od 5 minuta. U nastavku su prikazani grafici koji predstavljaju udeo nerešenih instanci za grafove sa 20 i 30 čvorova. Postoji razlika u rešivosti instanci u zavisnosti od toga da li sadrže ili ne sadrže Hamiltonov ciklus. U eksperimentu sa 20 čvorova veliki deo nerešenih instanci ne sadrži Hamiltonov ciklus. U eksperimentu sa 30 čvorova postoji i značajniji broj instanci koje sadrže Hamiltonov ciklus, ali nisu rešene. Činjenica da veći deo nerešenih instanci ne sadrže Hamiltonov ciklus sugeriše da svođenje ne prenosi ravnomerno težinu problema.

Pošto se čini da ovo svođenje ne prenosi težinu problema ravnomerno, možda



Slika 8.8: Udeo nerešenih instanci rešavačem Minisat u odnosu na prosečan stepen čvora grafa

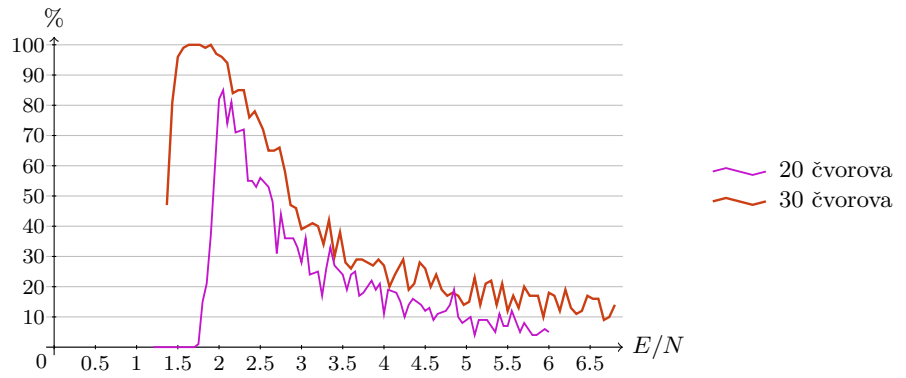


Slika 8.9: Udeo instanci sa Hamiltonovim ciklusom koje nisu rešene rešavačem Minisat u odnosu na prosečan stepen čvora grafa

postoji neko kod koga to ne bi bio slučaj, pa bi samim time dalo i bolje rezultate. Iako ovo svođenje nije praktično upotrebljivo, možda prostor za unapređenje postoji ukoliko bi se osmislilo bolje kodiranje uslova.

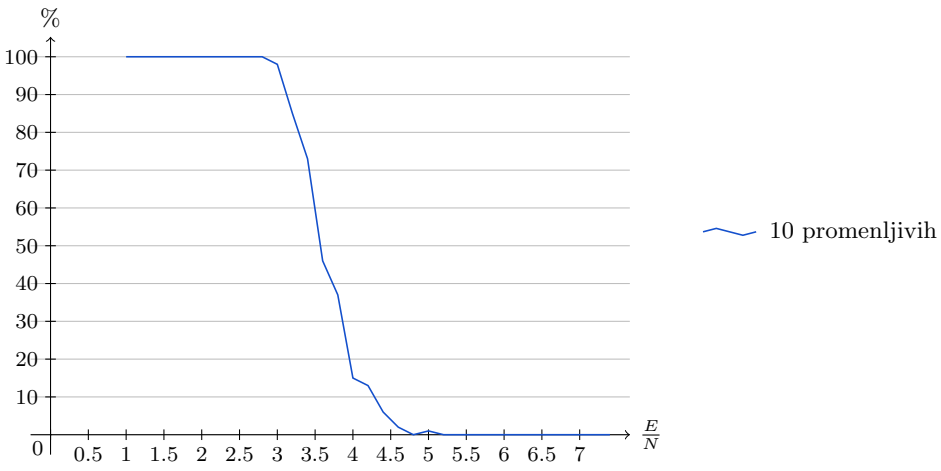
8.3 Rešavanje SAT problema svođenjem na problem klike

U ovom odeljku su prikazani rezultati rešavanja SAT problema svođenjem na problem klike. Generisane su slučajne 3-SAT instance koje prolaze kroz region fazne promene. Znači, odnos broja klauza i broja promenljvih se kreće od vrednosti 1, korakom 0.2. Za svaku vrednost odnosa generisano je po 100 slučajnih 3-SAT instanci. Instance su prvo rešavane SAT rešavačem - Minisat, a zatim je vršeno svođenje i rešavanje cliquer-om. Za razliku od prethodnih eksperimenata gde je izvođeno svođenje pa potom rešavanje nad dobijenim izlazom, u ovom slučaju rezultat svođenja se u programu direktno prosleđuje rešavaču. Ne koriste se datoteke sa opisom instance kako bi se uštedelo na vremenu potrebnom za ulazno—izlazne operacije. Smatra se da je instanca uspešno rešena ukoliko je za njeno rešavanje trebalo manje od 5 minuta. U suprotnom, rešavanje je prekidano i instanca se smatra nerešenom. Minisat je sve instance rešio za manje od 0.01 sekunde.Na slici 8.11 se vidi udeo 3-SAT instanci koje su, nakon



Slika 8.10: Udeo instanci koje ne sadrže Hamiltonov ciklus i nisu rešene rešavačem Minisat u roku od 5 minuta, prikazane u odnosu na prosečan stepen čvora grafa

svođenja na kliku, rešene rešavačem cliquer za manje od 5 minuta. Može se primetiti da kad odnos $\frac{C}{N}$ dostigne vrednost 3 počinju da se javljaju nerešene instance i njihov broj nastavlja da raste. Ovakvi rezultati jasno pokazuju da svođenje koje je korišćeno nije praktično upotrebljivo, jer ne može rešiti ni najmanje instance. Na slici 8.12 se vidi kako vreme svođenja problema zavisi od odnosa broja klauza i broja promenljivih u formuli. Ponašanje vremena svođenja ne iznenađuje, jer se za svaku klauzu dodaju 3 nova čvora grafu i potrebno je razmotriti dodavanje grana ka već postojećim čvorovima. Na slici 8.13 je prikazano prosečno vreme rešavanja 3-SAT instanci sa 10 promenljivih, koje su posle svođenja na problem klike i rešavane cliquer rešavačem. Deluje da samo svođenje verno prenosi složenost problema, jer se kriva ponaša kao kriva rešavanja većih instanci SAT problema koje prolaze kroz faznu promenu. Čak i ukoliko to jeste slučaj, moglo bi se zaključiti da problem klike nije dobar izbor za rešavanje svođenjem, jer i najbolji algoritam za njegovo rešavanje zapravo predstavlja potpunu pretragu sa odsecanjem. Čini se da se prilikom svođenja generišu upravo teške instance za problem klike, jer veličina klike koja se traži spada u kategoriju "premalih da bi se izvršilo odsecanje, a prevelikih da bi se lako pronašle". Kako je veličina grafa dobijenog svođenjem srazmerna broju literala u formuli, možda bi se moglo primetiti bolje ponašanje za k-SAT instance, kod kojih je k veće od 3. da ponašanje algoritma pretrage zavisi od redosleda čvorova grafa, tako da bi možda mogao postojati način da se uvođenjem poretka među čvorovima poboljša vreme rešavanja, možda čak i da se reši problem nerešenih instanci.



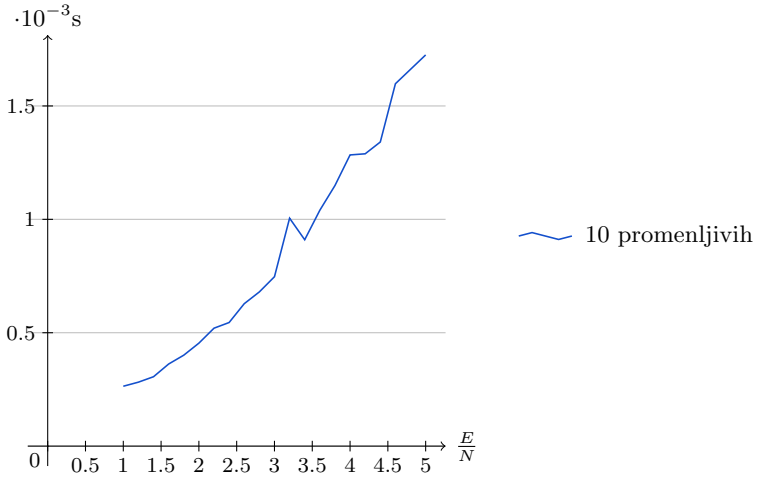
Slika 8.11: Udeo 3-SAT instanci rešenih svođenjem na problem klike

8.4 Rešavanje problema klike svođenjem na SAT problem

U ovom odeljku su prikazani rezultati rešavanja problema klike svođenjem na SAT problem. U do sada prikazanim eksperimentima rešavane su instance koje se nalaze u regionu fazne promene. Kako za problem klike nije formalno opisana tačka fazne promene, eksperiment je izveden za instance koje kombinuju različite vrednosti parametra gustine grafa i veličine klike. Eksperiment je izveden za grafove sa 20 i 30 čvorova. Kako je reč o podacima po dva parametra, gustini grafa i veličini klike, rezultati su prikazani u tabelama. Ćelije tabele su obojene različitim nijansama iste boje, ali tako da oponašaju vrednosti koje se u njima nalaze. Granica uspešnog rešavanja je 5 minuta. Rešavanje je prekidano posle datog roka i u tom slučaju se instance smatraju nerešenim. Na slikama 8.14 i 8.15 je prikazan udeo uspešno rešenih instanci svođenjem na SAT problem. Može se primetiti da su guste instance u kojima se traži klika veća od polovine grafa teške za rešavanje svođenjem na problem SAT. Na osnovu ovoga se može zaključiti da ovo svođenje nije praktično upotrebljivo, ali će uprkos tome biti razmotrena i ostala svojstva.

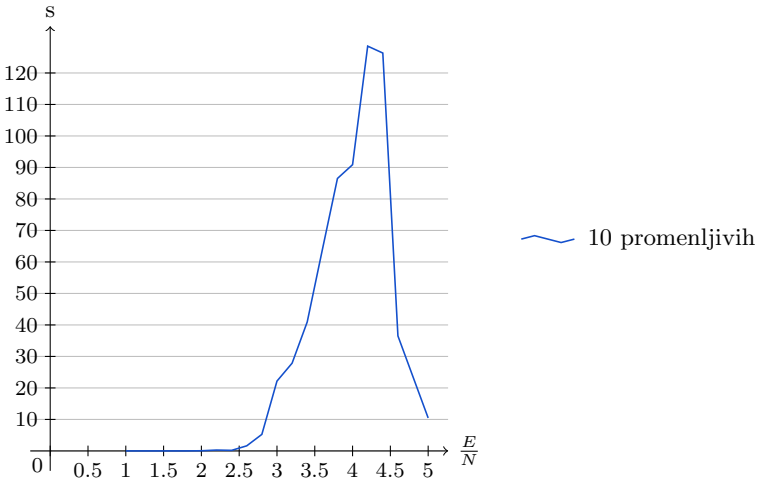
Na slikama 8.16 i 8.17 se može videti prosečno vreme svođenja problema klike na SAT problem. Prikazani rezultati se poklapaju sa očekivanjima, sa porastom veličine klike raste i broj koraka algoritma, a sa rastom gustine grafa opada broj koraka algoritma. Na slikama 8.18 i 8.19 se može videti broj klauza formula dobijenih svođenjem. Sa navedenih slika može se videti da se procenjene veličine instanci i prosečna vremena rešavanja ponašaju slično.

Na slikama 8.20 i 8.21 su prikazana prosečna vremena uspešnih rešavanja problema klike svođenjem na SAT problem za grafove sa 20 i 30 čvorova. Može se uočiti da je većina instanci i dalje "laka" za rešavanje, ali se oblast nerešenih



Slika 8.12: Prosečno vreme svođenja SAT problema na Klika problem

insatnci drastično povećala sa porastom broja čvorova. Očigledno je da ni ovo svođenje nije praktično primenljivo. Kako u problemu klike nije formalno opisana tačka fazne promene, nije jasno da kako se prenosi težina instanci kroz svođenje. Ono što se može zapaziti je da je ovo svođenje problema klike na SAT problem idejno slično svođenju problema Hamiltonovog ciklusa na problem SAT, tako da u suštini ta dva svođenja dele mane.



Slika 8.13: Prosečno vreme rešavanja uspešno rešenih instanci 3-SAT problema svođenjem na problem klike rešavačem Cliquer

$\frac{E}{N^2-N}$ \ k	2	4	6	8	10	12	14	16	18
0.9	100	100	100	100	95	12	0	0	0
0.8	100	100	100	100	71	56	57	27	14
0.7	100	100	100	100	99	98	100	100	100
0.6	100	100	100	100	100	100	100	100	100
0.5	100	100	100	100	100	100	100	100	100
0.4	100	100	100	100	100	100	100	100	100
0.3	100	100	100	100	100	100	100	100	100
0.2	100	100	100	100	100	100	100	100	100
0.1	100	100	100	100	100	100	100	100	100

Slika 8.14: Udeo instanci među grafovima sa 20 čvorova koji je rešen svođenjem na SAT u roku od 5 minuta prikazan u odnosu na veličinu klike i gustinu grafa.

$\frac{E}{N^2-N}$ \ k	3	6	9	12	15	18	21	24	27
0.9	100	100	100	98	22	0	0	0	0
0.8	100	100	100	6	0	0	0	0	0
0.7	100	100	62	0	0	0	0	0	0
0.6	100	100	76	46	26	26	6	0	0
0.5	100	100	100	100	100	100	94	96	98
0.4	100	100	100	100	100	100	100	100	100
0.3	100	100	100	100	100	100	100	100	100
0.2	100	100	100	100	100	100	100	100	100
0.1	100	100	100	100	100	100	100	100	100

Slika 8.15: Udeo instanci među grafovima sa 30 čvorova koji je rešen svođenjem na SAT u roku od 5 minuta prikazan u odnosu na veličinu klike i gustinu grafa.

$\frac{E}{N^2-N}$ \ k	2	4	6	8	10	12	14	16	18
0.9	0.00	0.00	0.00	0.01	0.01	0.02	0.03	0.03	0.04
0.8	0.00	0.00	0.00	0.01	0.02	0.03	0.04	0.05	0.06
0.7	0.00	0.00	0.00	0.01	0.02	0.03	0.05	0.07	0.09
0.6	0.00	0.00	0.01	0.02	0.03	0.04	0.06	0.08	0.10
0.5	0.00	0.00	0.01	0.02	0.03	0.05	0.07	0.09	0.12
0.4	0.00	0.00	0.01	0.02	0.04	0.06	0.09	0.12	0.15
0.3	0.00	0.00	0.01	0.03	0.05	0.07	0.10	0.13	0.16
0.2	0.00	0.00	0.01	0.03	0.05	0.08	0.11	0.15	0.19
0.1	0.00	0.00	0.02	0.03	0.06	0.09	0.12	0.16	0.20

Slika 8.16: Vreme svođenja instanci grafova sa 20 čvorova izraženo u sekundama, a prikazano u odnosu na veličinu klike i gustinu grafa.

$\frac{E}{N^2-N}$ \ k	3	6	9	12	15	18	21	24	27
0.9	0.00	0.00	0.02	0.03	0.05	0.08	0.11	0.15	0.19
0.8	0.00	0.01	0.03	0.06	0.09	0.13	0.18	0.23	0.30
0.7	0.00	0.01	0.04	0.07	0.12	0.17	0.24	0.31	0.37
0.6	0.00	0.02	0.05	0.10	0.15	0.22	0.30	0.36	0.43
0.5	0.00	0.02	0.06	0.12	0.19	0.27	0.34	0.41	0.47
0.4	0.00	0.03	0.07	0.13	0.21	0.28	0.35	0.41	0.49
0.3	0.00	0.03	0.08	0.14	0.21	0.26	0.36	0.42	0.51
0.2	0.00	0.04	0.08	0.17	0.25	0.31	0.40	0.47	0.55
0.1	0.01	0.04	0.11	0.20	0.30	0.34	0.44	0.53	0.62

Slika 8.17: Vreme svođenja instanci grafova sa 30 čvorova izraženo u sekundama, a prikazano u odnosu na veličinu klike i gustinu grafa.

$\frac{E}{N^2-N}$ \ k	2	4	6	8	10	12	14	16	18
0.9	106	596	1470	2728	4370	6396	8806	11600	14778
0.8	182	1052	2610	4856	7790	11412	15722	20720	26406
0.7	258	1508	3750	6984	11210	16428	22638	29840	38034
0.6	334	1964	4890	9112	14630	21444	29554	38960	49662
0.5	410	2420	6030	11240	18050	26460	36470	48080	61290
0.4	486	2876	7170	13368	21470	31476	43386	57200	72918
0.3	562	3332	8310	15496	24890	36492	50302	66320	84546
0.2	638	3788	9450	17624	28310	41508	57218	75440	96174
0.1	714	4244	10590	19752	31730	46524	64134	84560	107802

Slika 8.18: Broj klauza u formulama dobijenih svođenjem grafova sa 20 čvorova.

$\frac{E}{N^2-N}$ \ k	3	6	9	12	15	18	21	24	27
0.9	618	3072	7362	13488	21450	31248	42882	56352	71658
0.8	1140	5682	13626	24972	39720	57870	79422	104376	132732
0.7	1662	8292	19890	36456	57990	84492	115962	152400	193806
0.6	2184	10902	26154	47940	76260	111114	152502	200424	254880
0.5	2706	13512	32418	59424	94530	137736	189042	248448	315954
0.4	3228	16122	38682	70908	112800	164358	225582	296472	377028
0.3	3750	18732	44946	82392	131070	190980	262122	344496	438102
0.2	4272	21342	51210	93876	149340	217602	298662	392520	499176
0.1	4794	23952	57474	105360	167610	244224	335202	440544	560250

Slika 8.19: Broj klauza u formulama dobijenih svođenjem grafova sa 30 čvorova.

$\frac{E}{N^2-N}$ \ k	2	4	6	8	10	12	14	16	18
0.9	1.00	1.00	1.00	1.00	4.91	16.35			
0.8	1.00	1.00	1.00	3.41	135.19	148.51	136.37	131.74	182.14
0.7	1.00	1.00	1.00	7.53	14.50	24.81	25.63	34.59	46.53
0.6	1.00	1.00	1.00	2.27	3.21	4.04	3.82	4.87	4.62
0.5	1.00	1.00	1.00	1.09	1.09	1.23	1.24	1.18	1.19
0.4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Slika 8.20: Prosečno vreme rešavanja svođenjem uspešno rešenih grafova sa 20 čvorova. Vreme je izraženo u sekundama i prikazano u odnosu na veličinu klike i gustinu grafa. Bela polja predstavljaju nepoznato vreme rešavanja.

$\frac{E}{N^2-N}$ \ k	3	6	9	12	15	18	21	24	27
0.9	1.00	1.00	1.00	2.39	9.56				
0.8	1.00	1.00	2.52	1.67					
0.7	1.00	1.00	15.27						
0.6	1.00	1.00	124.83	153.23	213.03	209.11	208.55		
0.5	1.00	1.30	23.18	24.28	39.58	73.62	77.47	88.93	84.15
0.4	1.00	1.96	3.44	4.47	4.54	6.15	5.81	7.03	7.57
0.3	1.00	1.04	1.20	1.20	1.22	1.36	1.38	1.76	1.70
0.2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.06
0.1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.16

Slika 8.21: Prosečno vreme rešavanja svođenjem uspešno rešenih grafova sa 30 čvorova. Vreme je izraženo u sekundama i prikazano u odnosu na veličinu klike i gustinu grafa. Bela polja predstavljaju nepoznato vreme rešavanja.

Glava 9

Zaključci i dalji rad

Jedna od hipoteza ovog rada je da postoje instance NP—kompletnih problema koje se mogu rešiti svođenjem brže nego specijalizovanim rešavačem. Nisu pronađene klase takvih instanci, ali to ne znači da one ne postoje. U ovom radu su opisana i ispitivana svođenja poznata iz literature, često kao standardna, ali se pokazalo da ona ne vode do praktične upotrebljivosti. Ispitana svođenja su uglavnom korišćena za dokaz NP-kompletnosti odgovarajućih problema, pa su iz tog razloga možda neefikasna i rasipna. Na primer, svođenje SAT problema na UHC problem bi se moglo prilagoditi konkretnoj instanci, kako bi se uštedelo na broju čvorova. Takve izmene u algoritmima svođenja ne mogu nikako dovesti do promene klase složenosti, ali bi u slučaju slabijih rešavača mogle napraviti značajnu razliku u vremenu rešavanja. Među prikazanim svođenjima eksperimentalni rezultati sugerišu da svođenje SAT problema na Klika problem i SAT problema na NHC problem čuvaju relativnu težinu instanci. S druge strane, eksperimentalni rezultati sugerišu da svođenje problema Hamiltonovog ciklusa na SAT problem ne čuva težinu instanci. Naime, SAT rešavači imaju jake mehanizme navođenja kroz prostor pretrage ka zadovoljavajućoj valuaciji, a ti mehanizmi nisu od velike koristi kada je ulazna instanca nezadovoljiva. Razmatrajući dimenzije problema i vreme potrebno za njihovo rešavanje stiče se utisak da su SAT rešavači bolji od rešavača za problem klike i problem Hamiltonovog ciklusa, a da je rešavač za problem Hamiltonovog ciklusa bolji od rešavača za problem klike. Takođe, postoji mogućnost da problemi među kojima su vršena svođenja nisu bili najpogodniji za ovaj pristup rešavanju. Eksperimenti su pokazali da pristup rešavanju putem svođenja nije praktično upotrebljiv za NP-kompletne probleme, ali to ne znači da nije praktično upotrebljiv za lakše probleme — za mnoge lakše probleme, svođenje na SAT se već koristi sa velikim uspehom.

Postoji još jedan aspekt svođenja koji nije došao do izražaja, jer je većina eksperimenata izvođena nad malim instancama. Sve i da se nađu pogodnija svođenja, skoro je sigurno da se rešavanje putem svođenja ne može primeniti na zaista velike instance polaznih problema.

U daljem radu bi mogle biti razmatrane modifikacije pojedinačnih svođenja, jer sigurno postoji dosta prostora za poboljšanje. Nije jasno kakav je odnos

samog svođenja i prenošenja relativne težine instanci. Naime, na težinu instanci dobijenih svođenjem utiče i polazna instanca, algoritam svođenja, ali i ciljni alogritam rešavanja (nisu sve instance problema podjednako teške za sve rešavače). Ovaj odnos bi se možda mogao podrobnije ispitati, ukoliko bi se odabrao par NP-kompletnih problema takav da postoje dva "suštinski" različita svođenja sa polaznog problema na ciljni i da postoje dva "suštinski" različita rešavača za rešavanje ciljnog problema. Mozda bi se tada moglo utvrditi kakav je odnos svođenja i prenošenja relativne težine instanci.

Kako su u okviru teze razmatrana svojstava svođenja na instancama u regionu fazne promene SAT problema i problema Hamiltonovog ciklusa, bilo je pogodno ispitati (vrlo grubo) i fenomen fazne promene u Klika problemu (koji nije detaljno ispitan). Zanimljivo je to što ima indikacija da bi fazna promena Klika problema mogla počivati na dva parametra, gustini grafa i veličini klike koja se traži.

Glava 10

Dodatak

10.1 DIMACS formati

Centar za diskretnu matematiku i teorijsko računarstvo (DIMACS[1]) pri Rutgers univerzitetu u Novom Džersiju je u periodu od 1990. do 2005. godine sponzorisao nekoliko implementacionih izazova za razne probleme od značaja. Među rešavanim problemima su problem SAT i problem maksimalne klike. Za potrebe izazova bilo je zgodno da svi poštuju isti ulazni format, i formati koji su korišćeni na DIMACS izazovima su postali izvestan standard koji se i danas koristi. U nastavku su opisana dva DIMACS formata, jedan za predstavljanje iskaznih formula u KNF i drugi za predstavljanje grafova.

DIMACS formati se najčešće sastoje od 3 tipa linija. Linije koje počinju karakterom 'c' predstavljaju komentar linije, i one se mogu nalaziti samo na početku fajla. Linija koja opisuje problem počinje karakterom 'p', i njen oblik se razlikuje od problema do problema. Treći tip, tzv. opisne linije (eng. *descriptor lines*) se takođe razlikuju od problema do problema. Kod formata koji opisuje iskazne formule u KNF, problemska linija je oblika: $p\ cnf\ n\ c$, gde n predstavlja broj promenljivih u formuli, a c predstavlja broj klauza u formuli. Podrazumeva se da su brojevima $1, 2, \ldots, n$ označene promenljive. Svaka opisna linija će predstavljati jednu klauzu, koja je predstavljena nizom literala razdvojenih belinama. Literali se predstavljaju pozitivnim i negativnom brojevima. Pozitivnim brojevima se označava literal koji predstavlja promenljivu sa tim rednim brojem, dok se negativnim brojevima označavaju literali koji predstavljaju negaciju promenljive sa rednim brojem koji je jednak apsolutnoj vrednosti oznake literala. Svaka klauza se završava karakterom '0' kao oznakom kraja klauze (a i same opisne linije). Kod formata koji opisuje grafove, problemska linija ima oblik: $p\ edge\ n\ c$, gde n predstavlja broj čvorova grafa, a c predstavlja broj grana grafa. Podrazumeva se da su brojevima $1, 2, \ldots, n$ označeni čvorovi grafa. Opisne linije imaju sledeći oblik: $e\ i\ j$, gde i i j predstavljaju oznake čvorova. Broj opisnih linija u datoteci mora biti jednak broju grana grafa.

[1] `http://dimacs.rutgers.edu`

Bibliografija

[1] Cook, Stephen A. (1971) - The Complexity of Theorem Proving Procedures Proceedings of the Third Annual ACM Symposium on Theory of Computing, pp. 151-158

[2] Karp, Richard M. (1972) - Reducibility Among Combinatorial Problems Complexity of Computer Computations. New York Plenum, pp 85-103

[3] Iwama, K. and Miyazaki, S. - SAT-Varibale Complexity of Hard Combinatorial Problems Proceedings of the IFIP 13th World Computer Congress, Vol. 1, pp. 253-258, 1994 (Hamburg, Germany)

[4] Levin, Leonid A. (1973) - Universal Sequential Search Problems Problems of Information Transmission, 1973, Vol. 9, Iss. 3, pp. 265-266

[5] Chalaturnyk, Andrew - A Fast Algorithm For Finding Hamilton Cycles Master's thesis. Winnipeg, Manitoba, Canada: University od Manitoba, 2008

[6] Gerold Jäger and Weixiong Zhang - A SAT Based Effective Algorithm for the Directed Hamiltonian Cycle Problem. Computer Science - Theory and Aplications, Lecture Notes in Computer Science, 2010, Vol. 6072/2010, pp. 216-227,Springer, doi:10.1007/978-3-642-13182-0_20

[7] Janičić, Predrag - Matematička Logika u Računarstvu, Matematički fakultet, Beograd, 2008, 4. izdanje (elektronsko izdanje), `http://poincare.matf.bg.ac.rs/~janicic/mlr.zip`

[8] Živković, Miodrag - Algoritmi, Matematički fakultet, Beograd, 2000, `http://poincare.matf.bg.ac.rs/~ezivkovm/nastava/algoritmi.pdf`

[9] Sampo Niskanen and Patric R. J.Östergård - Cliquer User's Guide, Version 1.0, Communications Laboratory, Helsinki University of Technology, Espoo, Finland, Tech. Rep. T48, 2003.

[10] Ratnesh Kumar, Haomin Li - On Asymmetric TSP: Transformation to Symmetric TSP and Performance Bound, Department of Electrical Engineering, University of Kentucky, Lexington, KY 40506-0046

[11] Jon Kleinberg, Eva Tardos - Algorithm Design, Addison-Wesley, 2005

[12] Peter Cheesman, Bob Kanefsky and William M. Taylor (1991) - Where the really hard problems are. In John Myopoulos and Ray Reiter, editors, proceedings of the 12^{th} International Joint Conference on Artificial Intelligence, pages 331-340. Morgan Kaufmann, 1991

[13] Martin Davis, George Logemann and Donald Loveland (1962) - A machine program for theorem proving Communications of the ACM, 5(7):394-397, 1962

[14] G. Tseitin. - On the complexity of proofs in propositional logics. In Automation of Reasoning: Classical Papers in Computational Logic 19671970, vol.2. 1983.

[15] M. James Crawford and D. Larry Auton - Experimental results on the crossover point in random 3-sat. Artificial Intelligence, 81:3157, 1996.

[16] Plotnikov A. D. - Experimental algorithm for the Maximum Independent Set Problem, `http://lanl.arxiv.org/abs/0706.3536`

[17] Basil Vandegriend, Joseph Culbertson (1998)- The G_n, m Phase Transition is Not Hard for the Hamiltonian Cycle Problem, Journal of Artifical Intelligence Research 9, 1998, pg. 219-245

[18] Filip Marić, Predrag Janičić (2011) - Formalization of Abstract State Transition Systems for SAT, eprint arXiv1108.4368, 08/2011