NP-kompletnost

Redukcije

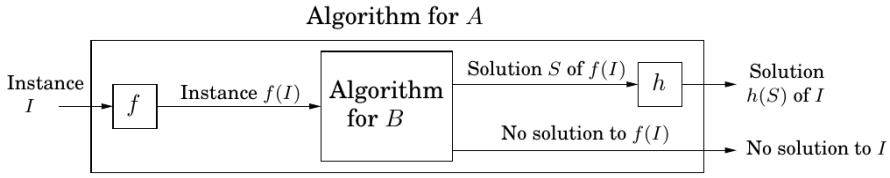
Započećemo ovo poglavlje jednom anegdotom. Matematičaru je objašnjeno kako može da skuva čaj: treba da uzme čajnik, napuni ga vodom iz slavine, stavi čajnik na šporet, sačeka da voda provri, i na kraju stavi kesica čaja u vodu. Kada su ga pitali kako može da skuva čaj ako ima čajnik pun vrele vode? Jednostavno, kaže on: prosuće vodu i tako svesti problem na već rešen.

U ovom poglavlju pozabavićemo se idejom *redukcije* (eng. reduction), odnosno svođenja jednog problema na drugi. Pokazaćemo da redukcije, iako ponekad neracionalne, mogu biti i vrlo korisne. Na primer, ako ubacite u sanduče pošte na Novom Beogradu pismo za čoveka koji stanuje odmah pored te pošte, pismo će, bez obzira na blizinu odredišta, preći put do Glavne pošte u Beogradu, pa nazad do pošte na Novom Beogradu, i tek onda će ga poštar odneti vašem poznaniku. U mnogo slučajeva nije lako prepoznati specijalni slučaj i za njega skrojiti efikasnije specijalno rešenje. U praksi je često efikasnije sve specijalne slučajeve tretirati na isti način. Na takvu situaciju redovno se nailazi i pri konstrukciji algoritama. Kad naiđemo na problem koji se može shvatiti kao specijalni slučaj drugog problema, koristimo poznato rešenje. To rešenje ponekad može biti previše opšte ili previše skupo. Međutim, u mnogim slučajevima je korišćenje opšteg rešenja najlakši, najbrži i najelegantniji način da se problem reši.

Pretpostavimo da je dat problem A, koji izgleda komplikovano, ali liči na poznati problem B. Može se nezavisno (od početka) rešavati problem A, ili se može iskoristiti neki od metoda za rešavanje B i primeniti na A. Postoji, međutim, i treća mogućnost. Može se pokušati sa nalaženjem redukcije, odnosno svođenja jednog problema na drugi. Neformalno, redukcija je rešavanje jednog problema korišćenjem "crne kutije" koja rešava drugi problem. Preciznije, postoji funkcija f kojom se instanca I problema A svodi na instancu $f(I)$ problema B, kao i funkcija h kojom se svako rešenje S instance $f(I)$ prevodi u rešenje $h(S)$ instance I. Redukcijom se može postići jedan od dva cilja, zavisno od smera:

- Rešenje problema A, koje koristi crnu kutiju za rešavanje problema B, može se transformisati u algoritam za rešavanje problema A, ako se zna algoritam za rešavanje problema B.
- S druge strane, ako se za A zna da je težak problem, i zna se donja granica složenosti za algoritme koji rešavaju A, onda je to istovremeno i donja granica složenosti za problem B.

U prvom slučaju je redukcija iskorišćena za dobijanje informacije o problemu A, a u drugom slučaju za dobijanje informacije o problemu B (slika 1).



Slika 1: Redukcija problema A na problem B.

Nalaženje redukcije jednog problema na drugi može biti korisno čak i ako ne daje novu donju ili gornju granicu složenosti problema. Redukcija omogućuje bolje razumevanje oba problema. Ona se može iskoristiti za nalaženje novih tehnika za napad na problem ili njegove varijante.

Pomenimo i to da smo se i tokom prethodnih kurseva susretali sa primerima redukcija — na primer, sa redukcijom problema nalaženja tranzitivnog zatvorenja grafa na problem nalaženja svih najkraćih puteva u grafu.

Klase P i NP

Do sada smo se susreli sa različitim tehnikama za rešavanje algoritamskih problema. Na primer, razmatrali smo algoritme za rešavanje problema određivanja optimalnog uparivanja u bipartitnim grafovima, kao i maksimizaciju toka u transportnoj mreži. Za ove algoritme možemo reći da su efikasni jer im se vreme izvršavanja ponaša kao neka polinomijalna funkcija veličine ulaza. Da bismo više cenili ovako efikasne algoritme, razmotrimo šta je alternativa: u svim ovim problemima mi tražimo rešenje među eksponencijalnim skupom mogućih rešenja. Zaista, postoji $n!$ mogućih uparivanja u grafu, a proizvoljna transportna mreža može imati eksponencijalni broj puteva od izvora s do ponora t. Dakle, ovi problemi bi se mogli rešiti razmatranjem svih mogućih kandidata za rešenje, jednog po jednog, ali bi se na taj način dobio algoritam čija je složenost barem eksponencijalna. Potraga za efikasnim algoritmima se zasniva na pametnim načinima da se zaobiđe proces iscrpne pretrage i da se prostor pretrage suzi.

Bilo bi lepo kad bi svi problemi imali elegantne efikasne algoritme, do kojih se dolazi korišćenjem malog skupa tehnika. Međutim, postoji mnogo problema koji se ne pokoravaju do sada razmotrenim tehnikama. Moguće je da prilikom njihovog rešavanja nije uloženo dovoljno napora, ali se sa dosta razloga može pretpostaviti da postoje problemi koji *nemaju efikasno opšte rešenje*. U ovom poglavlju opisaćemo tehnike za prepoznavanje nekih takvih problema.

Vremenska složenost većine do sada razmatranih algoritama ograničena je nekim polinomom od veličine ulaza. Za takve algoritme kažemo da su *efikasni*, a za odgovarajuće probleme da su *rešivi*. Drugim rečima, za algoritam kažemo da je efikasan ako je njegova vremenska složenost $O(P(n))$, gde je $P(n)$ polinom po veličini problema n. Klasu svih problema koji se mogu rešiti efikasnim al-

goritmom označavaćemo sa P (polinomijalno vreme). Ova definicija može da izgleda čudno na prvi pogled. Tako, na primer, algoritam složenosti $O(n^{10})$ se ni po kojim merilima ne može smatrati efikasnim; slično, teško je algoritam sa vremenom izvršavanja $10^7 n$ smatrati efikasnim, iako je linearne složenosti. Ipak, ova definicija ima smisla. Naime, ispostavlja se da ogromna većina rešivih problema ima praktično upotrebljiva rešenja. Drugim rečima, vremenska složenost polinomijalnih algoritama na koje se u praksi nailazi je najčešće polinom malog stepena, retko iznad kvadratnog. Obrnuto je obično takođe tačno: algoritmi sa vremenskom složenošću većom od proizvoljnog polinoma obično se ne mogu praktično izvršavati za velike ulaze.

Postoji dosta problema za koje se ne zna ni jedan algoritam polinomijalne složenosti. Neki od tih problema će jednom možda biti rešeni efikasnim algoritmima. Međutim, ima razloga za verovanje da se mnogi problemi ne mogu rešiti efikasno. Voleli bismo da budemo u stanju da prepoznamo takve probleme, da ne bismo gubili vreme na traženje nepostojećeg algoritma. U ovom poglavlju razmotrićemo kako pristupati problemima za koje se ne zna da li su u klasi P. Specijalno, razmotrićemo jednu potklasu takvih problema, klasu takozvanih *NP-kompletnih problema* (eng. NP-complete problem). Ovi problemi mogu se grupisati u jednu klasu jer su svi međusobno strogo ekvivalentni — efikasan algoritam za neki NP-kompletan problem postoji ako i samo ako za svaki NP-kompletan problem postoji efikasan algoritam. Široko je rasprostranjeno verovanje da ne postoji efikasan algoritam ni za jedan NP-kompletan problem, ali se ne zna dokaz ovakvog tvrđenja. Čak i kad bi postojali efikasni algoritmi za rešavanje NP-kompletnih problema, oni bi sigurno bili vrlo komplikovani, jer se takvim problemima mnogo istraživača bavilo tokom dugog niza godina. Do ovog trenutka se za stotine (odnosno hiljade, u zavisnosti od načina brojanja tih problema) problema zna da su NP-kompletni, što ovu oblast čini vrlo značajnom.

Ovo poglavlje sastoji se iz dva dela. U prvom delu se definiše klasa NP-kompletnih problema i navode primeri dokaza pripadnosti nekih problema toj klasi. U drugom delu poglavlja ćemo razmotriti nekoliko tehnika za rad sa NP-kompletnim problemima: jedan od njih je konstrukcija približnih algoritama polinomijalne vremenske složenosti za njihovo rešavanje.

Redukcije polinomijalne vremenske složenosti

U ovom odeljku ograničićemo se na *probleme odlučivanja*, tj. razmatraćemo samo one probleme na koje se posle izvršavanja određenog algoritma može odgovoriti sa "da" ili "ne". Ovo ograničenje uprošćava razmatranja, a ne menja težinu problema koji se rešava, jer se ove dve verzije mogu svesti jedna na drugu. Na primer, umesto da tražimo maksimalnu kliku (potpuno povezani podraf grafa, tj. podskup čvorova grafa takav da između svaka dva čvora u podskupu postoji grana), možemo da postavimo pitanje da li za zadato k postoji klika veličine bar k. Ako umemo da rešimo problem odlučivanja, obično možemo da rešimo i polazni problem – binarnom pretragom, na primer. Obratno, takođe, trivijalno važi: ako znamo da odredimo maksimalnu kliku u grafu, jedino je potrebno

proveriti da li je njena veličina veća ili jednaka k.

Problem odlučivanja može se posmatrati kao *problem prepoznavanja jezika.* Neka je U skup mogućih ulaza za problem odlučivanja. Neka je $L \subseteq U$ skup svih ulaza za koje je rešenje problema "da". Za L se kaže da je *jezik* koji odgovara problemu, pa pojmovi problem i jezik mogu da se koriste ravnopravno. Problem odlučivanja je ustanoviti da li zadati ulaz pripada jeziku L. Sada ćemo uvesti pojam redukcije polinomijalne složenosti između jezika, kao osnovni alat u ovom poglavlju.

Definicija 1. *Neka su L_1 i L_2 dva jezika, podskupa redom skupova ulaza U_1 i U_2. Kažemo da je L_1* polinomijalno svodljiv *na L_2, ako postoji algoritam polinomijalne vremenske složenosti, koji dati ulaz $u_1 \in U_1$ prevodi u ulaz $u_2 \in U_2$, tako da $u_1 \in L_1$ ako i samo ako $u_2 \in L_2$. Pritom algoritam treba da bude polinomijalan u odnosu na veličinu ulaza u_1.*

Algoritam iz definicije svodi jedan problem na drugi.

Ako znamo algoritam za prepoznavanje jezika L_2, onda ga možemo superponirati sa algoritmom redukcije i tako dobiti algoritam za prepoznavanje jezika L_1. Označimo algoritam redukcije sa AR, a algoritam za prepoznavanje jezika L_2 sa AL_2. Proizvoljni ulaz $u_1 \in U_1$ može se primenom algoritma AR transformisati u ulaz $u_2 \in U_2$, i primenom algoritma AL_2 ustanoviti da li $u_2 \in L_2$, a time i da li $u_1 \in L_1$. Specijalno, ako je problem prepoznavanja jezika L_2 iz klase P, zaključujemo da je i problem prepoznavanja jezika L_1 iz klase P.

Teorema 1. *Ako je jezik L_1 polinomijalno svodljiv na jezik L_2, i $L_2 \in P$, onda je i $L_1 \in P$.*

Relacija polinomijalne svodljivosti nije simetrična: polinomijalna svodljivost jezika L_1 na jezik L_2 ne povlači polinomijalnu svodljivost jezika L_2 na jezik L_1. Ova asimetrija potiče od činjenice da definicija svodljivosti zahteva da se proizvoljan ulaz za L_1 može transformisati u ekvivalentan ulaz za L_2, ali ne i obrnuto. Moguće je da ulazi za L_2, dobijeni na ovaj način, predstavljaju samo mali deo svih mogućih ulaza za L_2. Prema tome, ako je jezik L_1 polinomijalno svodljiv na jezik L_2, onda možemo smatrati da je problem prepoznavanja jezika L_2 teži.

Dva jezika L_1 i L_2 su *polinomijalno ekvivalentni*, ako je svaki od njih polinomijalno svodljiv na drugi. Specijalno, svi netrivijalni problemi iz klase P su ekvivalentni. Dokažimo ovo tvrđenje. Neka su B i C dva netrivijalna problema iz P (tj. za oba postoje ulazi sa odgovorom "da" i ulazi sa odgovorom "ne"). Neka su u_0 i u_1 dva ulaza za problem C, takva da je za u_0 rešenje "ne", a za u_1 "da" (odnosno $u_0 \notin L_C$ i $u_1 \in L_C$). Za svođenje problema B na problem C može se iskoristiti sledeći polinomijalni algoritam: ulazu v za problem B pridružuje se $\phi(v) = u_0$ ako $v \notin L_B$, odnosno $\phi(v) = u_1$ ako $v \in L_B$[1]. Tada za proizvoljan ulaz v za problem B važi $v \in L_B$ akko $\phi(v) \in L_C$.

[1] Primetimo da je provera da li $v \in L_B$ polinomijalne složenosti.

Relacija polinomijalne svodljivosti je tranzitivna, kao što pokazuje sledeća teorema.

Teorema 2. *Ako je L_1 polinomijalno svodljiv na L_2 i L_2 je polinomijalno svodljiv na L_3, onda je L_1 polinomijalno svodljiv na L_3.*

Dokaz: Neka su jezici L_1, L_2 i L_3 podskupovi skupova mogućih ulaza redom U_1, U_2 i U_3. Superponiranjem algoritama redukcije L_1 na L_2, odnosno L_2 na L_3 dobija se algoritam redukcije L_1 na L_3. Proizvoljan ulaz $u_1 \in U_1$ konvertuje se najpre u ulaz $u_2 \in U_2$, koji se zatim konvertuje u ulaz $u_3 \in U_3$. Pošto su redukcije polinomijalne složenosti, a kompozicija dve polinomijalne funkcije je polinomijalna funkcija, rezultujući algoritam redukcije je takođe polinomijalne složenosti (to je jedan od razloga zašto su za meru složenosti rešivih problema izabrani polinomi). □

Suština metoda koji ćemo primenjivati u ovom poglavlju je da ako se za neki problem ne može naći efikasan algoritam, onda pokušavamo da ustanovimo da li je on ekvivalentan nekom od problema za koje znamo da su teški. Klasa NP-kompletnih problema sadrži stotine takvih međusobno ekvivalentnih problema.

Nedeterminizam i Kukova teorema

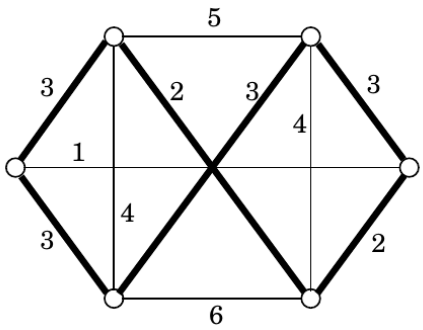
Sada ćemo definisati drugu važnu klasu jezika (odnosno problema odlučivanja) — klasu NP. Razmotrimo problem trgovačkog putnika.

Problem. Zadat je težinski graf $G = (V, E)$ sa $|V| = n$ čvorova, tako da je za proizvoljne čvorove – gradove $v_i, v_j \in V$ težina grane $(v_i, v_j) \in E$ jednaka $d(v_i, v_j)$ i broj $B \in \mathbf{Z}^+$ – budžet. Ustanoviti da li u grafu G postoji Hamiltonov ciklus sa zbirom težina grana manjim ili jednakim od B (slika 2). Drugim rečima, postoji li permutacija $\pi(1), \pi(2), \ldots, \pi(n)$ čvorova grafa tako da kada se oni posete u ovom redosledu ukupni pređeni put je najviše B:

$$\sum_{i=1}^{n-1} d(v_{\pi(i)}, v_{\pi(i+1)}) + d(v_{\pi(n)}, v_{\pi(1)}) \leq B?$$

Polinomijalni algoritam za rešavanje ovog problema nije poznat. Pretpostavimo, međutim, da je za neki konkretan ulaz za ovaj problem dobijen odgovor "da". Ako u to posumnjamo, možemo zahtevati "dokaz" tog tvrđenja — niz čvorova koji ima traženu osobinu. Kad imamo ovakav niz čvorova, lako možemo da proverimo da li je to Hamiltonov ciklus, da izračunamo njegovu dužinu i uporedimo je sa zadatom granicom B. Pored toga, očigledno je vremenska složenost ovakvog algoritma provere polinomijalna u odnosu na veličinu ulaza.

Upravo pojam *polinomijalne proverljivosti* karakteriše klasu NP. Primetimo da proverljivost za polinomijalno vreme ne povlači i mogućnost rešavanja za polinomijalno vreme. Utvrđujući da se za polinomijalno vreme može proveriti odgovor "da" za problem trgovačkog putnika, mi ne uzimamo u obzir vreme



Slika 2: Problem trgovačkog putnika čije je optimalno rešenje dužine 18.

koje nam može biti potrebno za pronalaženje jednog od mogućih Hamiltonovih ciklusa, čiji broj raste eksponencijalno sa brojem čvorova grafa. Mi samo tvrdimo da se za svaki zadati niz čvorova i za konkretan ulaz u, za polinomijalno vreme može proveriti da li taj niz "dokazuje" da je za ulaz u odgovor "da".

Klasa NP se na drugi način neformalno može definisati pomoću pojma *nedeterminističkog algoritma*. Takav algoritam sastoji se od dve različite faze: *faze pogađanja* i *faze provere*. Za zadati ulaz u, u prvoj fazi se izvodi prosto "pogađanje" nekog rešenja S. Zatim se u i S zajedno predaju kao ulaz fazi provere, koja se izvodi na običan (deterministički) način, pa se završava odgovorom "da" ili "ne", ili se izvršava beskonačno dugo. Nedeterministički algoritam "rešava" problem odlučivanja Π, ako su za proizvoljni ulaz $u \in U_\Pi$ za ovaj problem ispunjena sledeća dva uslova:

1. Ako $u \in L_\Pi$, onda postoji predloženo rešenje S, čije bi pogađanje za ulaz u dovelo do toga da se faza provere sa ulazom (u, S) završi odgovorom "da".
2. Ako $u \notin L_\Pi$, onda ne postoji predloženo rešenje S, čije bi pogađanje za ulaz u obezbedilo završavanje faze provere sa ulazom (u, S) odgovorom "da".

Na primer, nedeterministički algoritam za rešavanje problema trgovačkog putnika mogao bi se konstruisati koristeći kao fazu pogađanja izbor proizvoljnog niza gradova (čvorova), a kao fazu provere — gore pomenuti algoritam "provere dokaza" za problem trgovačkog putnika. Očigledno je da za proizvoljan konkretan ulaz u postoji takvo pogađanje S, da se kao rezultat rada faze provere sa ulazom (u, S) dobija "da", onda i samo onda, ako za ulaz u postoji Hamiltonov ciklus tražene dužine.

Kaže se da nedeterministički algoritam koji rešava problem odlučivanja Π radi za "polinomijalno vreme", ako postoji polinom p takav da za svaki ulaz $u \in U_\Pi$ postoji takvo pogađanje S, da se faza provere sa ulazom (u, S) završava sa odgovorom "da" za vreme $p(|u|)$. Iz toga sledi da je veličina rešenja S obavezno ograničena polinomom od veličine ulaza, jer se na proveru pogađanja S može

utrošiti najviše polinomijalno vreme.

Klasa NP, definisana neformalno, — to je klasa svih problema odlučivanja koji pri razumnom kodiranju mogu biti rešeni nedeterminističkim (N – nondeterministic) algoritmom za polinomijalno (P – polynomial) vreme. Na primer, problem trgovačkog putnika pripada klasi NP.

U sličnim neformalnim definicijama termin "rešava" treba oprezno koristiti. Treba da bude jasno da je osnovni smisao "polinomijalnog nedeterminističkog algoritma" u tome da objasni pojam "proverljivosti za polinomijalno vreme", a ne u tome da bude realni metod rešavanja problema odlučivanja. Za svaki konkretan ulaz takav algoritam ima ne jedno, nego nekoliko mogućih izvršavanja — po jedno za svako moguće pogađanje.

Nedeterministički algoritmi su vrlo moćni, ali njihova snaga nije neograničena. Postoje problemi koji se ne mogu efikasno rešiti nedeterminističkim algoritmom.

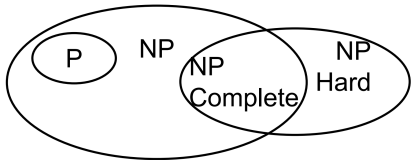
Na primer, posmatrajmo sledeći problem: da li je veličina maksimalne klike u zadatom grafu jednaka tačno k? Nedeterminističkim algoritmom lako se može pronaći klika veličine k, ako ona postoji, ali se ne može lako ustanoviti (čak ni nedeterministički) da ne postoji veća klika.

Izgleda razumno smatrati da su nedeterministički algoritmi moćniji od determinističkih, ali da li je to tačno? Jedan način da se to dokaže bio bi da se pronađe neki problem koji je u klasi NP a koji nije u klasi P. To do sada nikome nije pošlo za rukom. U protivnom, da bi se dokazalo da su ove dve klase jednake (odnosno $P = NP$), trebalo bi pokazati da svaki problem iz klase NP može da bude rešen determinističkim algoritmom polinomijalne vremenske složenosti. Ni ovakvo tvrđenje niko nije uspeo da dokaže (i malo je onih koji veruju u njegovu tačnost). Problem utvrđivanja odnosa između P i NP poznat je kao problem $P = NP$.

Definicija 2. *Problem X je NP-težak* problem *ako je svaki problem iz klase NP polinomijalno svodljiv na X.*

Definicija 3. *Problem X je NP-kompletan* problem *ako (1) pripada klasi NP, i (2) X je NP-težak.*

Posledice definicije klase NP-teških problema je da, ako se za bilo koji NP-težak problem dokaže da pripada klasi P, onda je $P = NP$.



Slika 3: Odnos klasa složenosti.

Kuk je 1971. godine dokazao da postoje NP-kompletni problemi, tako što je za jedan problem dokazao da je NP-kompletan. Primetimo da kada se zna jedan NP-kompletan problem, dokazi da su drugi problemi NP-kompletni postaju jednostavniji. Naime, kako bi se za novi problem X dokazalo da je NP-težak, nije više neophodno dokazivati da je svaki problem iz klase NP polinomijalno svodljiv na problem X, već je dovoljno dokazati da je neki NP-kompletan problem polinomijalno svodljiv na X. To sledi iz sledeće leme.

Lema 1. Problem X je NP-kompletan ako (1) X pripada klasi NP, i (2') postoji NP-kompletan problem Y koji je polinomijalno svodljiv na X.

Dokaz: Problem Y je prema uslovu (2) NP-težak, pa je svaki problem u klasi NP polinomijalno svodljiv na Y. Pošto je Y polinomijalno svodljiv na X, a polinomijalna svodljivost je tranzitivna relacija, svaki problem u klasi NP je polinomijalno svodljiv i na problem X. Odavde sledi da je problem X NP-težak. □

Mnogo je lakše dokazati da su dva problema polinomijalno svodljiva, nego direktno dokazati da je ispunjen uslov (2). Zbog toga je Kukov rezultat kamen temeljac cele teorije[2]. Dalje, sa pojavom novih problema za koje se zna da su NP-kompletni, raste broj mogućnosti za dokazivanje uslova (2'). Ubrzo pošto je Kukov rezultat postao poznat, Karp (R.M.Karp) je za 24 važna problema pokazao da su NP-kompletni. Od tog vremena je za stotine problema (možda hiljade, zavisno od načina brojanja varijacija istog problema) dokazano da su NP-kompletni. U sledećem odeljku prikazaćemo nekoliko primera NP-kompletnih problema, sa dokazima njihove NP-kompletnosti. Navešćemo takođe (bez dokaza) više NP-kompletnih problema. Najteži deo dokaza je obično (ne uvek) dokaz ispunjenosti uslova (2').

Izložićemo sada problem za koji je Kuk dokazao da je NP-kompletan. Problem je poznat kao *problem zadovoljivosti* (SAT, skraćenica od satisfiability). Neka je S Bulov izraz u konjuktivnoj normalnoj formi. Drugim rečima, S je konjunkcija više klauza (disjunkcija grupa literala — simbola promenljivih ili njihovih negacija). Kao primer može da posluži izraz $S = (x + y + \bar{z}) \cdot (\bar{x} + y + z) \cdot (\bar{x} + \bar{y} + \bar{z})$, gde sabiranje, odnosno množenje odgovaraju disjunkciji, odnosno konjunkciji (ili, odnosno i), a svaka promenljiva ima vrednost 0 (netačno) ili 1 (tačno). Poznato je da se svaka Bulova funkcija može predstaviti izrazom u KNF. Za Bulov izraz se kaže da je *zadovoljiv*, ako postoji takvo dodeljivanje nula i jedinica promenljivim, da izraz ima vrednost 1. Problem SAT sastoji se u utvrđivanju da li je zadati izraz zadovoljiv (pri čemu nije neophodno pronaći odgovarajuće vrednosti promenljivih). U navedenom primeru izraz S je zadovoljiv, jer za $x = 1$, $y = 1$ i $z = 0$ ima vrednost $S = 1$.

Problem SAT je u klasi NP jer se za (nedeterministički) izabrane vrednosti promenljivih za polinomijalno vreme (od veličine ulaza — ukupne dužine formule)

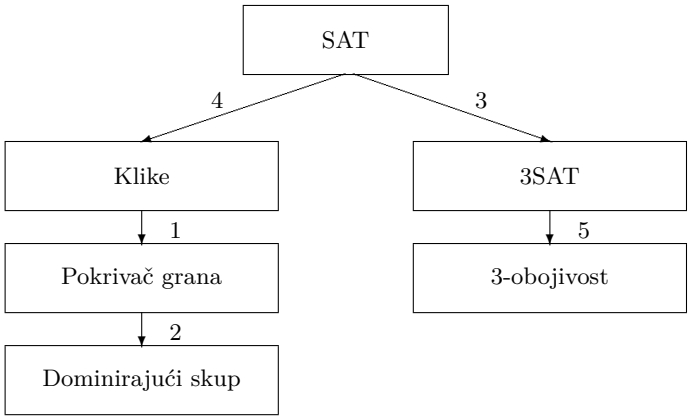
[2]Stiven Kuk je 1982. godine dobio Tjuringovu nagradu za doprinose teoriji složenosti.

može proveriti da li je izraz tačan. Problem SAT je i NP-težak. To nije lako dokazati.

Teorema 3 (Kukova teorema)**.** *Problem SAT je NP-kompletan.*

Primeri dokaza NP-kompletnosti

U ovom odeljku dokazaćemo da su NP-kompletni sledećih pet problema: pokrivač grana, dominirajući skup, 3SAT, 3-obojivost i problem klika. Svaki od ovih problema biće detaljnije izložen. Dokazivanje NP-kompletnosti zasniva se na tehnikama koje će biti rezimirane na kraju odeljka. Da bi se dokazala NP-kompletnost nekog problema, mora se najpre dokazati da on pripada klasi NP, što je obično (ali ne uvek!) lako, a zatim treba pronaći redukciju polinomijalne vremenske složenosti nekog problema za koji se zna da je NP-kompletan na naš problem. Redosled redukcija u dokazima NP-kompletnosti pet navedenih problema prikazan je na slici~4. Da bi se ovi dokazi lakše razumeli, navedeni su redosledom prema težini, a ne prema rastojanju od korena stabla na slici~4; redosled dokaza prikazan je brojevima uz grane.



Slika 4: Redosled dokaza NP-kompletnosti u tekstu.

Pokrivač grana

Neka je $G = (V, E)$ neusmereni graf. *Pokrivač grana* grafa G je takav skup čvorova, da je svaka grana G susedna bar jednom od čvorova iz skupa.

Problem. Zadat je neusmereni graf $G = (V, E)$ i prirodni broj k. Ustanoviti da li u G postoji pokrivač grana sa $\leq k$ čvorova.

Teorema 4. *Problem pokrivač grana je NP-kompletan.*

Dokaz: Problem pokrivač grana je u klasi NP, jer se za pretpostavljeni podskup od $\leq k$ čvorova lako proverava za polinomijalno vreme da li čini pokrivač grana grafa.

Da bismo dokazali da je problem pokrivač grana NP-kompletan, treba da na njega svedemo neki NP-kompletan problem. U tu svrhu iskoristićemo problem klika (dokaz da je problem klika NP-kompletan biće dat u nastavku. U neusmerenom grafu $G = (V, E)$ *klika* je takav podgraf C grafa G u kome su svi čvorovi međusobno povezani granama iz G. Drugim rečima, klika je kompletan podgraf. Problem klika glasi: za zadati graf G i prirodni broj k, ustanoviti da li G sadrži kliku veličine k. Potrebno je transformisati proizvoljan ulaz za problem klika u ulaz za problem pokrivač grana, tako da je rešenje problema klika "da" akko je "da" rešenje odgovarajućeg problema pokrivač grana. Neka je $G = (V, E)$, k proizvoljan ulaz za problem klika. Neka je $\bar{G} = (V, \bar{E})$ komplement grafa G, odnosno graf sa istim skupom čvorova kao i G, i komplementarnim skupom grana u odnosu na G (odnosno između proizvoljna dva čvora u $\bar{G}$ grana postoji akko između ta dva čvora u G ne postoji grana). Neka je $n = |V|$. Tvrdimo da je problem klika (G, k) sveden na problem pokrivač grana grafa, sa ulazom $\bar{G}$, $n - k$ (videti primer na slici 5, gde su isprekidanim linijama prikazane grane iz G).

- Pretpostavimo da je $C = (U, F)$ klika u G. Skup čvorova $V \setminus U$ pokriva sve grane u $\bar{G}$, jer u $\bar{G}$ nema grana koje povezuju čvorove iz U (sve te grane su u G). Prema tome, $V \setminus U$ je pokrivač grana za $\bar{G}$. Drugim rečima, ako G ima kliku veličine k, onda $\bar{G}$ ima pokrivač grana veličine $n - k$.
- Obrnuto, neka je D pokrivač grana u $\bar{G}$. Tada D pokriva sve grane u $\bar{G}$, pa u $\bar{G}$ ne može da postoji grana koja povezuje neka dva čvora iz $V \setminus D$. Prema tome, $V \setminus D$ je klika u G. Zaključujemo da ako u $\bar{G}$ postoji pokrivač grana veličine $n - k$, onda u G postoji klika veličine k.

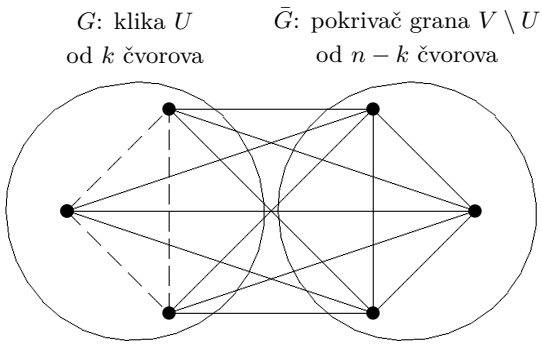
Ova redukcija se očigledno može izvršiti za polinomijalno vreme: potrebno je samo konstruisati graf $\bar{G}$ polazeći od grafa G i izračunati razliku $n - k$. □

Dominirajući skup

Neka je $G = (V, E)$ neusmereni graf. *Dominirajući skup* je skup čvorova $D \subset V$, takav da za svaki čvor grafa G važi da ili pripada skupu D, ili je susedan bar jednom čvoru iz D.

Problem. Dat je neusmereni graf $G = (V, E)$ i prirodni broj k. Ustanoviti da li u G postoji dominirajući skup sa najviše k čvorova.

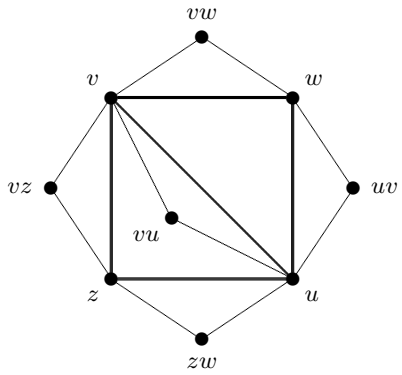
Teorema 5. *Problem dominirajući skup je NP-kompletan.*



Slika 5: Redukcija problema klika na problem pokrivač grana.

Dokaz: Problem dominirajući skup je u klasi NP, jer se za pretpostavljeni podskup od najviše k čvorova lako za polinomijalno vreme proverava da li je dominirajući skup.

Izvešćemo redukciju problema pokrivač grana na problem dominirajući skup. Ako je zadat proizvoljan ulaz (G, k) za problem pokrivač grana, cilj je konstruisati novi graf G' koji ima dominirajući skup određene veličine akko G ima pokrivač grana veličine najviše k. Pri tome se, bez smanjenja opštosti, može pretpostaviti da G nema izolovanih čvorova (oni ne utiču na pokrivač grana, ali moraju biti uključeni u dominirajući skup). Polazeći od grafa G, dodajemo mu $|E|$ novih čvorova i $2|E|$ novih grana na sledeći način. Za svaku granu (v, w) iz G dodajemo novi čvor vw i dve nove grane (v, vw) i (w, vw) (videti primer na slici 6). Drugim rečima, svaku granu transformišemo u trougao. Označimo novi graf sa G'. Graf G' je lako konstruisati za polinomijalno vreme.



Slika 6: Redukcija problema pokrivač grana na problem dominirajući skup.

Tvrdimo da G ima pokrivač grana veličine m akko G' ima dominirajući skup veličine m.

- Neka je D dominirajući skup grafa G'. Ako D sadrži bilo koji od novih čvorova vw, onda se takav čvor može zameniti bilo čvorom v, bilo čvorom w, posle čega će i novi skup biti dominirajući skup (v i w pokrivaju sve čvorove koje pokriva vw). Prema tome, bez smanjenja opštosti može se pretpostaviti da D sadrži samo čvorove iz G. Međutim, pošto D "dominira" nad svim novim čvorovima, on mora za svaku granu iz G da sadrži bar jedan njen kraj, pa je zbog toga D istovremeno i pokrivač grana grafa G.
- Obrnuto, ako je C pokrivač grana u G, onda je svaka grana G susedna nekom čvoru iz C, pa je i svaki novi čvor iz G' susedan nekom čvoru iz C. Stari čvorovi su takođe pokriveni čvorovima iz C, jer po pretpostavci čvorovi iz C pokrivaju sve grane.

□

3SAT

Problem 3SAT je uprošćena verzija običnog problema SAT. Ulaz za problem 3SAT je KNF u kojoj svaka klauza ima tačno tri literala.

Problem. Zadat je Bulov izraz u KNF, u kome svaka klauza sadrži tačno tri literala. Ustanoviti da li je izraz zadovoljiv.

Teorema 6. *Problem 3SAT je NP-kompletan.*

Dokaz: Ovo je interesantna i uobičajena vrsta redukcije, sa problema na njegov specijalan slučaj. Ovaj problem je na prvi pogled lakši od običnog problema SAT, zbog dopunskog ograničenja da svaka klauza ima po tri promenljive. Međutim, pokazaćemo da ovaj problem ostaje težak i kada je ulaz restrihovaniji. Pokazaćemo da algoritam koji rešava 3SAT može da se iskoristi da reši običan problem SAT (odnosno da se SAT može svesti na 3SAT). Pre toga, jasno je da SAT pripada klasi NP. Mogu se izabrati ("pogoditi") vrednosti promenljivih i za polinomijalno vreme proveriti da li je izraz tačan. Neka je E proizvoljan ulaz za SAT. Svaku klauzu u E zamenićemo sa nekoliko klauza od po tačno tri literala. Neka je $C = (x_1 + x_2 + \cdots + x_k)$ proizvoljna klauza iz E, takva da je $k \geq 4$. Ovde je zbog udobnosti sa x_i označen literal, odnosno bilo promenljiva, bilo negacija promenljive. Pokazaćemo kako se C može ekvivalentno zameniti sa nekoliko klauza od po tačno tri literala. Ideja je uvesti nove promenljive $y_1, y_2, \ldots, y_{k-3}$, koje klauzu transformišu u deo ulaza za 3SAT, ne menjajući zadovoljivost izraza. Za svaku klauzu iz E uvode se nove, različite promenljive; C se zamenjuje konjunkcijom klauza C', tako da je

$$C' = (x_1+x_2+y_1)(x_3+\bar{y}_1+y_2)(x_4+\bar{y}_2+y_3)\cdots(x_{k-2}+\bar{y}_{k-4}+y_{k-3})(x_{k-1}+x_k+\bar{y}_{k-3}).$$

Tvrdimo da je izraz, dobijen od E zamenom C sa C', zadovoljiv akko je zadovoljiv izraz E.

- Ako je izraz E zadovoljiv, onda bar jedan od literala x_i mora imati vrednost 1. U tom slučaju se mogu izabrati vrednosti promenljivih y_i u C' tako da sve klauze u C' budu tačne. Na primer, ako je $x_3 = 1$, onda se može staviti $y_1 = 1$ (što čini tačnom prvu klauzu), $y_2 = 0$ (druga klauza je tačna zbog $x_3 = 1$), i $y_i = 0$ za sve $i > 2$. Uopšte, ako je $x_i = 1$, onda stavljamo $y_1 = y_2 = \cdots = y_{i-2} = 1$ i $y_{i-1} = y_i = \cdots = y_{k-3} = 0$, što obezbeđuje da bude $C' = 1$.
- Obrnuto, ako izraz C' ima vrednost 1, tvrdimo da bar jedan od literala x_i mora imati vrednost 1. Zaista, ako bi svi literali x_i imali vrednost 0, onda bi izraz C' imao istu tačnost kao i izraz $C'' = (y_1) \cdot (\bar{y}_1 + y_2) \cdot (\bar{y}_2 + y_3) \cdots (\bar{y}_{k-4} + y_{k-3}) \cdot (\bar{y}_{k-3})$, koji očigledno nije zadovoljiv (da bi bilo $C'' = 1$, moralo bi da bude redom $y_1 = 1$, pa $y_2 = 1$, itd, $y_{k-4} = 1$, $y_{k-3} = 0$, i $y_{k-3} = 1$, što je kontradikcija).

Pomoću ove redukcije sve klauze sa više od tri literala mogu se zameniti sa nekoliko klauza od po tačno tri literala. Ostaje da se transformišu klauze sa jednim ili dva literala. Klauza oblika $C = (x_1 + x_2)$ zamenjuje se ekvivalentnom izrazom

$$C' = (x_1 + x_2 + z)(x_1 + x_2 + \bar{z}),$$

gde je z nova promenljiva. Lako se pokazuje da je početni izraz zadovoljiv akko je zadovoljiv izraz dobijen zamenom C sa C'.

Konačno, klauza oblika $C = x_1$ može se zameniti izrazom

$$C' = (x_1 + y + z)(x_1 + \bar{y} + z)(x_1 + y + \bar{z})(x_1 + \bar{y} + \bar{z}),$$

u kome su y i z nove promenljive. Lako se pokazuje da je početni izraz zadovoljiv akko je zadovoljiv izraz dobijen zamenom C sa C'.

Prema tome, proizvoljni ulaz za problem SAT može se svesti na ulaz za problem 3SAT, tako da je prvi izraz zadovoljiv akko je zadovoljiv drugi. Jasno je da se ova redukcija izvodi za polinomijalno vreme. □

Klike

Problem klika definisan je u odeljku, u kome je razmatran problem pokrivač grana.

Problem. Dat je neusmereni graf $G = (V, E)$ i prirodni broj k. Ustanoviti da li G sadrži kliku veličine bar k.

Teorema 7. *Problem klika je NP-kompletan.*

Dokaz: Problem klika je u klasi NP, jer se za svaki pretpostavljeni podskup od k čvorova za polinomijalno vreme može proveriti da li je klika.

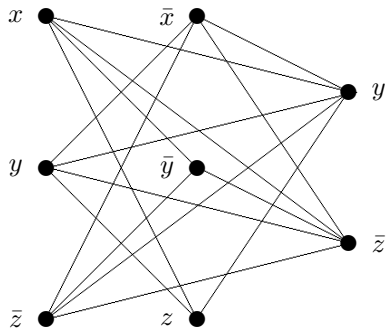
Pokazaćemo sada da se problem SAT može svesti na problem klika. Ova redukcija povezuje dva veoma različita problema, jedan koji radi nad

skupom klauza i drugi koji radi nad grafovima. Neka je E proizvoljni Bulov izraz u KNF, $E = E_1 \cdot E_2 \cdots E_m$. Posmatrajmo, na primer, klauzu $E_i = (x + y + z + w)$. Njoj pridružujemo "kolonu" od četiri čvora, označena literalima iz E_i (bez obzira što se neki od njih možda pojavljuju i u drugim klauzama). Drugim rečima, graf G koji konstruišemo imaće po jedan čvor za svaku pojavu bilo koje promenljive. Ostaje pitanje kako povezati ove čvorove, tako da G sadrži kliku veličine bar k akko je izraz E zadovoljiv. Primetimo da se vrednost k može izabrati proizvoljno, jer je potrebno svesti problem SAT na problem klika, odnosno rešiti problem SAT koristeći algoritam za rešavanje problema klika. Naravno, algoritam za rešavanje problema klika mora da radi za svaku vrednost k. Ovo je važna fleksibilnost, koja se često koristi u dokazima NP-kompletnosti. U ovom slučaju za k ćemo izabrati vrednost jednaku broju klauza m.

Grane u grafu G mogu se zadati na sledeći način. Čvorovi iz iste kolone (odnosno čvorovi pridruženi literalima iz iste klauze) ne povezuju se granama. Čvorovi iz različitih kolona su skoro uvek povezani: izuzetak je slučaj dva čvora od kojih jedan odgovara promenljivoj, a drugi komplementu te iste promenljive. Primer grafa koji odgovara izrazu

$$E = (x + y + \bar{z}) \cdot (\bar{x} + \bar{y} + z) \cdot (y + \bar{z})$$

prikazan je na slici 7. Jasno je da se G može konstruisati za polinomijalno vreme.



Slika 7: Redukcija problema SAT sa ulazom $(x + y + \bar{z}) \cdot (\bar{x} + \bar{y} + z) \cdot (y + \bar{z})$ na problem klika.

Tvrdimo da G ima kliku veličine bar m, akko je izraz E zadovoljiv. Najpre zapažamo da zbog konstrukcije maksimalna klika ne može imati više od m čvorova, nezavisno od E.

- Pretpostavimo da je izraz E zadovoljiv. Tada postoji takvo dodeljivanje vrednosti promenljivim, da u svakoj klauzi postoji bar jedan literal sa vrednošću 1. Čvor koji odgovara tom literalu priključuje se kliki (ako ima više takvih literala, bira se proizvoljan od njih).

Dobijeni podgraf jeste klika, jer jedini način da dva čvora iz različitih kolona ne budu povezana je da jedan od njih bude komplement drugog — što je nemoguće, jer je svim izabranim literalima dodeljena vrednost 1.

- Obrnuto, pretpostavimo da G sadrži kliku veličine bar m. Klika mora da sadrži tačno jedan čvor iz svake kolone (jer po konstrukciji čvorovi iz iste kolone nisu povezani). Odgovarajućim literalima dodeljujemo vrednost 1. Ako na ovaj način nekoj promenljivoj nije dodeljena vrednost, to se može učiniti na proizvoljan način. Izvedeno dodeljivanje vrednosti promenljivima je neprotivrečno: ako bi nekoj promenljivoj x i njenom komplementu $\bar{x}$, uključenim u kliku, istovremeno bila dodeljena vrednost 1, to bi značilo da odgovarajući čvorovi (po konstrukciji) nisu povezani — suprotno pretpostavci da su oba čvora u kliki.

□

3-obojivost

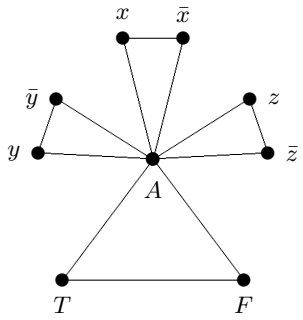
Neka je $G = (V, E)$ neusmereni graf. *Ispravno bojenje* (ili samo bojenje) grafa G je takvo pridruživanje boja čvorovima, da je svakom čvoru pridružena neka boja, a da su susednim čvorovima uvek pridružene različite boje.

Problem (3-obojivost)**.** Dat je neusmereni graf $G = (V, E)$. Ustanoviti da li se G može obojiti sa tri boje.

Teorema 8. *Problem 3-obojivost je NP-kompletan.*

Dokaz: Problem 3-obojivost pripada klasi NP, jer se može pretpostaviti proizvoljno bojenje grafa sa 3 boje, a zatim za polinomijalno vreme proveriti da li je pretpostavljeno bojenje ispravno. Izvešćemo redukciju problema 3SAT na problem 3-obojivost. Dokaz je nešto komplikovaniji iz dva razloga. Najpre, problemi se odnose na različite objekte — Bulove izraze u KNF, odnosno grafove. Drugo, ne može se prosto zameniti jedan objekat (na primer čvor ili grana) drugim (na primer klauzom); mora se voditi računa o kompletnoj strukturi. Ideja je da se iskoriste sastavni elementi, koji se povezuju u celinu. Neka je E proizvoljan ulaz za problem 3SAT. Treba konstruisati graf G, tako da je izraz E zadovoljiv akko je G 3-obojiv. Najpre konstruišemo osnovni trougao M. Pošto je M trougao (kompletni graf sa tri čvora), za njegovo bojenje potrebne su tačno tri boje. Označimo te boje sa T (tačno), F (netačno) i A, videti donji trougao na slici 8. Pored toga, za svaku promenljivu x dodajemo novi trougao M_x, čije čvorove označavamo sa x, $\bar{x}$ i A, pri čemu se čvor označen sa A poklapa sa čvorom iz M sa istom oznakom. Prema tome, ako se u izrazu pojavljuje k promenljivih, imamo $k + 1$ trouglova sa zajedničkim čvorom A (slika 8). Ovo za sada obezbeđuje da, ako je, na primer, čvor x obojen

bojom T, onda čvor $\bar{x}$ mora biti obojen bojom F (jer su oba čvora susedna čvoru obojenom bojom A), i obrnuto. Ovo je u skladu sa značenjem $\bar{x}$.

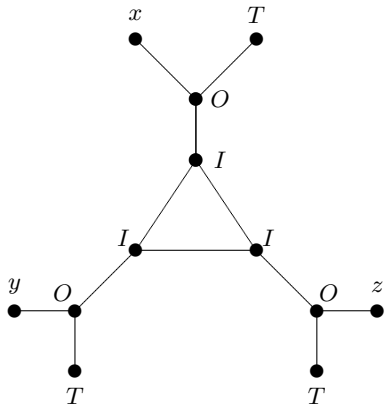


Slika 8: Prvi deo konstrukcije za redukciju 3SAT na 3-obojivost grafa.

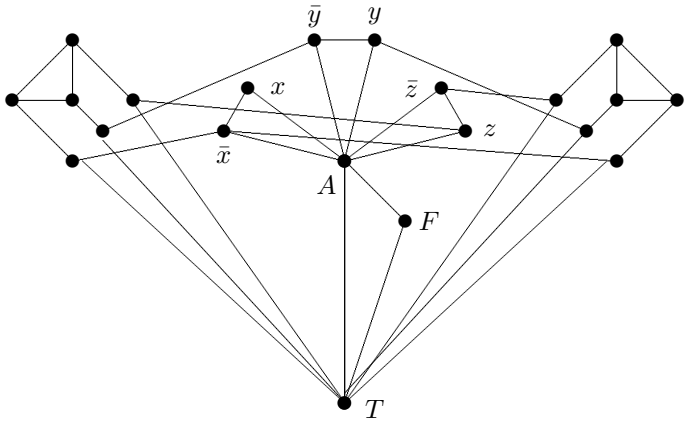
Potrebno je dodati uslov, koji bi obezbeđivao da u svakoj klauzi bar jedan literal ima vrednost 1. To se može obezbediti na sledeći način. Pretpostavimo, na primer, da se radi o klauzi $(x + y + z)$. Ovde se x, y, z mogu smatrati literalima, tj. nedostatak negacija nad simbolima x, y, z ne smanjuje opštost razmatranja. Za ovu klauzu uvodimo šest novih čvorova i povezujemo ih sa postojećim čvorovima na način prikazan na slici 9. Nazovimo tri nova čvora povezana sa T i x, y ili z spoljašnjim čvorovima (označeni su sa O na slici), a tri nova čvora u trouglu — unutrašnjim čvorovima (označeni su sa I na slici). Tvrdimo da ova konstrukcija obezbeđuje (ako je moguće bojenje sa tri boje) da bar jedan od čvorova x, y ili z mora biti obojen bojom T. Ni jedan od čvorova x, y, z ne može biti obojen bojom A, jer su svi oni povezani sa čvorom A (videti sliku 8). Ako bi sva tri čvora x, y, z bili obojeni bojom F, onda bi tri nova spoljašnja čvora povezana sa njima morali biti obojeni bojom A, pa se unutrašnji trougao ne bi mogao obojiti sa tri boje! Kompletan graf koji odgovara izrazu $(\bar{x} + y + \bar{z}) \cdot (\bar{x} + \bar{y} + z)$ prikazan je na slici 10.

Sada možemo da kompletiramo dokaz, i to u dva smera: (1) ako je izraz E zadovoljiv, G se može obojiti sa tri boje, i (2) ako se G može obojiti sa tri boje, onda je izraz E zadovoljiv.

- Ako je izraz E zadovoljiv, onda postoji takvo dodeljivanje vrednosti promenljivim, da u svakoj klauzi bar jedan literal ima vrednost 1. Obojimo čvorove grafa u skladu sa njihovim vrednostima (sa T ako je vrednost 1, odnosno sa F u protivnom). Trougao M obojen je bojama T, F i A na opisani način. U podgrafu koji odgovara klauzi bar jedan literal ima vrednost 1; odgovarajući spoljašnji čvor bojimo sa F, a ostale spoljašnje čvorove sa A, posle čega je bojenje unutrašnjih čvorova lako izvesti. Prema tome, G se može obojiti sa tri boje.
- Obrnuto, ako se G može obojiti sa tri boje, nazovimo boje u skladu sa bojenjem trougla M (koji mora biti obojen sa tri boje). Zbog trouglova



Slika 9: Podgrafovi koji odgovaraju klauzama u redukciji 3SAT na 3-obojivost.



Slika 10: Graf koji odgovara izrazu $(\bar{x} + y + \bar{z}) \cdot (\bar{x} + \bar{y} + z)$ u redukciji 3SAT na 3-obojivost.

na slici 8, boje promenljivih omogućuju neprotivrečno dodeljivanje vrednosti promenljivim. Konstrukcija bloka sa slike 9 obezbeđuje da je bar jedan literal u svakoj klauzi obojen sa T.

Konačno, jasno je da se graf G može konstruisati za polinomijalno vreme, čime je dokaz završen. □

Opšta zapažanja

Istaknimo nekoliko zapažanja u vezi dokazivanja NP-kompletnosti nekog problema Q. Prvi uslov – da problem Q pripada klasi NP se obično lako dokazuje. Posle toga treba izabrati neki problem za koji se zna da je NP-kompletan, a za koji izgleda da je povezan, ili sličan sa Q. Teško je definisati ovu "sličnost", jer ponekad izabrani problem i Q izgledaju jako različito (na primer, problem klika i SAT). Izbor pravog problema koji će biti sveden na Q je ponekad izuzetno težak, i zahteva veliko iskustvo. Mora se pokušati sa nekoliko redukcija, dok se ne dođe do pogodnog problema.

Važno je da se izvede redukcija na Q, polazeći od *proizvoljnog ulaza* poznatog NP-kompletnog problema. Najčešća greška u ovakvim dokazima je izvođenje redukcije u obrnutom smeru.

Postoji više "stepeni slobode" koji se mogu koristiti pri redukciji. Na primer, ako Q sadrži neki parametar, onda se njegova vrednost može fiksirati na proizvoljan način (suprotno od parametara u problemu koji se svodi na Q, koji se ne smeju fiksirati!). Pošto je Q samo alat za rešavanje poznatog NP-kompletnog problema, može se iskoristiti na proizvoljan način. Pored fiksiranja parametara, mogu se koristiti restrikcije Q na specijalne slučajeve dobijene i na druge načine. Na primer, mogu se koristiti samo neki tipovi ulaza za Q (ako se radi o grafovima — bipartitni grafovi, stabla i slično). Drugi važan izvor fleksibilnosti je činjenica da je efikasnost redukcije nebitna — dovoljno je da se redukcija može izvesti za polinomijalno vreme. Mogu se ignorisati ne samo konstante, tako što bi se, na primer, duplirala veličina problema: isto tako može se i kvadrirati veličina problema! Može se uvesti polinomijalno mnogo novih promenljivih, može se zameniti svaki čvor u grafu novim velikim grafom i slično. Ne postoji potreba za efikasnošću (sve dok se ostaje u granicama polinomijalnog), jer svrha redukcije nije da se transformiše u algoritam.

Postoje neke uobičajene tehnike za dobijanje redukcija. Najjednostavnija je dokazati da je poznati NP-kompletan problem specijalan slučaj problema Q. Ako je tako, dokaz je direktan, jer je rešavanje Q istovremeno i rešavanje poznatog NP-kompletnog problema. Posmatrajmo, na primer, problem *pokrivanje skupova*. Ulaz je kolekcija podskupova $S_1, S_2, \ldots, S_n$ zadatog skupa U i prirodni broj k. Problem je ustanoviti da li postoji podskup $W \subseteq U$ sa najviše k elemenata, koji sadrži bar po jedan elemenat svakog od podskupova S_i. Zapažamo da je problem pokrivač grana specijalni slučaj problema pokrivanja skupova u kome U odgovara skupu čvorova V, a svaki skup S_i odgovara jednoj grani i sadrži dva

čvora kojima je ta grana susedna. Prema tome, ako znamo da rešimo problem pokrivanja skupova za proizvoljne skupove, onda možemo da rešimo i problem pokrivač grana.

Moramo, međutim, biti pažljivi kad koristimo ovaj pristup. U opštem slučaju nije tačno da dodavanje novih zahteva čini problem težim. Posmatrajmo problem pokrivač grana. Pretpostavimo da smo dodali ograničenje da pokrivač grana ne sme da sadrži dva susedna čvora. Drugim rečima, tražimo mali skup čvorova koji je istovremeno i pokrivač grana i nezavisan skup (nezavisan skup je podskup čvorova grafa, takav da između njegova dva proizvoljna elementa ne postoji grana). Ovaj problem je na prvi pogled teži i od problema pokrivač grana i od problema nezavisan skup, jer treba brinuti o više zahteva. Ispostavlja se, međutim, da je ovo lakši problem, i da se može rešiti za polinomijalno vreme. Razlog ovoj pojavi je u tome što dopunski zahtevi toliko sužavaju familiju podskupova kandidata, da se minimum lako nalazi.

Najkomplikovanija tehnika je upotreba sastavnih elemenata – blokova, kao što je to učinjeno pri dokazu NP-kompletnosti problema 3-obojivost. Blokovi obično zavise jedan od drugog, pa je njihovo nezavisno projektovanje neizvodljivo. Moraju se imati na umu svi ciljevi problema, da bi se moglo koordinirati projektovanje različitih blokova.

Još neki NP-kompletni problemi

Sledeći spisak sadrži još nekoliko NP-kompletnih problema, koji mogu biti korisni kao polazna osnova za nove redukcije. Pronalaženje pravog problema za redukciju obično je više od polovine dokaza NP-kompletnosti.

Hamiltonov ciklus: Hamiltonov ciklus u grafu je prost ciklus koji sadrži svaki čvor grafa tačno jednom. Problem je ustanoviti da li zadati graf sadrži Hamiltonov ciklus. Problem je NP-kompletan i za neusmerene i za usmerene grafove. (Redukcija problema pokrivač grana.)

Hamiltonov put: Hamiltonov put u grafu je prosti put koji sadrži svaki čvor grafa tačno jednom. Problem je ustanoviti da li zadati graf sadrži Hamiltonov put. Problem je NP-kompletan i za neusmerene i usmerene grafove. (Redukcija problema pokrivač grana.)

Problem trgovačkog putnika: Neka je zadat težinski kompletan graf G i broj W. Ustanoviti da li u G postoji Hamiltonov ciklus sa zbirom težina grana $\leq W$. (Direktna redukcija problema Hamiltonov ciklus.)

Nezavisan skup: Nezavisan skup u grafu je podskup čvorova grafa, takav da između njegova dva proizvoljna elementa ne postoji grana. Ako je zadat graf G i prirodni broj k, ustanoviti da li G sadrži nezavisni skup sa bar k čvorova. (Direktna redukcija problema klika.)

3-dimenzionalno uparivanje: Neka su X, Y i Z disjunktni skupovi od po k elemenata, i neka je M zadati skup trojki (x, y, z) takvih da je $x \in X$, $y \in Y$

i $z \in Z$. Problem je ustanoviti da li postoji takav podskup skupa M koji svaki elemenat sadrži tačno jednom. Odgovarajući dvodimenzionalni problem uparivanja je običan problem bipartitnog uparivanja. (Redukcija problema 3SAT.)

Particija: Ulaz je skup X čijem je svakom elementu x pridružena njegova veličina $s(x)$. Problem je ustanoviti da li je moguće podeliti skup na dva disjunktna podskupa sa jednakim sumama veličina. (Redukcija problema 3-dimenzionalno uparivanje.)

Primetimo da se ovaj i sledeći problem mogu efikasno rešiti algoritmom *Ranac*, ako su veličine mali celi brojevi. Međutim, pošto je veličina ulaza srazmerna broju bita potrebnih da se predstavi ulaz, ovakvi algoritmi (koji se zovu *pseudopolinomijalni algoritmi*) su ustvari eksponencijalni u odnosu na veličinu ulaza.

Problem ranca: Ulaz su dva broja S, V i skup X čijem je svakom elementu x pridružena veličina $s(x)$ i vrednost $v(x)$. Problem je ustanoviti da li postoji podskup $B \subseteq X$ sa ukupnom veličinom $\leq S$ i ukupnom vrednošću $\geq V$. (Redukcija problema particije.)

Problem pakovanja: Ulaz je niz brojeva $a_1, a_2, \ldots, a_n$ i dva broja b, k. Problem je ustanoviti da li se ovaj skup može razložiti u k podskupova, tako da je suma brojeva u svakom podskupu $\leq b$. (Redukcija problema particije.)

Tehnike za rad sa NP-kompletnim problemima

Pojam NP-kompletnosti je osnova za teoriju koja omogućuje prepoznavanje problema za koje najverovatnije ne postoji polinomijalni algoritam. Međutim, dokazivanjem da je problem NP-kompletan, sam problem nije eliminisan: i dalje je potrebno rešiti ga. Tehnike za rešavanje NP-kompletnih problema su ponekad drugačije od tehnika koje smo do sada razmatrali. Ni jedan NP-kompletan problem se (najverovatnije) ne može rešiti tačno i kompletno algoritmom polinomijalne vremenske složenosti. Zbog toga smo prinuđeni na kompromise. Najčešći kompromisi odnose se na optimalnost, garantovanu efikasnost, ili kompletnost rešenja. Postoje i druge alternative, od kojih svaka ponešto žrtvuje. Isti algoritam se može koristiti u različitim situacijama, primenjujući različite kompromise.

Algoritam koji ne daje uvek optimalan (ili tačan) rezultat zove se *približan algoritam* (eng. approximate algorithm). Posebno su važni približni algoritmi koji mogu da garantuju koji je maksimalan stepen odstupanja od tačnog rešenja.

Drugi kompromis mogući je u vezi sa zahtevom da u najgorem slučaju vreme izvršavanja bude polinomijalno. Može se pokušati sa rešavanjem NP-kompletnih problema za polinomijalno srednje vreme. Algoritmi predviđeni za pojedine tipove slučajnih ulaza mogu da budu korisni ako je stvarna raspodela verovatnoća ulaza u skladu sa pretpostavljenom. Međutim, obično je nalaženje tačne raspodele vrlo teško. Najteži deo posla pri konstrukciji algoritama, koji u proseku dobro rade, je najčešće njihova analiza.

Konačno, mogu se praviti kompromisi u vezi sa kompletnošću algoritama; naime, može se dozvoliti da algoritam radi efikasno samo za neke specijalne ulaze. Na primer, problem pokrivač grana može se rešiti za polinomijalno vreme za bipartitne grafove. Prema tome, kad se formuliše apstraktni problem polazeći od situacije iz realnog života, treba obezbediti da svi dopunski uslovi koje ulaz zadovoljava budu uključeni u apstraktnu definiciju. Drugi primer su algoritmi sa eksponencijalnom vremenskom složenošću, koji se ipak mogu izvršavati za male ulaze, što je često potpuno zadovoljavajuće.

Približni algoritmi sa garantovanim kvalitetom rešenja

U ovom odeljku razmotrićemo približne algoritme za nekoliko NP-kompletnih problema: to su problem pokrivač grana, problem jednodimenzionog pakovanja, euklidski problem trgovačkog putnika i klasterovanje sa k centara. Svi razmatrani algoritmi imaju garantovani kvalitet rešenja; drugim rečima, može se dokazati da rešenja koja oni daju nisu predaleko od optimalnih rešenja.

Pokrivač grana

Podsetimo se problema pokrivač grana u grafu: on se sastoji u određivanju minimalnog podskupa skupa čvorova za koji važi da je svaka grana grafa susedna bar jednom od čvorova tog skupa.

Razmotrimo jednostavan približni algoritam za nalaženje minimalnog pokrivača grana datog grafa. Za zadati neusmereni graf $G = (V, E)$ uparivanje je skup disjunktnih grana (grana bez zajedničkih čvorova). Maksimalno uparivanje je uparivanje koje se ne može proširiti dodavanjem nove grane. Maksimalno uparivanje može se naći prostim dodavanjem grana sve dotle dok je to moguće (odnosno dok postoje neka dva neuparena, a susedna čvora).

Neka je G graf, i neka je M maksimalno uparivanje u G. Pošto je M uparivanje, njegove grane nemaju zajedničkih tačaka, a pošto je M maksimalno uparivanje, sve ostale grane imaju bar jedan zajednički čvor sa nekom granom iz M.

Teorema 9. *Skup čvorova susednih granama maksimalnog uparivanja M je pokrivač grana, sa najviše dva puta više čvorova nego što ih ima minimalni pokrivač.*

Dokaz: Skup krajeva grana iz M je pokrivač grana, jer je M maksimalno uparivanje (ako bi postojala nepokrivena grana, ta grana mogla bi da se uključi u uparivanje, što je nemoguće, jer je pretpostavljeno da je uparivanje maksimalno). Označimo skup krajnjih čvorova grana uparivanja M sa P. Svaki pokrivač grana, pa i minimalni (označimo ga sa P^*), mora da pokrije sve grane — specijalno, sve grane uparivanja M. Pošto je M uparivanje, bilo koji čvor iz P može da pokrije najviše jednu granu uparivanja M. Prema tome, bar jedan kraj svake grane iz M mora da pripada minimalnom

pokrivaču grana, tj. važi:

$$|P^*| \geq \frac{|P|}{2}$$

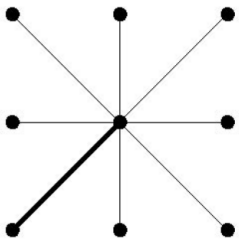
odnosno:

$$|P| \leq 2|P^*|$$

□

Primetimo da je računanje proizvoljnog maksimalnog uparivanja u grafu polinomijalne vremenske složenosti, te ćemo za ovaj algoritam reći da je polinomijalni 2-približni algoritam za računanje minimalnog pokrivača grana grafa, odnosno da je njegov *faktor aproksimacije* jednak 2.

Primetimo da je granica data u prethodnoj teoremi tesna. Razmotrimo graf sa slike 11: minimalni pokrivač sadrži samo jedan, središnji čvor u grafu, dok će predloženi algoritam vratiti dvočlan pokrivač koji odgovara proizvoljnoj grani u grafu. Dakle, dobijeni pokrivač ima tačno dva puta više čvorova nego minimalni.



Slika 11: Ilustracija grafa kod koga pokrivač grana koji daje približni algoritam ima tačno dva puta više čvorova od minimalnog pokrivača grana.

Jednodimenzionalno pakovanje

Problem pakovanja (eng. bin packing) odnosi se na pakovanje objekata različitih veličina u kutije tako da se iskoristi najmanji mogući broj kutija. Na primer, prilikom selidbe je cilj preneti sve stvari, koristeći kamion najmanji mogući broj puta, pakujući stvari što je moguće bolje. Ovo je, naravno, trodimenzionalni problem. Mi ćemo se u ovom materijalu pozabaviti *jednodimenzionim problemom pakovanja*. Zbog jednostavnosti se pretpostavlja da sve kutije imaju veličinu 1.

Problem. Neka je $x_1, x_2, \ldots, x_n$ skup realnih brojeva između 0 i 1. Podeliti ih u najmanji mogući broj podskupova, tako da suma brojeva u svakom podskupu bude najviše 1.

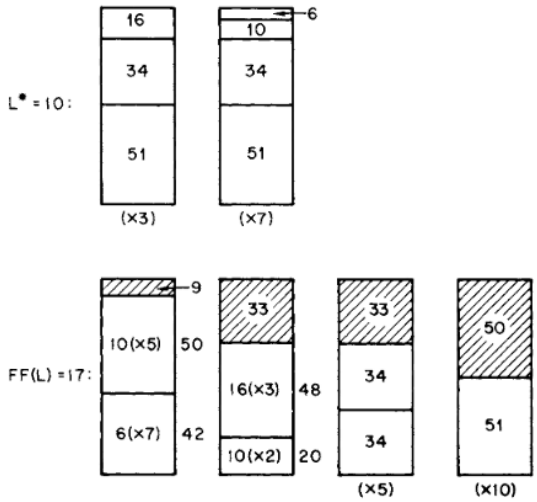
Na jednodimenzionalni problem pakovanja nailazi se, na primer, u problemima upravljanja memorijom, kad postoje zahtevi za dodelom memorijskih blokova različite veličine, a dodeljivanje se vrši iz nekoliko jednakih velikih blokova memorije. Jednodimenzionalni problem pakovanja je NP-kompletan.

Jedan od mogućih heurističkih algoritama za rešavanje ovog problema je staviti x_1 u prvu kutiju, a zatim, za svako i, staviti x_i u prvu kutiju u kojoj ima dovoljno mesta, ili započeti sa novom kutijom, ako nema dovoljno mesta ni u jednoj od korišćenih kutija. Ovaj algoritam zove se *prvi odgovarajući* (eng. first fit) i, kao što pokazuje sledeća teorema, dovoljno je dobar u najgorem slučaju.

Teorema 10. *Algoritam prvi odgovarajući zahteva najviše* $2m$ *kutija, gde je* m *najmanji mogući broj kutija.*

Dokaz: Po završetku algoritma prvi odgovarajući ne postoje dve kutije sa iskorišćenjem manjim od $1/2$. Prema tome, ako sa k označimo broj upotrebljenih kutija, biće $m \geq \sum_{i=1}^{n} x_i > (k-1)/2$, odakle je $k < 2m+1$, odnosno $k \leq 2m$. □

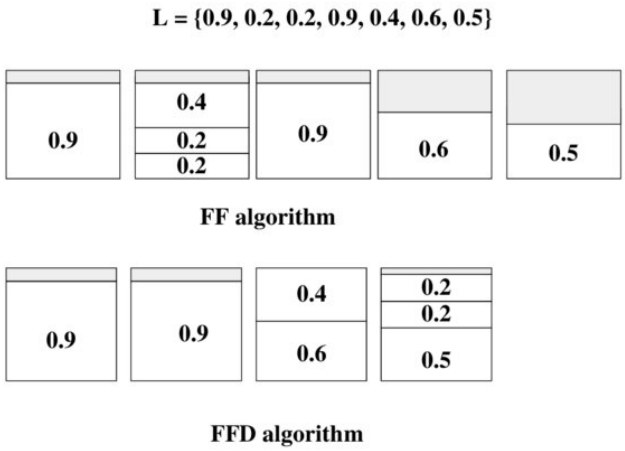
Ispostavlja se da je granica definisana ovom teoremom prilično gruba. Konstanta 2 iz Teoreme 10 može se smanjiti na 1.7 posle nešto komplikovanije analize. Konstanta 1.7 ne može se dalje smanjiti, jer postoje primeri u kojima algoritam prvi odgovarajući zahteva tačno 1.7 puta više kutija od optimalnog algoritma. Na slici 12 prikazan je problem kada su kapaciteti kutija 101, a potrebno je u njih smestiti 7 predmeta od 6, 7 predmeta od 10, 3 predmeta od 16, 10 predmeta od 34 i 10 predmeta od 51 – pokazuje se da je minimalni potreban broj kutija za pakovanje ovih predmeta 10, a da algoritam prvi odgovarajući koristi 17 kutija, tj. tačno 1.7 puta više od minimalnog broja kutija.



Slika 12: Primer kada algoritam prvi odgovarajući koristi 17 kutija, a minimalni potrebni broj kutija je 10.

Opisani algoritam može se jednostavno poboljšati. Do najgoreg slučaja dolazi kad se mnogo malih brojeva pojavljuje na početku. Umesto da se brojevi pakuju u

redu kojim nailaze, oni se najpre sortiraju u nerastući niz. Promenjeni algoritam zove se *opadajući prvi odgovarajući* (eng. first fit decreasing), i u najgorem slučaju troši najviše oko 1.22 puta više kutija od optimalnog algoritma (teoremu dajemo bez dokaza).



Slika 13: Ilustracija dve varijante jednodimenzionog problema pakovanja.

Teorema 11. *Algoritam opadajući prvi odgovarajući zahteva najviše* $\frac{11}{9}m + 4$ *kutija, gde je* m *najmanji mogući broj kutija.*

Konstanta 11/9 je takođe najbolja moguća. Prvi odgovarajući i opadajući prvi odgovarajući su jednostavne heuristike i ilustrovane su na slici 13. Postoje drugi metodi koji garantuju još manje konstante. Njihova analiza je u većini slučajeva komplikovana.

Strategije koje smo opisali tipične su za heurističke algoritme. One odražavaju prirodne pristupe, odnosno odgovaraju načinu na koji bi neko ručno rešavao ove probleme. Međutim, videli smo mnogo slučajeva u kojima se direktni pristupi ponašaju loše za velike ulaze. Zbog toga je veoma važno analizirati ponašanje takvih algoritama.

Euklidski problem trgovačkog putnika

Problem trgovačkog putnika je važan problem sa mnogo primena. Razmotrićemo ovde varijantu TSP sa dopunskim ograničenjem da težine grana odgovaraju euklidskim rastojanjima.

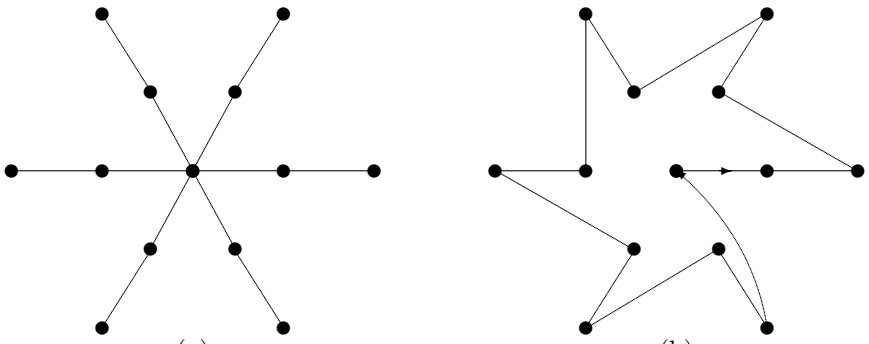
Problem. Neka je $C_1, C_2, \ldots, C_n$ skup tačaka u ravni koje odgovaraju položajima n gradova. Pronaći Hamiltonov ciklus minimalne dužine (maršrutu trgovačkog putnika) među njima.

Problem je i dalje NP-kompletan (ovo tvrđenje navodimo bez dokaza), ali videćemo da pretpostavka da su rastojanja euklidska omogućuje konstrukciju približnog algoritma za njegovo rešavanje. Preciznije, ova pretpostavka može se uopštiti, zamenjujući je pretpostavkom da rastojanja zadovoljavaju nejednakost trougla, odnosno da je rastojanje između dva čvora uvek manje ili jednako od dužine proizvoljnog puta između njih preko ostalih čvorova.

Najpre se konstruiše minimalno povezujuće stablo grafa MCST, pri čemu su cene grana njihove dužine. Tvrdimo da cena stabla nije veća od dužine najboljeg ciklusa TSP. Zaista, ciklus TSP sadrži sve čvorove, pa ga uklanjanje jedne grane čini povezujućim stablom, čija je cena veća ili jednaka od cene MCST.

Od povezujućeg stabla se, međutim, ne dobija direktno ciklus TSP. Posmatrajmo najpre ciklus koji se dobija pretragom u dubinu ovog stabla (ne celog grafa) polazeći od proizvoljnog čvora, i sadrži granu u obrnutom smeru uvek kad se grana prilikom vraćanja prolazi u suprotnom smeru (ovaj ciklus odgovara, na primer, obilasku galerije u obliku stabla, sa slikama na oba zida svakog hodnika, idući uvek udesno). Svaka grana se tako prolazi tačno dva puta, pa je cena ovog ciklusa najviše dva puta veća od cene MCST. Na kraju se ovaj ciklus prepravlja u ciklus TSP, idući prečicom uvek kad treba proći kroz granu već uključenu u ciklus (slika 14). Drugim rečima, umesto povratka starom granom, idemo direktno do prvog nepregledanog čvora. Pretpostavka da su rastojanja euklidska je važna, jer obezbeđuje da je uvek direktni put između dva grada bar toliko dobar, koliko bilo koji zaobilazni put.

Teorema 12. *Dužina dobijenog ciklusa* TSP *je manja od dvostruke dužine minimalnog ciklusa* TSP*.*

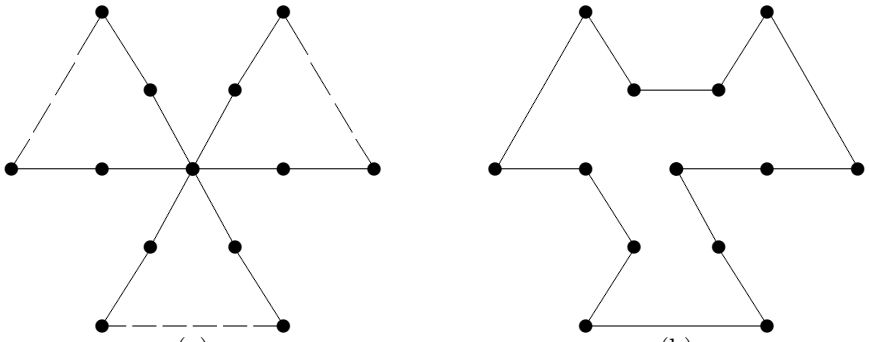


Slika 14: (a) Minimalno povezujuće stablo (MCST) (b) ciklus TSP dobijen od MCST polazeći od srednjeg čvora i idući uvek udesno.

Vremenska složenost ovog algoritma određena je brojem koraka u algoritmu za konstrukciju MCST, što je svakako polinomijalna složenost.

Poboljšanje Algoritam koji smo upravo opisali može se poboljšati na sledeći način. Njegov "najgrublji" deo je pretvaranje obilaska stabla u ciklus TSP. Ta konverzija može se posmatrati i na drugi način: ona formira Ojlerov ciklus od stabla u kome je svaka grana udvostručena. Posle toga se konstruiše ciklus TSP korišćenjem prečica u Ojlerovom ciklusu. Konverzija stabla u Ojlerov ciklus može se izvesti mnogo racionalnije. Ojlerov graf može da sadrži samo čvorove parnog stepena. Posmatrajmo sve čvorove neparnog stepena u stablu. Broj takvih čvorova mora biti paran (u protivnom bi suma stepena čvorova bila neparna, što je nemoguće, jer je ta suma jednaka dvostrukom broju grana). Ojlerov graf se od stabla može dobiti dodavanjem dovoljnog broja grana, čime se postiže da stepeni svih čvorova postanu parni. Pošto se ciklus TSP sastoji od Ojlerovog ciklusa (sa nekim prečicama), voleli bismo da minimiziramo zbir dužina dodatih grana. Razmotrimo sada taj problem.

Dato je stablo u ravni, a cilj je dodati mu neke grane, sa minimalnom sumom dužina, tako da dobijeni graf bude Ojlerov. Svakom čvoru neparnog stepena mora se dodati bar jedna grana. Pokušajmo da postignemo cilj dodavanjem tačno jedne grane svakom takvom čvoru. Pretpostavimo da ima $2k$ čvorova neparnog stepena. Ako dodamo k grana, tako da svaka od njih spaja dva čvora neparnog stepena, onda će stepeni svih čvorova postati parni. Problem tako postaje problem uparivanja. Potrebno je pronaći uparivanje minimalne dužine koje pokriva sve čvorove neparnog stepena. Nalaženje optimalnog uparivanja može se izvesti za $O(n^3)$ koraka za proizvoljni graf (ovo tvrđenje navodimo bez dokaza). Konačni ciklus TSP se zatim dobija od Ojlerovog grafa (koji obuhvata minimalno povezujuće stablo i uparivanje minimalne dužine) korišćenjem prečica. Ciklus TSP dobijen ovim algoritmom od stabla sa slike 14 prikazan je na slici 15.

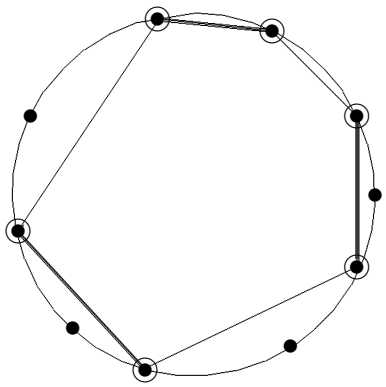


Slika 15: Minimalni Ojlerov ciklus i odgovarajući ciklus TSP (a) minimalno povezujuće stablo sa uparivanjem, (b) ciklus TSP dobijen od Ojlerovog ciklusa.

TSP dobijen od Ojlerovog ciklusa.} \end{figure}

Teorema 13. *Poboljšani algoritam daje ciklus* TSP *čija je dužina najviše* 1.5 *puta veća od dužine minimalnog ciklusa* TSP.

Dokaz: Potrebno je oceniti dužinu Ojlerovog ciklusa, jer se dužina ciklusa posle uvođenja eventualnih prečica može samo smanjiti. Ojlerov ciklus se sastoji od stabla i uparivanja. Označimo sa Q minimalni ciklus TSP, a sa $|Q|$ njegovu dužinu. Videli smo već da je dužina stabla manja ili jednaka od $|Q|$; prema tome, dovoljno je dokazati da dužina uparivanja ne prelazi $|Q|/2$. Ciklus Q sadrži sve čvorove. Neka je D skup čvorova neparnog stepena u polaznom stablu. Za skup D mogu se formirati dva disjunktna uparivanja sa zbirom dužina $\leq |Q|$ na sledeći način (videti sliku 16, na kojoj su zaokruženi čvorovi iz D). Započinjemo sa proizvoljnim čvorom $v \in D$ i uparujemo ga sa sa najbližim čvorom (u smeru kazaljke na satu) duž ciklusa Q. Posle toga nastavlja se sa uparivanjem u istom smeru. Ako upareni čvorovi nisu susedi u Q, onda je rastojanje između njih manje ili jednako od dužine puta koji ih povezuje u Q (zbog nejednakosti trougla). Ovaj proces daje jedno uparivanje. Drugo uparivanje dobija se ponavljanjem istog procesa u smeru suprotnom od kazaljke na satu. Suma dužina oba uparivanja je najviše $|Q|$, videti sliku 16. Međutim, pošto je M uparivanje minimalne težine u D, njegova dužina je manja ili jednaka od dužine boljeg od dva spomenuta uparivanja (sa zbirom $|Q|$), pa je dakle manja ili jednaka od $|Q|/2$. □



Slika 16: Dva uparivanja čija suma dužina ne prelazi dužinu minimalnog ciklusa TSP.

Nalaženje uparivanja minimalne težine znatno je složenije od nalaženja minimalnog povezujućeg stabla, ali se time dobija bolja granica. Navedeni primer ilustruje osnovnu karakteristiku ovog tipa algoritama: uopštava se lakši problem — ili se oslabljuju (relaksiraju) neki elementi polaznog problema — i tako se formira heuristika.

Opšti problem trgovačkog putnika

Kao što smo videli, postoje približni algoritmi polinomijalne složenosti za rešavanje euklidskog problema trgovačkog putnika, takvi da pronalaze Hamiltonov ciklus dužine najviše $\rho \cdot opt$, gde je ρ jednako 2 ili $3/2$, a opt je minimalna dužina Hamiltonovog ciklusa. Pokazuje se da u opštem slučaju za rešavanje problema trgovačkog putnika ne postoji približni algoritam polinomijalne složenosti za bilo koje $\rho > 1$, osim ako nije $P = NP$. Ovo tvrđenje predmet je teoreme 14.

Teorema 14 (Nepostojanje približnog algoritma za rešavanje TSP)**.** *Ako je $P \neq NP$, onda za bilo koju konstantu $\rho \geq 1$ važi da ne postoji približni algoritam polinomijalne vremenske složenosti za rešavanje (opšteg) problema trgovačkog putnika u polinomijalnoj vremenskoj složenosti sa aproksimacijom do na faktor ρ (odnosno, da je ciklus koji približni algoritam vraća najviše ρ puta duži od dužine optimalnog Hamiltonovog ciklusa).*

Dokaz: Dokaz ćemo izvesti izvođenjem kontradikcije. Pretpostavimo suprotno, odnosno, da za neki broj $\rho \geq 1$, postoji približni algoritam A polinomijalne vremenske složenosti sa aproksimacijom do na faktor ρ. Bez gubitka na opštosti, pretpostavimo da je ρ celobrojna vrednost, uz zaokruživanje ukoliko je to neophodno. Sada ćemo pokazati kako je moguće iskoristiti algoritam A za rešavanje instanci problema Hamiltonovog ciklusa algoritmom polinomijalne vremenske složenosti. Kako znamo da je problem pronalaženja Hamiltonovog ciklusa NP-kompletan problem, sledi da ako možemo da ga rešimo algoritmom polinomijalne vremenske složenosti, onda je $P = NP$.

Neka je $G = (V, E)$ zadati graf – instanca problema pronalaženja Hamiltonovog ciklusa; cilj je utvrditi da li G sadrži Hamiltonov ciklus koristeći hipotetički približni algoritam A. Polazeći od grafa G može se formirati instanca problema TSP na sledeći način. Neka je $G' = (V, E')$ kompletan graf sa skupom čvorova V, odnosno,

$$E' = \{(u, v) \mid u, v \in V \text{ i } u \neq v\}.$$

Pridružujemo cenu svakoj grani iz E' na sledeći način:

$$c(u, v) = \begin{cases} 1, & \text{ako } (u, v) \in E, \\ \rho|V| + 1, & \text{inače.} \end{cases}$$

Jasno je da je složenost formiranja težinskog grafa G' od polaznog grafa G polinom od $|V|$ i $|E|$.

Razmotrimo problem trgovačkog putnika (G', c). Ako početni graf G ima Hamiltonov ciklus H, onda funkcija cene c pridružuje svakoj grani iz H cenu 1, odakle se dobija da (G', c) sadrži Hamiltonov ciklus cene $|V|$. Sa druge strane, ako G ne sadrži Hamiltonov ciklus, onda bilo koji Hamiltonov ciklus u G' mora da sadrži neku granu koja se ne nalazi u E. Međutim,

proizvoljan takav ciklus koji sadrži neku granu koja se ne nalazi u E ima cenu barem

$$\begin{aligned}(\rho|V|+1)+(|V|-1) &= \rho|V|+|V| \\ &= (\rho+1)|V|.\end{aligned}$$

Dakle, cena Hamiltonovog ciklusa u G' je barem za faktor $\rho+1$ veća od cene Hamiltonovog ciklusa u G' koji jeste Hamiltonov ciklus u G.

Primenimo približni algoritam A polinomijalne složenosti (za koji smo pretpostavili da postoji) na instancu (G', c) problema TSP. S obzirom da algoritam A daje kao rezultat Hamiltonov ciklus cene ne veće od ρ puta cene optimalnog puta, ako G sadrži Hamiltonov ciklus, onda A mora da ga vrati. Ako G nema Hamiltonov ciklus, onda A pronalazi u G' Hamiltonov ciklus dužine bar $(\rho+1)V > \rho V$. Dakle, algoritam A polinomijalne složenosti može se iskoristiti za utvrđivanje da li u G postoji Hamiltonov ciklus. Iz ovog zaključka sledi da je $P = NP$, suprotno pretpostavci teoreme. □

Klasterovanje sa k centara

Klasterovanje je važan problem sa različitim primenama u oblastima poput statistike i mašinskog učenja.

Neformalno klasterovanje sa k centara podrazumeva particionisanje potencijalno velikog skupa tačaka P u mali broj podskupova čiji su elementi međusobno blizu. Rezultat problema klasterovanja je skup $C = \{c_1, c_2, \ldots, c_k\}$ centara klastera, pri čemu su tačke c_i iz skupa P.

Problem. Dat je skup P od n tačaka u prostoru. Za svake dve tačke u, v is skupa P neka $d(u, v)$ označava rastojanje između tačaka u i v, pri čemu funkcija rastojanja zadovoljava svojstva metrike (nenegativnost, simetričnost i nejednakost trougla). Za dati ceo broj $k \leq n$ potrebno je odrediti podskup C skupa P od k tačaka, koje nazivamo *centrima klastera*, kojim se minimizuje maksimalno rastojanje proizvoljne tačke skupa P do njemu najbližeg elementa iz C.

Rastojanje tačke i skupa P do njenog centra označićemo sa:

$$d(i, C) = \min_{c \in C} d(i, c)$$

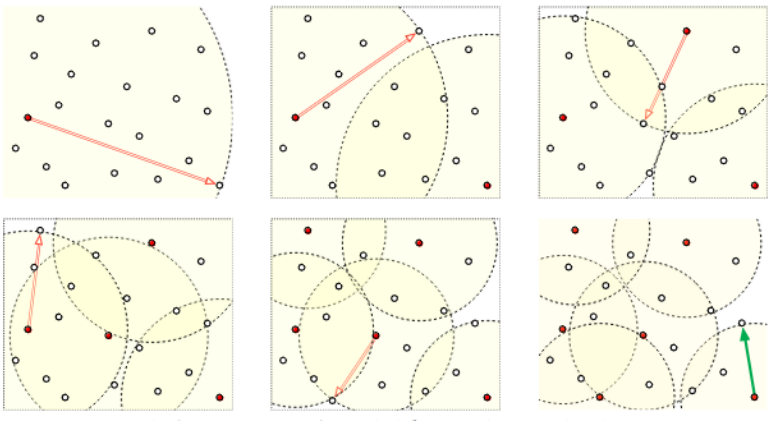
Za dati skup C centara klastera, radijus skupa C označićemo sa:

$$r = \max_{i \in P} d(i, C)$$

Zašto je dovoljno odrediti samo centre ovih klastera, a ne i same klastere? Naime, funkcija cilja nam omogućava da efikasno rekonstruišemo klastere: za svaku

tačku i prolazimo kroz k centara klastera kako bismo pronašli najbliži među njima i tačku i pridružujemo particiji sa datim centrom.

Pokazuje se da je ovaj problem NP-kompletan. Ipak, postoji prirodna pohlepna strategija, koja daje rešenje dobrog kvaliteta. Na početku na proizvoljan način biramo tačku $i \in P$ i dodajemo je skupu C centara klastera. Nakon toga za naredni centar klastera biramo tačku što dalje od prvog centra klastera. Na isti način nastavljamo i u narednim koracima algoritma: sve dok je $|C| < k$ tražimo tačku $j \in P$ za koju je vrednost rastojanja $d(j, C)$ maksimalna i dodajemo je skupu C. Kada $|C|$ dostigne vrednost k stajemo (slika 17). Ovaj algoritam je prvi predložio Teofilo Gonzalez 1985. godine.



Slika 17: Ilustracija prvih šest koraka u Gonzalezovom algoritmu.

Postavlja se pitanje kojeg je kvaliteta dobijeno rešenje. Lako se pokazuje da algoritam u opštem slučaju ne vraća optimalno rešenje, ali bismo voleli da imamo neku garanciju kvaliteta.

Teorema 15. *Predloženim pohlepnim algoritmom dobija se skup čiji je radijus najviše dva puta veći od optimalnog.*

Dokaz: Označimo sa $C^* = \{c_1, \ldots, c_k\}$ optimalno rešenje problema k-klasterovanja i neka je r^* njegov radijus. Neka optimalno rešenje particioniše tačke skupa P u klastere $V_1^*, \ldots, V_k^*$, pri čemu je svaka tačka $i \in P$ smeštena u klaster V_l^* ako je od svih tačaka iz C^* tački i najbliža tačka c_l.

Najpre, primetimo da za proizvoljan par tačaka i i j iz istog klastera V_l^* važi da su najviše na rastojanju $2r^*$. Naime, na osnovu nejednakosti trougla i simetričnosti relacije rastojanja, za rastojanje između tačaka i i j važi:

$$d(i, j) \leq d(i, c_l) + d(c_l, j) = d(i, c_l) + d(j, c_l) \leq 2r^*$$

Razmotrimo sada skup $C \subset P$ tačaka koje je izabrao pohlepni algoritam i razmotrimo prvu iteraciju u kojoj algoritam skupu C dodaje tačku $i \in V_l^*$ za neko l, iako je algoritam u nekoj od prethodnih iteracija već izabrao neku tačku $i' \in V_l^*$. Pre tačke i' svi centri dodati skupu C su bili iz različitih optimalnih klastera C^*. Za sve tačke j koje su pokrivene centrima klastera dodatim pre i' važi $d(j, C) \leq 2r^*$.

S obzirom na to da pohlepni algoritam uvek bira tačke koje su najudaljenije od tekućeg skupa tačaka u C, za svaku drugu tačku $j \in P$ gde j nije dodato skupu C važi:

$$d(j, C) \leq d(i', i)$$

inače bi j bilo dodato skupu C pre i'. S obzirom na to da i i i' pripadaju istom optimalnom klasteru V_l^*, važi $d(i', i) \leq 2r^*$. Stoga za svaku tačku pokrivenu nakon što je i' dodato centrima klastera važi da je njeno rastojanje od najbližeg centra manje ili jednako $2r^*$. Dakle, za sve tačke $i \in P$, rastojanje je ograničeno sa $d(i, S) \leq 2r^*$. Ako r predstavlja radijus skupa C koji je vratilo pohlepno rešenje, možemo zaključiti da je:

$$r^* \leq r \leq 2r^*,$$

odakle sledi da je radijus dobijenog skupa najviše dva puta veći od optimalnog. □

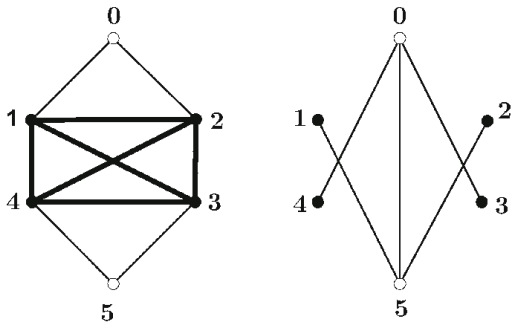
Redukcije koje čuvaju približna rešenja

Postavlja se pitanje da li možemo da prenesemo rezultate dobijene približnim algoritmom sa jednog problema na drugi. Odgovor na ovo pitanje nam daju tzv. *redukcije koje čuvaju približna rešenja*. Bilo koja od redukcija ovog tipa nameće određene uslove na koji način se odnos sa optimalnim rešenjem, odnosno količnik aproksimacije transformiše iz jednog problema u drugi.

Redukcije koje čuvaju približna rešenja predstavljaju alternativni način za postizanje novih rezultata u domenu približnih algoritama za neki težak problem. Naime, kada se pokušava sa određivanjem približnog rešenja nekog novog problema, jedan mogući način je da se sprovede detaljna studija problema od nule. Drugi mogući način je pokušati da se iskoristi poznato znanje o sličnim problemima datom problemu.

Klike

Nezavisan skup (eng. independent set) u grafu je poskup čvorova grafa, takav da između nikoja dva njegova elementa ne postoji grana. Važno pitanje je da li dati graf G sadrži nezavisan skup veličine bar k. Pokazuje se da je ovaj problem NP-kompletan. Naime, postoji redukcija sa problema klika na ovaj problem: klika veličine k u grafu G postaje nezavisan skup veličine k u grafu G' koji je komplement grafa G i obratno.



Slika 18: Redukcija sa problema klika na problem nezavisan skup.

Pretpostavimo sada da znamo približni algoritam A za problem maksimalni nezavisni skup, koji vraća skup čvorova čija se veličina od optimalnog rešenja razlikuje do na neki faktor koji je funkcija polaznog grafa. Pretpostavimo da želimo da dođemo da približnog algoritma za problem maksimalne klike u grafu G. Možemo konstruisati graf G' i na njemu pokrenuti približni algoritam A: on će vratiti nezavisan skup grafa G' koji predstavlja kliku u G. Ova klika je iste veličine kao i nezavisni skup u polaznom grafu, a s obzirom da su grafovi G i G' iste veličine, faktor aproksimacije problema nezavisan skup se takođe postiže i za problem maksimalne klike.

Važi i inverzno tvrđenje.