# Robusna diskretna optimizacija

**Bogović, Dario**

**SVEUČILIŠTE U ZAGREBU**

**PRIRODOSLOVNO–MATEMATIČKI FAKULTET**

**MATEMATIČKI ODSJEK**

Dario Bogović

# ROBUSNA DISKRETNA OPTIMIZACIJA

Diplomski rad

Voditelj rada:
prof. dr. sc. Robert Manger

Zagreb, veljača, 2023.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

# Sadržaj

# Uvod

Cilj ovog diplomskog rada je istražiti vremensku složenost robusnih varijanti nekih diskretnih problema optimizacije.

Proučavaju se robusne varijante problema najkraćeg puta, problema najmanjeg razapinjućeg stabla i problema naprtnjače. Ovi problemi su od većeg interesa jer su robusne varijante sva tri spomenuta problema NP-teški dok su konvencionalne varijante polinomne ili pseudo-polinomne vremenske složenosti.
Robusnost je jedan od načina suočavanja s nesigurnosti podataka. Do realizacije samog projekta se ne zna stvarni trošak, stoga su podaci nesigurni. Različiti troškovi i načini kako je došlo do tih troškova nazivaju se scenarijima.

U prvom poglavlju se definira pojam scenarija. Definiraju se tri kriterija robusnosti: apsolutni, devijacijski i relativni. Time je precizno definirano što znači odabir rezultata čije "najgore ponašanje" je "najbolje moguće". Uz svaki od kriterija robusnosti dan je komentar kada ga je najbolje koristiti i što on stvarno predstavlja u praktičnoj primjeni. U nastavku rada fokus je na kriteriju apsolutne robusnosti za navedene probleme.

Zatim se, u drugom poglavlju, daje kratak opis klasa P i NP. Fokus poglavlja je na teoremima i tvrdnjama potrebnim za shvaćanje dokaza u nastavku rada te se ne ulazi u širinu niti dublju tematiku vezanu za te klase.

U trećem poglavlju iskazan je problem najkraćeg puta i njegova robusna varijanta, a zatim dokazana vremenska složenost robusne varijante. U istom poglavlju prikazan je i Dijkstrin algoritam koji rješava konvencionalni (ne-robusni) problem najkraćeg puta. Prvo se dokazuje da su apsolutna i devijacijska robusna varijanta problema NP-teške i u slučajevima kada je graf oblika slojevita mreža širine dva i postoje samo dva scenarija. U trećem teoremu dan je pseudo-polinomni algoritam za rješavanje ta dva problema s konačnim brojem scenarija. Zadnja dva teorema pokazuju da su ta dva problema NP-teški za neograničen broj scenarija.

Nakon toga, u četvrtom poglavlju dokazana je vremenska složenost robusnog problema najmanjeg razapinjućeg stabla. Za postizanje tog rezultata potreban je i poznati Primov algoritam. U prvom teoremu dana su ograničenja apsolutnog robusnog problema i pokazano je da je i s tim ograničenjima problem NP-težak. U slijedećem teoremu se promatra pseudo-polinomni algoritam koji rješava apsolutnu robusnu varijantu problema, a u zadnjem teoremu dokazno je da je apsolutno-robusni problem s neograničenim brojem scenarija također NP-težak.

Peto poglavlje se odnosi na robusni problem naprtnjače i dokaz vremenske složenosti algoritma za isti. Od svih navedenih problema ovaj je od najmanjeg interesa obzirom da je njegova konvecionalna varijanta već NP-teška, ali je rješiv u pseudo-polinomnom vremenu.

U zadnjem poglavlju nalazi se implementacija algoritama koji rješava robusni problem najkraćeg puta na slojevitim mrežama s 4 ili manje scenarija. Implementacija je izrađena u jeziku Ruby (verzija 3.0.0).

# Poglavlje 1

# Kriteriji robusnosti

Kada se govori o diskretnim minimizacijskim problemima koristi se model scenarija kako bi se reprezentirala nesigurnost (odnosno neizvjesnost) ulaznih podataka u donošenju odluke. U radu [8] su scenariji opisani na slijedeći način: Svaki scenarij određuje jednu kombinaciju konkretnih vrijednosti parametara. Promatraju se samo ona rješenja koja su dopustiva za sve scenarije. Za svako rješenje promatra se njegovo ponašanje na svakom od scenarija. Bilježi se najgore ponašanje tog rješenja (na nekom od scenarija). Kao konačno robusno rješenje problema izabire se ono čije najgore ponašanje je najbolje moguće. Riječ je o min-max, odnosno max-min problemu, ovisno o tome je li polazni problem minimizacija, odnosno maksimizacija.

U skladu sa [1], skup scenarija se ozačava sa $S$, a svaki njegov element $s \in S$ predstavlja po jednu realizaciju modela (jedan konkretni slučaj koji bi se mogao dogoditi). Elemente reprezentira odgovarajući vektor $c^s = (c_1^s, c_2^s, ..., c_n^s)$ gdje je $n \in \mathbb{N}$ unaprijed određen konkretnim problemom koji se promatra (primjerice: broj bridova u grafu). Ta potencijalna realizacija mora imati neku pozitivnu vjerojatnost pojavljivanja koja ne mora biti nužno poznata

Iako su oznake slične u većini literature, u [8] su jasno opisane:

- Sa $S$ se označava skup mogućih scenarija, a njegovi se elementi (*scenarij*) označavaju sa $s$
- Sa $X$ se označava vektor s vrijednostima varijabli odlučivanja ($X$ je zapis jednog rješenja problema)
- S $D^s$ se označava kombinacija vrijednosti parametara koji odgovaraju scenariju $s$
- S $F_s$ se označava skup dopustivih rješenja (dopustivih $X$-eva u scenariju $s$)

- $f(X, D^s)$ je vrijednost funkcije cilja za rješenje $X \in F_s$ pod scenarijem $s$
- $X_s^*$ je optimalno rješenje za scenarij $s$
- $z^s$ je vrijednost funkcije cilja za optimalno rješenje scenarija $s$

$z^s$ je dan jednakostima:

$$z^s = f(X_s^*, D^s) = \min_{X \in F_s} f(X, D^s); \tag{1.1}$$

gdje se s $D$ označava skup ulaznih podataka, a $D^s$ je oznaka konkretnih podataka iz scenaraija $s$.

S definiranim skupom odluka i funkcijom evaluacije definiraju se:

**Definicija 1.0.1.** ***Apsolutno robusno rješenje*** *$X_A$ je ono koje među svim dopustivim rješenjima minimizira maksimum cijene nad svim scenarijima, odnosno:*

$$z_A = \max_{s \in S} f(X_A, D^s) = \min_{X \in \cap_{s \in S} F_S} \max_{s \in S} f(X, D^s); \tag{1.2}$$

Apsolutnom robusnosti evaluira se donesena odluka u svim scenarijima te se zatim najgori slučaj uzima kao indikator robusnosti odluke. Ovaj kriterij ima sklonost da vodi ka konzervativnim odlukama zbog usporedbe s najgorim mogućim slučajem. On nastoji zaštititi od najgoreg slučaja, a koristi se kada postizanje optimalnosti nije od velike važnosti, već je bitno da funkcija cilja ne prijeđe neku gornju granicu niti u najgorem slučaju.

**Definicija 1.0.2.** ***Robusno devijacijsko rješenje*** *$X_A$ je ono koje među svim dopustivim rješenjima minimizira maksimalni otklon od optimuma gledajući sve scenarije, odnosno:*

$$z_D = \max_{s \in S} f(X_D, D^s) - f(X_s^*, D^s) = \min_{X \in \cap_{s \in S} F_S} \max_{s \in S} (f(X, D^s) - f(X_s^*, D^s)); \tag{1.3}$$

Rješenje robusnom devijacijom polazi od pretpostavke da funkcija cilja može poprimati vrlo različite vrijednosti ovisno o scenariju. Primjenjuje se kada je važniji otklon od optimuma, a ne apsolutna vrijednost funkcije cilja, dok se rješenje evaluira a-posteriori, to jest kada se zna koji je scenarij nastupio.

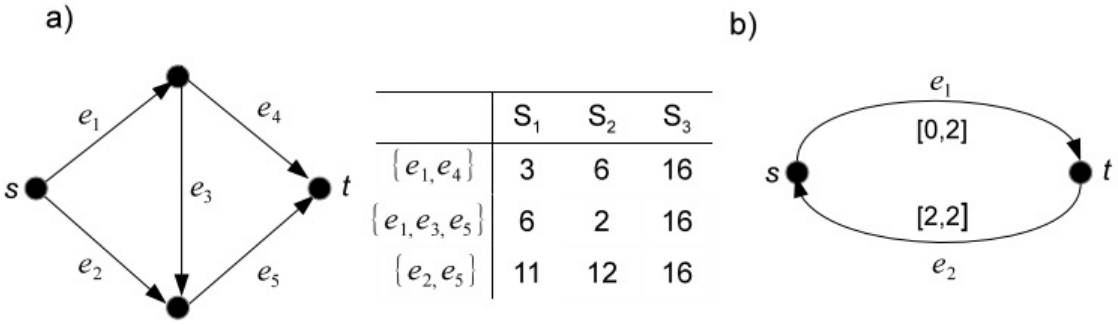
**Definicija 1.0.3.** ***Relativno robusno rješenje*** *$X_A$ XR je ono koje među svim dopustivim rješenjima minimizira maksimalni relativni otklon od optimuma gledajući sve scenarije, odnosno:*

$$z_R = \max_{s \in S} \frac{f(X_R, D^s) - f(X_s^*, D^s)}{f(X_s^*, D^s)} = \min_{X \in \cap_{s \in S} F_S} \max_{s \in S} \frac{f(X, D^s) - f(X_s^*, D^s)}{f(X_s^*, D^s)}; \tag{1.4}$$

Relativno robusno rješenje se izabire u slučajevima kao i rješenje robusnom devijacijom, ali kada je važniji relativni otklon od optimuma (izražen postotkom), a ne otklon u apsolutnim jedinicama.

Pri donošenju devijacijske ili relativne odluke svako se rješenje evaluira obzirom na najbolje moguće rješenje za pojedini scenarij. Devijacijski kriterij sklon je iskorištavanju mogućnosti za napredak. Odstupanje od optimalnosti je vrlo dobar indikator jer upućuje na to koliki se napredak može postići u pojedinim segmentima procesa ako se nesigurnost ulaznih podataka može barem djelomice riješiti. Ove odluke bi se trebale koristiti u okruženjima u kojima dolazi do velikih fluktuacija u performansama.

Iako se u ovom poglavlju ne proučava niti jedan problem, na slučajevima problema najkraćeg puta se može jasno vidjeti kada bi bilo bolje koristiti apsolutni, a kada devijacijski kriterij. Iskaz apsolutnog robusnog problema najkraćeg puta, kao i devijacijskog se može naći u poglavlju 3. Na slici 1.1 pod *a*) prikazan je usmjereni graf u kojemu svi putevi od $s$ do $t$ imaju ukupnu cijenu 16 pa se za riješenje izabire put $\{e_1, e_4\}$. U slučaju *b*) je jasno da se u skoro svim slučajevima put $\{e_1\}$ bolji od $\{e_2\}$, ali su oba apsolutno robusno rješenje budući da imaju istu vrijednost u najgorem slučaju.

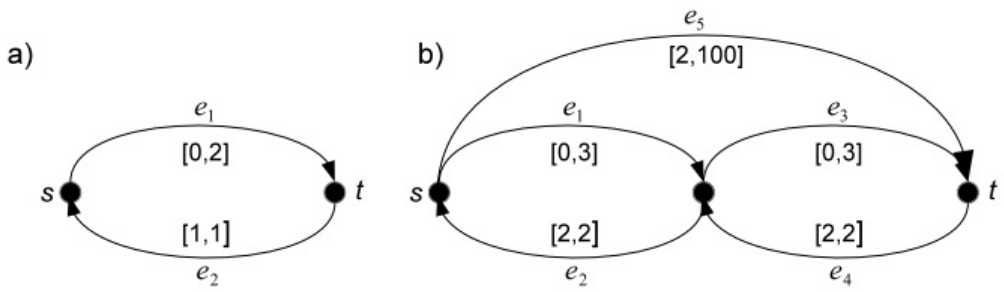



| | $S_1$ | $S_2$ | $S_3$ |
|---|---|---|---|
| $\{e_1, e_4\}$ | 3 | 6 | 16 |
| $\{e_1, e_3, e_5\}$ | 6 | 2 | 16 |
| $\{e_2, e_5\}$ | 11 | 12 | 16 |

Slika 1.1: *a*) Primjer problema najkraćeg puta sa scenarijima $S_1 = (2, 10, 3, 1, 1)$, $S_2 = (1, 11, 0, 5, 1)$, $S_3 = (8, 8, 0, 8, 8)$, *b*) Primjer problema najkraćeg puta sa cijenama koje dolaze iz intervala (Preuzeto iz [4])

Na slici 1.2 pod *a*) je dan primjer grafa kojemu u oba slučaja robusna devijacijska odluka daje vrijednost1, međutim, maksimalna cijena puta $\{e_1\}$ u najgorem slučaju je dvostruko veća od cijene $\{e_2\}$ u najgorem slučaju.

Pod *b*) je prikazan drugi problem ovog pristupa. Robusno devijacijsko rješenje je $\{e_1, e_4\}$, međutim izbacivanjem brida $e_5$ iz grafa, robusno devijacijsko rješenje postaje $\{e_1, e_3\}$. Iako je izbačen brid najveće moguće cijene i najvećeg mogućeg odstupanja, rješenje se promijenilo, što znači da je robusno devijacijsko rješenje ovisno i o irelevantnim scenarijima.




Slika 1.2: Primjeri problema najkraćeg puta sa scenarijima kojima vrijednosti dolaze iz intervala (Preuzeto iz [4])

# Poglavlje 2

# Klase P i NP

Kada je u pitanju promatranje algoritama i problema, to jest promatranje njihove vremenske složenosti, važno ih je klasificirati. Prvi problem koji se susreće je kako odrediti arhitekturu računala prema kojemu se određuje koja je vremenska složenost nekog algoritma. Jasno je da će suvremeni procesori daleko brže riješiti neke probleme nego procesori od prije 5, 10 ili više godina.
Zato se uzima Turingov stroj kao referentno računalo. To je teoretski stroj koji prelazi iz stanja u stanje ovisno o tome koji zapis se nalazi na traci te na istu traku upisuje i kalkulacije, a i konačno rješenje. Drugi problem je lako riješiti: problem određivanja „mjerne jedinice“ za brzinu izvršavanja, no za to se naprosto uzima u koliko stanja će ući glava Turingovog stroja u odnosu na duljinu ulaznog zapisa na traci. Naravno, neće se uvijek izvršiti jednak broj naredbi pa se u obzir uzima samo najgori mogući slučaj. Uvode se definicija i zapis:

**Definicija 2.0.1.** *Neka su $f, g : \mathbb{N} \to \mathbb{R}^+$ dvije funkcije. Funkcija $g$ je asimptotska gornja međa za funkciju $f$ ako postoji $c > 0$ i $n_0$ tako da za svaki $n \geq n_0$ vrijedi $f(n) \leq cg(n)$ u oznaci: $f(n) = O(g(n))$. Još se kaže da je $f$ vremenske složenosti $O(g)$.*

Za Turingov stroj kakav je gore naveden se može zaključiti da ima jedno završno stanje u kojemu staje kada završi s izračunom. Često se definiraju i Turingovi odlučitelji koji imaju dva završna stanja: $q_{da}$ i $q_{ne}$. Takvi strojevi provjeravaju je li dani ulaz rješenje nekog problema i staju u odgovarajućem stanju. U literaturi se često na taj način definira sam Turingov stroj ako se promatraju problemi odlučivanja. Formalna definicija stroja kao i odlučitelja je izvan dosega rada, ali se može pronaći u [7].
Još kratko o determinizmu Turingovih strojeva. Prelazak iz stanja u stanje deterministčkog Turingovog stroja je predvidljiv i unaprijed jasno definiran. Nedeterministički Turingov stroj nakon što je ušao u jedno stanje i pročitao simbol na traci može prijeći u više različitih stanja i zapisati različite simbole pa zato a-priori neće biti jasno kako bi mogao teći tok

izračunavanja na nedeterminističkom Turingovom stroju. Najbolje je mogućnost postupanja na različite načine zamišljati kao grananja kako je to opisano u skripti [10].
Izuzev ovoga, u ovom radu se neće govoriti o Turingovim strojevima, odlučiteljima i njihovom determinizmu. Riječ je o problemima optimizacije, tako da ni nema smisla govoriti o odlučivosti. Naime, iskazane tvrdnje su nužne kako bi se mogla dati barem okvirna definicija klase P, odnosno klase NP. Pa prema [10] slijedi:

Jezik se nalazi u klasi P ako postoji jednotračni deterministički Turingov stroj koji ga prepoznaje (staje u stanju $q_{da}$) u polinomnom vremenu.

Jezik se nalazi u klasi NP ako postoji jednotračni nedeterministički Turingov stroj koji ga prepoznaje u polinomnom vremenu.

S ovim definicijama je teško rukovati. Za svaki problem bilo bi potrebno definirati Turingov stroj sa svim pripadajućim stanjima i onda opravdavati njegovu vremensku složenost. U ovom iskazu teško je pojmiti definiciju klase NP. Teorem o certifikatu daje tvrdnju ekvivalentnu ovoj definiciji klase NP. Ovdje ga nije potrebno iskazati, ali ga ipak treba komentirati: često se za certifikat (koji je riječ iz jezika) uzima upravo rješenje problema te se u slučajevima kada je to moguće može reći da je problem u klasi NP kada je moguće provjeriti barem jedno njegovo rješenju u polinomnom vremenu.

No ni to neće biti dovoljno za potrebe ovog rada. Naime, klase P i NP sadrže probleme odlučivanja, ne optimizacijske probleme. Tome se može doskočiti na razne načine, jedan od kojih je promjena formulacije problema tako da umjesto da određuje optimalno rješenje, on provjerava je li neko dano rješenje upravo jednako onom optimalnom. Ipak, ni to neće biti u fokusu narednih poglavlja, već pojam NP-teških problema. Problem $L$ je NP-težak kada je svaki drugi problem $N$ iz NP moguće svesti na problem $L$ u polinomnom vremenu tako da su oni ekvivalentni ($w \in N$ ako $f(w) \in L$). Ako je pri tome problem L problem odlučivanja i nalazi se u klasi NP, onda se kaže da je on NP-potpun.

Zbog ovakve definicije, za potrebe ovog rada dovoljno je znati jedan problem koji se nalazi u NP, ali takav da je na njega moguće svesti druge jezike u polinomnom vremenu. 1971. Cook i Levin su u svojim radovima [2] i [6] pokazali da je problem SAT NP-potpun. Problem SAT je problem odlučivanja koji glasi: ”je li zadana KNF ispunjiva?”. Na njega se onda svode problemi kao što je 3-SAT pa na njega opet primjerice problem pokrivača bridova. Kako su sve transformacije u polinomnom vremenu, njihova kompozicija je ponovno polinomna tako da je svojstvo tranzitivno. U daljnjim poglavljima se problemi 2-particije, 3-particije i pokrivača skupa svode na apsolutne robusne probleme optimizacije, a iskazi tih problema su dani u definiciji:

**Definicija 2.0.2.** ***Problem 2-particije*** *glasi: za dani multiskup prirodnih brojeva $S$ treba odrediti postoji li $T \subseteq S$ takav da vrijedi $\sum_{x \in T} x = \sum_{x \in S \setminus T} x$.*
***Problem 3-particije*** *glasi: za dani multiskup prirodnih brojeva $S$ treba odrediti postoje li tročlani, po parovima disjunktni podskupovi od $S$ koji u uniji čine čitav $S$ takvi da su im sume elemenata jednake.*
***Problem pokrivač skupa*** *glasi: za dani univerzalni skup $U$, nenegativni broj $k \in \mathbb{N}$ i skup nekih njegovih podskupova $P$ kojima su pridružene nenegativne cijene potrebno je pronaći podskup skupa $P$ koji u uniji čini čitav $U$, a ima cijenu koja nije veća od $k$ (uz napomenu da $\cup P$ sigurno pokriva $U$).*

**Definicija 2.0.3.** *Kaže se da je vrijeme izvršavanja algoritma* ***pseudo-polinomno*** *ako je ograničeno polinomom u kojemu se kao varijable, osim skupa ulaznih podataka, pojavljuju i konkretne vrijednosti ulaznih podataka.*

Spomenute konkretne vrijednosti su najčešće neka ograničenja koja se prirodno pojavljuju u problemu. Primjerice, u problemu naprtnjače je to težinski kapacitet naprtnjače.
Sada je moguće definirati pojmove jako NP-teških i slabo NP-teških problema:

**Definicija 2.0.4.** *Problemi koji su NP-potpuni (odnosno NP-teški), ali su rješivi u pseudo-polinomnom vremenu se još nazivaju slabo NP-potpunim (slabo NP-teškim) problemima. U protivnom ih nazivamo jako NP-potpunim (jako NP-teškim problemima).*

**Napomena 2.0.5.** *Problem naprtanjče i problem 2-particije su slabo NP-teški, dok su problemi 3-particije i problem pokrivača skupa jako NP-teški.*

# Poglavlje 3

# Robusni problem najkraćeg puta

Problem najkraćeg puta je jedan od najpoznatijih problema kombinatorne optimizacije. Matematički je modeliran kao traženje puta od danog početnog vrha u grafu do danog završnog vrha pri tome minimizirajući sumu težina bridova kojima se prolazi:

Za dani povezani, usmjereni težinski graf $G = (V, A)$ s nenegativnim duljinama bridova $c_a$ za svaki $a \in A$, početnim i završnim vrhovima $v_0, v_t \in V$ treba pronaći put od $v_0$ do $v_t$ s najmanjom težinom.

1959. Dijkstra je dao algoritam koji rješava ovaj problem u polinomnom vremenu $O(|V|^2)$ u članku [3]. Dijkstrin algoritam "gradi" put tako da u početku definira skup $D$ u kojemu se nalazi samo početni vrh $v_0$ i skup bridova $B$ koji je prazan. k-ti korak algoritma izgleda ovako: uzmi slijedeći vrh iz $G$ te među svim bridovima koji su incidentni s tim vrhom i nekim od vrhova iz $D$ uzmi brid najmanje težine i dodaj taj brid u $B$, a vrh u $D$. Zaustavi se kada $D = G$. Vremenska složenost algoritma se lako vidi jer su u najgorem slučaju svi vrhovi grafa povezani sa svim ostalim vrhovima pa tako za svaki od $n$ vrhova postoji $n - 1$ brid čiju težinu treba usporediti s ostalima.

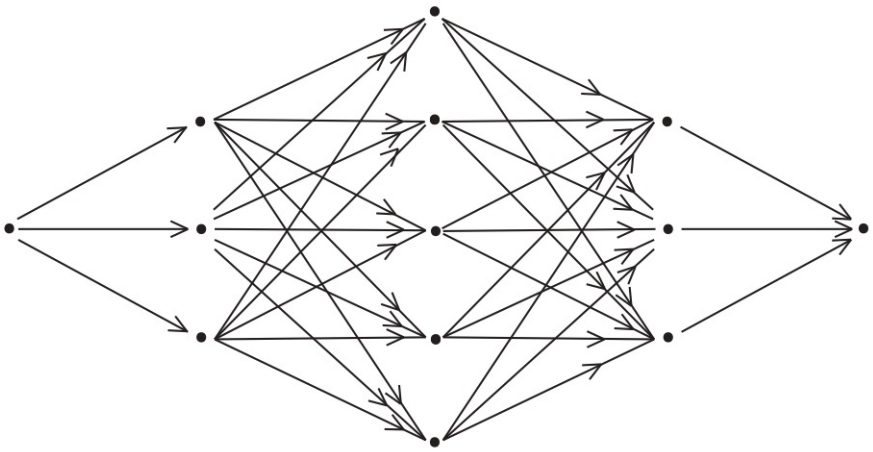
U robusnom problemu najkraćeg puta (apsolutni, relativni i devijacijski) skup težina bridova grafa je unaprijed definiran i jedan takav skup je scenarij. Problem je riješen apsolutno robusno kada se među svim putovima od početnog do završnog vrha u usmjerenom grafu pronađe onaj čija je maksimalna duljina mjerena po svim scenarijima najmanja. Motivacija za rješavanjem ovakvog problema je bilo kakva prometna situacija u kojoj je potrebno pronaći najbrži put u najgorim uvjetima.

## 3.1 Složenost robusnog problema najkraćeg puta

U nastavku se problem najkraćeg puta promatra na vrlo restringiranom grafu: na slojevitoj mreži.

**Definicija 3.1.1.** ***Slojevita mreža*** *je graf kojemu se skup vrhova može podijeliti na disjunktne podskupove* $V = \{v_0\} \cup V_1 \cup V_2 \cup ... \cup V_m \cup \{v_t\}$ *tako da bridovi postoje samo između skupova* $\{v_0\}$ *i* $V_1$*;* $V_k$ *i* $V_{k+1}$ *za k=2, 3,...,m-1; i* $V_m$ *i* $\{v_t\}$*.*

*Maksimalni kardinalitet skupova na koje se particionira skup V se naziva širinom slojevite mreže. Bridovi slojevite mreže se često nazivaju lukovima.*



Slika 3.1: Slojevita mreža širine 5

Ranije je opisan problem najkraćeg puta, no potrebno ga je precizirati:

$$(NP) \quad z^s = min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\text{t.d.} \quad \sum_{j\in V} x_{ij} - \sum_{k\in V} x_{ki} = \begin{cases} 1, & \text{za } i = v_0 \\ -1, & \text{za } i = v_t \\ 0, & \text{inače} \end{cases} \tag{3.1}$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in A \tag{3.2}$$

Gdje varijabla $x_{ij}$ poprima vrijednost 1 kada je brid $(i, j)$ izabran kao dio rješenja. Uvjet 3.1 osigurava da su izabrani bridovi koji čine put od početnog do završnog vrha grafa, a funkcija cilja je zbroj duljina tih bridova i ona se minimizira.
U odnosu na tako iznesen problem najkraćeg puta definiraju se apsolutni i devijacijski problem na slijedeći način:

$$(NP)_A \quad z_A(NP) = \min \max_{s\in S}\{\sum_{(i,j)\in A} c_{ij}^s x_{ij}\} \quad \text{t.d. 3.1 i 3.2} \tag{3.3}$$

$$(NP)_D \quad z_D(NP) = \min \max_{s\in S}\{\sum_{(i,j)\in A} c_{ij}^s x_{ij} - z^s\} \quad \text{t.d. 3.1 i 3.2} \tag{3.4}$$

Problem najkraćeg puta na svakoj mreži je rješiv u polinomnom vremenu, a robusni problem na mrežama širine jedan je trivijalan. Stoga se promatraju slojevite mreže širine 2 i sa 2 scenarija obzirom da je to najjednostavniji mogući slučaj robusnih problema najkraćeg puta koji su NP-potpuni.

**Teorem 3.1.2.** *Apsolutni robusni problem najkraćeg puta je NP-težak čak i u slojevitim mrežama širine dva i sa samo dva secnarija.*

*Dokaz.* Treba pokazati da je problem 2-particije polinomno reducibilan na apsolutni robusni problem najkraćeg puta $(NP)_A$.
Neka je $S$ proizvoljni skup s $m$ elemenata kojima su pridružene nenegativne vrijednosti $a_k$ za $k = 1, ..., m$. Postavlja se pitanje postoji li $S' \subseteq S$ koji rješava problem 2-particije na skupu $S$.
Konstruira se slojevita mreža prikazana na Slici 3.1 na slijedeći način:
Skup vrhova se dijeli na 2m+2 sloja: $V = \{v_0\} \cup V_1 \cup V_2 \cup ... \cup V_{2m} \cup \{v_t\}$ pri čemu su početni i završni vrh u zasebnim, jednočlanim slojevima, a preostali slojevi se sastoje od dva vrha koja označavamo $V_k = \{v_{k1}, v_{k2}\}$. Skup bridova dijeli se na tri dijela: početne i završne bridove, bridove između slojeva mreže i dijagonalne bridove među slojevima. Preciznije: $A = A_1 \cup A_2 \cup A_3$ gdje su:

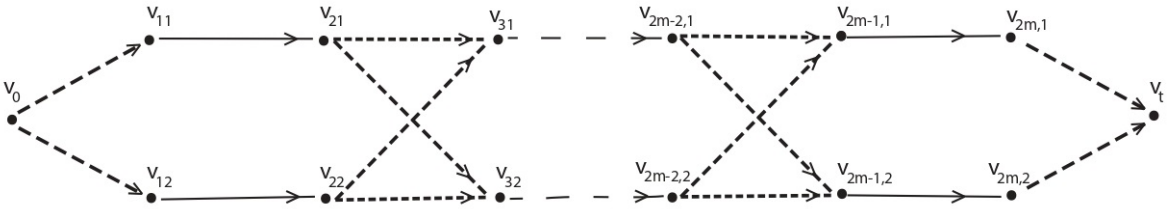
$$\begin{aligned} A_1 &= \{(v_0, v_{11}), (v_0, v_{12}), (v_{2m,1}, v_t), (v_{2m,2}, v_t)\}, \\ A_2 &= \cup_{k=1}^{m-1}\{(v_{2k,i}, v_{2k+1,j}) : i = 1, 2; j = 1, 2\}, \\ A_3 &= \{(v_{2k-1,i}, v_{2k,i}) : i = 1, 2; k = 1, 2, ..., m\} \end{aligned}$$

$(NP)_A$ definira se za 2 scenarija u kojima je svim bridovima iz $E_1$ i $E_2$ pridružena vrijednost 0 u oba scenarija. Bridovima u $E_3$ pridružene su slijedeće vrijednosti:

$$c^1_{v_{2k-1,i}, v_{2k,i}} = \begin{cases} a_k, & \text{za } i = 1 \\ 0, & \text{za } i = 2 \end{cases}$$

$$c^2_{v_{2k-1,i}, v_{2k,i}} = \begin{cases} 0, & \text{za } i = 1 \\ a_k, & \text{za } i = 2 \end{cases}$$

Neka je $\pi$ put od $v_0$ od $v_t$ te neka je $I' = \{k : (v_{2k-1,2}, v_{2k,12}) \in \pi \text{ za svaki } k \in I\}$. Tada je nužno $I \setminus I' = \{k : (v_{2k-1,1}, v_{2k,1}) \in \pi \text{ za svaki } k \in I\}$. Ukupna duljina puta u scenariju 1 se




Slika 3.2: Slojevita mreža konstruirana prema gornjem opisu; iscrtkane linije predstavljaju bridove duljine 0, pune linije u gornjem retku imaju vrijednosti $a_k$ u scenariju 1, a donje u scenariju 2 (inače iznose 0)

može zapisati kao $\sum_{k\in I'} a_k$, a u scenariju 2 kao $\sum_{k\in I\setminus I'} a_k$.
Dakle, 2-particija postoji ako i samo ako $(NP)_A$ problem ima rješenje ukupne duljine $z_A(NP) = \frac{1}{2}\sum_{k\in I} a_k$.

□

**Teorem 3.1.3.** *Devijacijski robusni problem najkraćeg puta je NP-težak čak i u slojevitim mrežama širine dva i sa samo dva secnarija.*

*Dokaz.* Konstruira se ista mreža kao u teoremu 3.1.2. Najkraći put u oba scenarija je duljine 0 iz čega slijedi da je član $z_s$ iz definicije problema $(NP)_D$ 3.4 jednak nuli. S time u vidu, nastavak dokaza je isti kao u prethodnom teoremu.

□

U dokazima se problem 2-particije, koji je rješiv u pseudo-polinomnom vremenu, reducira na problem $(NP)_A$, odnosno $(NP)_D$. Slijedeći teorem daje odgovor u kojim uvjetima su i $(NP)_A$ i $(NP)_D$ rješivi u pseudo-polinomnom vremenu.

**Teorem 3.1.4.** *$(NP)_A$ i $(NP)_D$ su rješivi u pseudo-polinomnom vremenu na slojevitim mrežam s ograničenim brojem scenarija.*

*Dokaz.* U dokazu se konstruira algoritam za rješavanje problema u pseudo-polinomnom vremenu.
Neka je $G = (V, A)$ slojevita mreža kojoj je skup $V$ podijeljen na slojeve $V = \{v_0\} \cup V_1 \cup V_2 \cup ... \cup V_m \cup \{v_t\}$ pri čemu $V_k = \{v_{ki} : i = 1, ..., n_k\}$. Skup bridova je dan: $E = \{(v_0, v_{1i}) : i = 1, ..., n_1\} \cup \{(v_{mi}, v_t) : i = 1, ..., n_m\} \cup \{(v_{ki}, v_{k+1,j}) : i = 1, ..., n_k; j = 1, ..., n_{k+1}; k = 1, ..., m-1\}$.
Algoritam je zasnovan na dinamičkom programiranju. Funkcija $f_v(\alpha_1, ..., \alpha_{|S|})$ definirana

je kao minimalna vrijednost skupa svih maksimalnih duljina puta od vrha $v$ do vrha $v_t$ po svim scenarijima. Pritom se pretpostavlja da je za dolazak do $v$ korišten početni dio puta koji u scenariju $s$ ima duljinu $\alpha_s$.
Za rekurziju još je potreban inicijalni uvjet:

$$f_{v_t}(\alpha_1, ..., \alpha_{|S|}) = \max_{s \in S} \alpha_s$$

Sada se definira rekurzivna relacija:

$$f_{v_{ki}}(\alpha_1, ..., \alpha_{|S|}) = \min_{j=1,...,n_{k+1}} \{f_{v_{k+1,j}}(\alpha_1 + c^1_{v_{ki},v_{k+1,j}}, ..., \alpha_{|S|} + c^{|S|}_{v_{ki},v_{k+1,j}})\}$$

Optimalno rješenje za aspolutni problem dobiva se kao $f_{v_0}(0, ..., 0)$, odnosno kao $f_{v_0}(-z^1, ..., -z^{|S|})$ za devijacijski problem.

Pseudokod algoritma za računanje $(NP)_A$ izgleda kako slijedi:

**Algorithm 1** RobusniNPA($G = (V, A)$ slojevita mreža; $c$ duljine bridova)

$L_s \leftarrow$ dinamičkim programiranjem izračunaj najdulji put od $v_0$ do $v_t$ za svaki scenarij $s \in S$
**for** $\alpha_1 = 0$ to $L_1$ **do**
  ...
  **for** $\alpha_{|S|} = 0$ to $L_{|S|}$ **do**
    $f_{v_t}(\alpha_1, ..., \alpha_{|S|}) = \max_{s \in S} \alpha_s$
  **end for**
  ...
**end for**
**for** $k = m$ downto 0 **do**
  **for** $i = 1$ to $n_k$ **do**
    **for** $\alpha_1 = 0$ to $L_1$ **do**
      ...
      **for** $\alpha_{|S|} = 0$ to $L_{|S|}$ **do**
        $f_{v_{ki}}(\alpha_1, ..., \alpha_{|S|}) = \min_{j=1,...,n_{k+1}} \{f_{v_{k+1,j}}(\alpha_1 + c^1_{v_{ki},v_{k+1,j}}, ..., \alpha_{|S|} + c^{|S|}_{v_{ki},v_{k+1,j}})\}$
      **end for**
      ...
    **end for**
  **end for**
**end for**
**return** $f_{v_0}(0, ..., 0)$

Dok je za $(NP)_D$ potrebno napraviti samo par izmjena:

**Algorithm 2** RobusniNPD($G = (V, A)$ slojevita mreža; $c$ duljine bridova)

$z^s \leftarrow$ dinamičkim programiranjem izračunaj najkraći put od $v_0$ do $v_t$ za svaki scenarij $s \in S$
**for** $\alpha_1 = -z^1$ to $L_1 - z^1$ **do**
  ...
  **for** $\alpha_{|S|} = -z^{|S|}$ to $L_{|S|} - z^{|S|}$ **do**
    $f_{v_t}(\alpha_1, ..., \alpha_{|S|}) = \max_{s \in S} \alpha_s$
  **end for**
  ...
**end for**
**for** $k = m$ downto 0 **do**
  **for** i = 1 to $n_k$ **do**
    **for** $\alpha_1 = 0$ to $-z^1$ **do**
      ...
      **for** $\alpha_{|S|} = 0$ to $-z^{|S|}$ **do**
        $f_{v_{ki}}(\alpha_1, ..., \alpha_{|S|}) = \min_{j=1,...,n_{k+1}} \{f_{v_{k+1,j}}(\alpha_1 + c^1_{v_{ki}, v_{k+1,j}}, ..., \alpha_{|S|} + c^{|S|}_{v_{ki}, v_{k+1,j}})\}$
      **end for**
      ...
    **end for**
  **end for**
**end for**
**return** $f_{v_0}(-z^1, ..., -z^{|S|})$

Inicijalizacija oba prikazana algoritma zahtjeva $O(\max\{|A||S|, \cap_{s \in S} L_s\})$ vremena dok je glavni dio algoritma vremenske složenosti $O(|A| L^{|S|}_{max})$ pri čemu je uvedena oznaka $L_{max}$ za $\max_{s \in S} L_s$.

□

**Teorem 3.1.5.** *$(NP)_A$ i $(NP)_D$ su jako NP-teški kada je skup scenarija neograničen.*

*Dokaz.* Potrebno je reducirati problem 3-particije na problem $(NP)_A$.
Neka je S proizvoljni skup s $m$ elemenata kojima su pridružene nenegativne vrijendosti $a_k$ za $k = 1, ..., m$. Potrebno je pronaći particiju skupa $S$ na tročlane disjunktne skupove takve da je svakom pojedinom suma jednaka. Neka je ta suma označena sa $B$.
Neka $(NP)_A$ ima $m$ scenarija. Konstruira se slojevita mreža sa $6m + 2$ sloja tako da se u nultom sloju nalazi samo vrh $v_0$, a u zadnjem samo vrh $v_t$, a preostali slojevi su istog kardinaliteta $m$. Uvode se oznake $V_k = \{v_{ki} : i = 1, ..., m\}$ za svaki sloj $k = 1, ..., 6m$ pri čemu kažemo da se vrhovi $v_{ki}$ i $v_{li}$ nalaze na istoj razini.
Skup bridova se dijeli na tri sloja: $A_1$ sadrži sve bridove od početnog vrha do $V_1$ i od $V_{6m}$ do završnog; $A_2$ sadrži sve bridove između slojeva koji ne povezuju vrhove na istoj razini;

a $A_3$ sadrži sve bridove koji povezuju vrhove iste razine.
Preciznije:

$$A = A_1 \cup A_2 \cup A_3$$
$$A_1 = \{(v_0, v_{1i}) : i = 1, ..., m\} \cup \{(v_{6m,i}, v_t) : i = 1, ..., m\}$$
$$A_2 = \cup_{k=1}^{3m-1}\{(v_{2k,i}, v_{2k+1,j}) : i = 1, ..., m; j = 1, ..., m; i \neq j\}$$
$$A_3 = \{(v_{2k-1,i}, v_{2k,i}) : i = 1, ..., m; k = 1, ..., 3m\}$$

Težine bridova su jednake 0 za svaki brid iz $A_1$ i $A_2$, a za bridove u $A_3$ vrijedi:

$$c^s_{v_{2k-1,i},v_{2k,i}} = \begin{cases} a_k, & \text{za } i = s \\ 0, & \text{inače.} \end{cases}$$

Tvrdnja koja se sada pokazuje je slijedeća: Postoji 3-particija definiranog skupa ako i samo ako $(NP)_A$ ima optimalnu vrijednost $z_s(NP) = B$.
Ako postoji 3-particija za skup $S$, tada se on može podijeliti u podskupove $P_1, ..., P_m$ pri čemu $\sum_{k\in I_i} a_k = B$ za $i = 1, ..., m$. Time svi putevi koji su na istoj razini imaju istu vrijednost $B$. Da bi graf bio potpun dodaju se potrebni bridovi do $v_0$ i $v_t$ te bridovi iz $E_2$ čija je težina 0 pa se suma ne mijenja. Obzirom da je duljina svih puteva od $v_0$ do $v_t$ jednaka $B$ za svaki scenarij, tada po 3.3 tvrdnja slijedi.

Za suprotni smjer se pretpostavlja $z_s(NP) = B$. Neka je $\pi$ optimalno robusno najkraći put i neka je $I_i$ skup svih bridova razine $i$ koje $\pi$ obilazi za $i = 1, ..., m$. Jasno je da su $I_i$-evi međusobno disjunktni. Po konstrukciji mreže slijedi da je duljina puta u scenariju $s$ jednaka $\sum_{i=1}^{m}\sum_{k\in I_i} c^s_{v_{2k-1,i},v_{2k,i}} = \sum_{k\in I_s} a_k$. Prema pretpostavci slijedi da je maksimum te sume po svim scenarijima upravo jednak B. $\sum_{k\in I} a_k = mB$ povlači da je $\sum_{k\in I_s} a_k = B$ što daje 3-particiju.
Prema 2.0.5 slijedi da je $(NP)_A$ jako NP-težak.
Dokaz je potpuno analogan i za $(NP)_D$ jer su optimalne vrijednosti za pojedini scenarij $z^s = 0, s \in S$.

□

# Poglavlje 4

# Robusni problem najmanjeg razapinjućeg stabla

Uz problem najkraćeg puta, problem najmanjeg razapinjućeg stabla je jedan od najproučavanijih problema kombinatorne optimizacije. Za dani povezani, neusmjereni graf $G = (V, E)$ se definiraju razapinjuća stabla kao povezani podgrafovi $T$ koji sadrže sve vrhove iz $G$, ali ne sadrže niti jedan ciklus.

Ako je graf $G$ težinski s nenegativnim cijenama bridova, tada ima smisla definirati pojam najmanjeg razapinjućeg stabla, a ono je razapinjuće stablo s najmanjom ukupnom cijenom bridova. Jasno je da najmanje razapinjuće stablo, kao ni najkraći put iz prošlog poglavlja ne moraju biti jedinstveni.
Češki matematičar Vojtěch Jarník je 1930. godine dao algoritam koji rješava problem najmanjeg razapinjućeg stabla u polinomnom vremenu. Taj je algoritam ponovno otkrio i objavio Robert Prim 1957. godine te ga se naziva Primovim algoritmom. Izuzev Primovog algoritma često se koristi i Kruskalov algoritam objavljen 1956. godine. Kruskalov algoritma rješava problem u polinomnom vremenu $O(|E|log|E|)$, a Primov u $O(min|V|^2, |E|log|V|)$, što je također polinomno. Primov algoritam započinje inicijalizacijom stabla na jedan proizvoljni vrh. Zatim se među svim bridovima koji povezuju stablo s ostatkom grafa izabire onaj s najmanjom težinom te se pripadajući vrh i brid dodaju u stablo. Drugi korak se ponavlja sve dok svi vrhovi ne budu u stablu. Po konstrukciji je jasno da je taj graf povezan i da sadrži sve vrhove grafa $G$, a ne sadrži cikluse jer se uvijek izabire samo jedan brid koji vodi do nekog vrha. Kruskalov algoritam polazi od skupa svih vrhova grafa $G$ u oznaci $F$ (*Forest*). Konstruira se skup bridova razapinjućeg stabla $S$ tako da se među svim bridovima grafa $G$ izabire onaj s najmanjom cijenom koji povezuje dva različita stabla iz $F$ čime ih spaja u jedno stablo (ovaj uvjet osigurava da nema ciklusa). Algoritam staje kada $F$ postane razapinjuće satblo, ili kada su svi bridovi iz $E$ dodani u skup $S$.

Apsolutni robusni problem najmanjeg razapinjućeg stabla podrazumjeva da je svakom bridu $e \in E$ pridružena nenegativna vrijednost $c_e^s$ ovisna o scenariju $s \in S$. Telekomunikacijske mreže ili mreže transporta su samo neki primjeri u kojima se pojavljuje ovaj problem. Rješavanjem ovog problema razvija se mreža koja osigurava da se najgori mogući slučaj po pitanju cijene puta (ili kašnjenja u komunikaciji) neće dogoditi.

## 4.1 Složenost robusnog problema najmanjeg razapinjućeg stabla

U nastavku se pokazuje da je problem najmanjeg razapinjućeg stabla već NP-težak i na rešetkastom grafu.

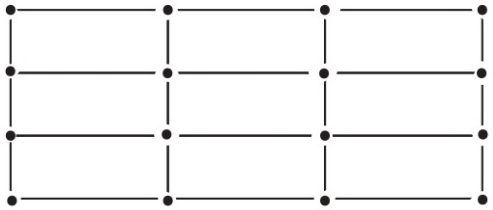
**Definicija 4.1.1.** ***Rešetkasti graf*** *veličine* $(m, n)$ *je definiran skupom vrhova* $V = \{v_{ij} : i = 1, ..., m; j = 1, ..., n\}$ *i skupom bridova* $E = E_r \cup E_c$*, pri čemu su:*

$$E_r = \{(v_{ij}, v_{i,j+1}) : i = 1, ..., m; j = 1, ..., n-1\}$$

*i*

$$E_c = \{(v_{ij}, v_{i+1,j}) : i = 1, ..., m-1; j = 1, ..., n\}$$

Elementi skupa $E_r$ nazivaju se bridovi redaka, a elementi od $E_c$ bridovi stupaca. Jasno je da je $|V| = mn$, a $|E| = 2mn - m - n$.



Slika 4.1: Rešetkasti graf dimenzija (4, 4)

Precizirani apsolutni robusni problem najmanjeg razapinjučeg stabla glasi:

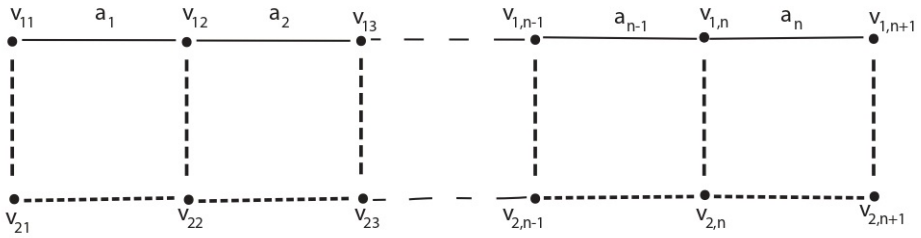
$$(NRS)_A \quad z_A(NRS) = min_T max_{s \in S} \sum_{e \in T} c_e^s \quad \text{t.d. } T \text{ razapinjuće stablo.} \tag{4.1}$$

**Teorem 4.1.2.** *Apsolutni robusni problem najmanjeg razapinjućeg stabla je NP-težak čak i u slučaju kada je:*

- *G je rešetkasti graf s dva retka (m = 2)*
- $c_e^s = 0$, $s \in S$, $e \in E_c$ *(ne nužno za* $e \in E_r$*)*
- $|S| = 2$

*Dokaz.* Problem 2-particije se svodi na ranije zapisani problem. Neka je $I$ skup kojeg se pokušava particionirati na dva dijela $(I', I \setminus I')$ tako da suma vrijednosti elemenata oba skupa ma istu vrijednost i neka mu elementi imaju pridružene vrijednosti $a_j$, $j \in I$. Prvo se konstruira rešetkasti graf kojemu su $m = 2$ i $n = |I| + 1$, a zatim se određuju cijene bridova za oba scenarija:

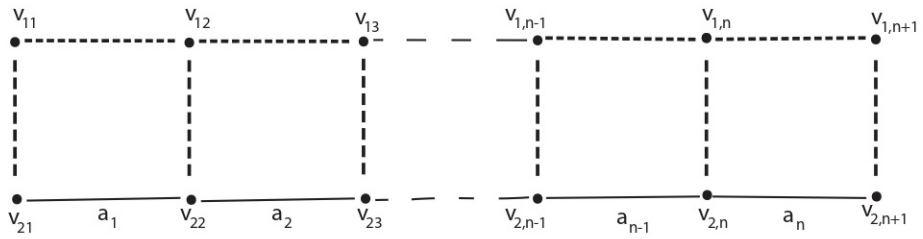
$$
\begin{aligned}
c_e^s &= 0 \quad e \in E_c; s \in S \\
c^1_{v_{1j},v_{1,j+1}} &= a_i \quad j = 1, ..., n-1 \\
c^1_{v_{2j},v_{2,j+1}} &= 0 \quad j = 1, ..., n-1 \\
c^2_{v_{1j},v_{1,j+1}} &= 0 \quad j = 1, ..., n-1 \\
c^2_{v_{2j},v_{2,j+1}} &= a_i \quad j = 1, ..., n-1
\end{aligned}
$$




Slika 4.2: Rešetkasti graf za scenarij 1

Rješenje za problem 2-particije postoji ako i samo ako problem $(NRS)_A$ ima optimalnu vrijednost $\frac{1}{2}\sum_{j\in I} a_j$.

Budući da su cijene svih bridova nenegativne, tada su to specijalno i cijene bridova redaka pa postoji optimalno min-max razapinjuće stablo koje sadrži sve bridove iz $E_c$. Pod pretpostavkom da su u prvom retku robusnog minimalnog razapinjućeg stabla izabrani samo bridovi $(v_{1j}, v_{1,j+1})$, $j \in I'$ slijedi da bridovi $(v_{2j}, v_{2,j+1})$, $j \in I \setminus I'$ iz drugog retka također moraju biti u stablu. Tada je ukupna cijena bridova u prvom scenariju jednaka $z^1 = \sum_{j\in I'} a_j$,




Slika 4.3: Rešetkasti graf za scenarij 2

a u drugom scenariju simetrično: $z^2 = \sum_{j \in I \setminus I'} a_j$. Ako je $z_A(NRS) = \frac{1}{2} \sum_{j \in I} a_j$, po definiciji je to $max\{z^1, z^2\}$ te tada direktno slijedi da su sume $z^1$i $z^2$ jednake što znači da 2-particija postoji.

Za drugi smjer se pretpostavja da 2-particija postoji. Tada se razapinjuće stablo sastoji od slijedećih bridova: $v_{1j}, v_{1,j+1}$ za $j \in I$ te $v_{2j}, v_{2,j+1}$ za $j \in I \setminus I'$. Kao u prethodnom slučaju, ukupna vrijednost bridova u stablu je $z_A(NRS) = \frac{1}{2} \sum_{j \in I} a_j$.

□

Gornji rezultat sugerira da bi apsolutni robusni problem najmanjeg razapinjućeg stabla mogao biti slabo NP-težak jer je moguće svesti problem 2-particije na njega. Uvjeti iz teorema su poprilično restringirajući, a moguće je dati nešto općenitije uvjete da bi problem bio slabo NP-težak. S druge strane, apsolutno robusni problem najmanjeg razapinjućeg stabla postaje jako NP-težak čim je broj scenarija neograničen, čak i ako se ograničimo na rešetkastu mrežu.

**Teorem 4.1.3.** *Apsolutni robusni problem najmanjeg razapinjućeg stabla* $(MST)_A$ *je slabo NP-težak i na rešetkastom grafu s ovim ograničenjima:*

- *G je rešetkasti graf*
- $c_e^s = 0$, $s \in S$, $e \in E_c$ *(ne nužno za* $e \in E_r$*)*
- $|S|$ *je ograničen konstantom neovisnom o rastu veličine grafa* $(m, n)$

*Dokaz.* Kako bi se dokazao ovaj teorem, daje se konkretan algoritam pseudo-polinomne vremenske složenosti. Prvo se izabiru svi bridovi stupaca, a bridovi iz $E_r$ se određuju rekurzivno:

$g_j(\alpha_1, ..., \alpha_{|S|})$ = min-max vrijednost stabla koje je razapeto na vrhovima od $j$ do $n$ i kada je nadodana vrijednost $\alpha_s$ za scenarij $s \in S$.

S time se može dati inicijalni uvjet rekurzije:

$$g_n(\alpha_1, ..., \alpha_{|S|}) = \max_{s \in S} \alpha_s,$$

a zatim i rekurzija:

$$g_j(\alpha_1, ..., \alpha_{|S|}) = \min_{i=1,...,m} g_{j+1}(\alpha_1 + c^1_{v_{ij}, v_{i,j+1}}, ..., \alpha_{|S|} + c^{|S|}_{v_{ij}, v_{i,j+1}})$$

.

Kako je i ranije spomenuto, stablo se konstruira "od kraja" što znači da je $z_A(NRS) = g_1(0, ..., 0)$. Takvo stablo sadrži sve bridove stupaca i po jedan brid koji ih povezuje određen prema forumli: $g_j(\alpha_1, ..., \alpha_{|S|}) = g_{j+1}(\alpha_1 + c^1_{v_{ij}, v_{i,j+1}}, ..., \alpha_{|S|} + c^{|S|}_{v_{ij}, v_{i,j+1}})$. U slučaju da dva brida imaju istu vrijednost može se izabrati bilo koji od njih. Pseudokod algoritma slijedi:

**Algorithm 3** RobusniNRS($G = (V, E)$ rešetkasti graf; $c$ duljine bridova)

Inicijalizacija: za svaki scenarij $s \in S$ izračunaj vrijednost razapinjućeg stabla $L_s$ koje uključuje sve bridove iz $E_c$ i bridove: $\{(v_{ij}, v_{i,j+1}) : i = argmax_{i=1,...,m} c^s_{v_{ij}, v_{i,j+1}}; j = 1, ..., n-1\}$
**for** $\alpha_1 = 0$ to $L_1$ **do**
  ...
  **for** $\alpha_{|S|} = 0$ to $L_{|S|}$ **do**
    $g_n(\alpha_1, ..., \alpha_{|S|}) = \max_{s \in S} \alpha_s$;
  **end for**
  ...
**end for**
**for** $j = n - 1$ down to 1 **do**
  **for** $\alpha_1 = 0$ to $L_1$ **do**
    ...
    **for** $\alpha_{|S|} = 0$ to $L_{|S|}$ **do**
      $g_j(\alpha_1, ..., \alpha_{|S|}) = \min_{i=1,...,m}\{g_{j+1}(\alpha_1 + c^1_{v_{ij}, v_{i,j+1}}, ..., \alpha_{|S|} + c^{|S|}_{v_{ij}, v_{i,j+1}})\}$;
    **end for**
    ...
  **end for**
**end for**
**return** $g_1(0, ..., 0)$

Inicijalizacija algoritma vremenske je složenosti $O(mn|S|)$, a glavna petlja $O(mn \prod_{s \in S} L_s)$ što je i ukupna složenost algoritma. Ta vrijednost je ograničena konstantom $|E|L^{|S|}_{max}$ (gdje

je $L_{max} = max_{s \in S} L_s$) pa je i $|S|$ ograničen konstantom. Prema 2.0.3 slijedi da je algoritam pseudo-polinomne vremenske složenosti.

□

**Teorem 4.1.4.** *Apsolutni robusni problem najmanjeg razapinjućeg stabla* $(NRS)_A$ *je jako NP-težak kada je broj scenarija neograničen čak i na rešetkastom grafu.*

*Dokaz.* Problem 3-particije se definira nad skupom $I$ koji sadrži $3l$ elemenata veličina $a_i, i \in I$ s odgovarajućom sumom elemenata $B$. Problem 3-particije je rješiv onda kada se skup $I$ može particionirati na $l$ tročlanih skupova sa svojstvom da je suma elemenata svakog pojedinog skupa jednaka upravo $B$.

Prvo se definiraju konstante $m = l$ i $n = 3l + 1$ kojima je onda definiran i rešetkasti graf $(m, n)$, a zatim i primjerak $(NRS)_A$ koji ima $l$ scenarija sa cijenama bridova:

$$c_e^s = 0 \qquad e \in E_c; s \in S$$

$$c^s_{v_{i,j},v_{i,j+1}} = \begin{cases} a_j, & \text{za } s = i \\ 0, & \text{inače.} \end{cases} \qquad i = 1, ..., m; j = 1, ..., n-1; s \in S \tag{4.2}$$

Pokazuje se tvrdnja: 3-particija skupa $I$ postoji ako i samo ako robusno minimalno razapinjuće stablo ima težinu $z_A(NRS) = B$.
Pretpostavka je da optimalno rješenje za $(NRS)_A$ uključuje bridove iz skupa $E_c \cup (\cup_{i=1}^{l} \{(v_{i,j}, v_{i,j+1}) : j \in I_i\})$. Razapinjuća stabla $I_i$ prirodno definiraju particiju skupa $I$ na skupove $I_i$, naime $I_i \cap I_{i'} = \emptyset, i \neq i'$ jer ta stabla ne mogu imati zajedničke vrhove. Ukupna cijena prolaska razapinjućeg stabla u scenariju $s$ iznosi $z^s = \sum_{k \in I_s} a_k$ po definiciji 4.2. Zbog particioniranja slijedi $z^s = \sum_{s \in S} \sum_{k \in I_s} a_k = \sum_{k \in I} a_k = lB$ pa je $z_A(NRS) = max_{s \in S} z^s$. $l$ i $B$ su konstante koje ne ovise o scenariju, a $|S| = l$, dakle $I$ (koji ima $3l$ elemenata) se može particionirati na $l$ tročlanih skupova čije su sume elemenata upravo jednake $B$. Odnosno da 3-particija skupa $I$ postoji ako i samo ako je $z^s = B$ za $s \in S$, to jest $z_A(NRS) = B$.

□

**Napomena 4.1.5.** *Devijacijski problem najmanjeg razapinjućeg stabla definira se na slijedeći način:*

$$(NRS)_D \quad z_D(NRS) = \min_T \max_{s \in S} \left( \sum_{e \in T} c_e^s - z^s \right)$$

*takvo da je $T$ razapinjuće stablo, a $z^s$ optimalno rješenje za scenarij $s$. $(NRS)_D$ je slabo (jako) NP-težak s istim uvjetima kao i kada je $(NRS)_A$ slabo (jako) NP-težak, a dokazi su sasvim analogni budući da u opisanim scenarijima optimalno rješenje $z^s$ iznosi 0 pa je formula jednaka formuli 4.1.*

# Poglavlje 5

# Robusni problem naprtnjače

Problem naprtnjače može se izreći na različite načine i svaki od tih načina dolazi sa pripadajućim varijantama problema ovisno o matematičkoj disciplini koja se promatra. Najčešće je izrečen ovako: za dani $W \in \mathbb{N}$ (težinski kapacitet naprtnjače) i skup uređenih parova $P = \{(w_i, v_i) : w_i > 0; v_i > 0; i = 1, ..., n; n \in \mathbb{N}\}$ potrebno je odrediti podskup $X$ skupa $P$ takav da $\sum_{k \in X} w_k \leq W$ i da je $\sum_{k \in X} v_k$ maksimalan. Kada se problem promatra u najširem smislu, skup P može biti multiskup ili čak neograničen. U slučajevima kada se promatra skup, a ne multiskup te je to potrebno specificirati, ovaj se problem naziva i (0/1)-problemom naprtnjače. U disciplinama koje se bave NP-potpunosti problema potrebno ga je definirati kao problem odlučivanja, a ne optimizacije pa se u tim slučajevima na ovako iskazane pretpostavke problema dodaje vrijednost $M$ i tada je potrebno odrediti postoji li način pakiranja naprtnjače kojemu je ukupna vrijednost odabranih predmeta veća ili jednaka $M$. U nastavku se upravo promatra (0/1)-problem naprtnjače za koji je poznato da je slabo NP-težak. Verzije problema se proučavaju već preko 100 godina, a pod imenom "knapsack problem" je poznat još od ranih radova Tobiasa Dantziga. U literaturi se referencira Paolo Tothov (1980.) algoritam koji rješava problem u pseudo-polinomnom vremenu [5], a sam algoritam se može pronaći u [9].
Problem se može iskazati preciznije:

$$(PR) \quad z = max \sum_{i=1}^{n} v_i x_i$$

$$t.d. \quad \sum_{i=1}^{n} w_i x_i \leq W$$

$$x_i \in \{0, 1\} \quad i = 1, ..., n$$

Pri čemu su svi $w_i, v_i$ i $W$ pozitivni, a s $x_i$ je označeno pripada li uređeni par $((w_i, v_i)$ skupu $X$.
U skladu s takvim zapisom problema, njemu apsolutni robusni ekvivalent se iskazuje:

$$(PR)_A \quad z_A = \max y$$
$$t.d \sum_{i=1}^{n} v_i^s x_i \geq y \quad s \in S$$
$$\sum_{i=1}^{n} w_i x_i \leq W$$
$$x_i \in \{0, 1\} \quad i = 1, ..., n$$

Problem se najčešće susreće u budžetiranju kapitala. Za danih $n$ projekata s očekivanom zaradom $v_i$ i cijenom početne investicije $W_i$, potrebno je budžetirati raspoloživi kapital $W$ na najbolji mogući način. Nesigurnost proizlazi iz nesigurnosti u pritok kapitala i stanje tržišta.
Važno je napomenuti da u svakom scenariju težine predmeta ostaju fiksirane, a cijena predmeta je nesigurna.

## 5.1 Složenost robusnog problema naprtnjače

Problem naprtnjače je već sam po sebi NP-težak, međutim u slabom smislu. U ovom odjeljku se pokazuje teorem koji daje uvjete uz koje je apsolutni robusni problem naprtnjače jako NP-težak.
Ispostavlja se kako je je apsolutni robusni problem naprtnjače jako NP-težak za neograničen broj scenarija, a ta se tvrdnja pokazuje svođenjem problema pokrivača skupa na $(PR)_A$.

**Teorem 5.1.1.** *Apsolutni robusni problem naprtnjače je jako NP-težak ako skup scenarija $S$ nije ograničen.*

*Dokaz.* Problem pokrivača skupa se svodi na problem $(PN)_A$. Neka su:

$$n = |P|,$$
$$S = U,$$
$$v_i^s = \delta_{is} \quad i = 1, ..., n; s \in S,$$
$$w_i = 1 \quad i = 1, ..., n,$$
$$W = k.$$

Gdje je $\delta_{is}$ varijanta Kroneckerovog simbola, to jest $\delta_{is} = 1$ kada kada $i$-ti podskup iz problema pokrivača sadrži $s$-ti element od $U$. Jasno je da će problem pokrivača skupa imati rješenja ako i samo ako problem $(PN)_A$ ima rješenja za $z_A(KP) \geq 1$. To slijedi iz činjence da se robusno rješenje sastoji od podskupova koji pokrivaju sve elemente od $U$. Da postoji neki nepokriveni element $s$ u odgovarajućem scenariju bi i vrijednost rješenja bila 0 pa bi to bio i minimum po svim scenarijima. Problem je dakle NP-težak jer je dano pridruživanje polinomno, a i svi parametri pridruživanja također.

□

**Napomena 5.1.2.** *Kada je skup scenarija S ograničen konstantom, tada je i $(PN)_A$ slabo NP-težak.*

Iako je robusni problem naprtnjače jako NP-težak, ipak postoji pseudo-polinomni algoritam koji ga računa u slučaju kada je veličina skupa scenarija ograničena konstantom. U prethodnim poglavljima se ovakva tvrdnja iskazivala kao teorem i formalno dokazivala davanjem algoritma, međutim, problem naprtnjače je slabo NP-težak i u konvencionalnom obliku tako da će tvrdnja biti obrazložena algoritmom, ali bez formalizacije teorema i dokaza.

$$g_k(d; \alpha_1, ..., \alpha_{|S|}) = \max_x \min_{s \in S} \{ \sum_{i=1}^{k} v_i^s x_i + \alpha_s : \sum_{i=1}^{k} \alpha_i x_i \leq d; x_i \in \{0, 1\}, i = 1, ..., k \}.$$

Funkcija $g_k(d; \alpha_1, ..., \alpha_{|S|})$ računa maksimalnu minimalnu vrijednost optimalnog izbora među prvih $k$ elemenata kada je kapacitet naprtnjače ograničen na $d$ uz oznaku $\alpha_s$ za odgovarajuću vrijednost sume iz scenarija s. Inicijalni uvjet je dan sa:

$$g_0(d; \alpha_1, ..., \alpha_{|S|}) = \begin{cases} -\infty, & d < 0 \\ \min_{s \in S} \alpha_s, & 0 \leq d \leq b. \end{cases}$$

A rekurzivna relacija sa:

$$g_{k+1}(d; \alpha_1, ..., \alpha_{|S|}) = max\{g_k(d; \alpha_1, ..., \alpha_{|S|}), g_k(d - a_{k+1}; \alpha_1 + v_{k+1}^1, ..., \alpha_{|S|} + k_{k+1}^{|S|})\}$$

Optimalna vrijednost je dobivena računajući $g_n(b; 0, ..., 0)$. Pseudokod algoritma se nalazi u nastavku:

**Algorithm 4** RobusniPN($a_i$ težine predmeta; $v_i$ vrijednosti predmeta; b težinski kapacitet naprtnjače)

**for** $d = 0$ to $b$ **do**
  **for** $\alpha_1 = 0$ to $\sum_{i=1}^{n} v_i^1$ **do**
    ...
    **for** $\alpha_{|S|} = 0$ to $\sum_{i=1}^{n} v_i^{|S|}$ **do**
      $g_0(d; \alpha_1, ..., \alpha_{|S|}) = \min_{s \in S} \alpha_s$
    **end for**
    ...
  **end for**
**end for**
**for** $k = 0$ to $n - 1$ **do**
  **for** $d = 0$ to $b$ **do**
    **for** $\alpha_1 = 0$ to $\sum_{i=1}^{n} v_i^1$ **do**
      ...
      **for** $\alpha_{|S|} = 0$ to $\sum_{i=1}^{n} v_i^{|S|}$ **do**
        **if** $d \geq a_{k+1}$ **then**
          $g_{k+1}(d; \alpha_1, ..., \alpha_{|S|}) = max\{g_k(d; \alpha_1, ..., \alpha_{|S|}), g_k(d - a_{k+1}; \alpha_1 + v_{k+1}^1, ..., \alpha_{|S|} + k_{k+1}^{|S|})\}$
        **else**
          $g_{k+1}(d; \alpha_1, ..., \alpha_{|S|}) = g_k(d; \alpha_1, ..., \alpha_{|S|})$
        **end if**
      **end for**
      ...
    **end for**
  **end for**
**end for**
**return** $g_n(b; 0, ..., 0)$

Vremenska složenost tog algoritma je $O(ng\pi_{s \in S}(\sum_{i=1}^{n} v_i^s))$ što je ograničeno sa $O(nbL^{|S|})$ Budući da je $|S|$ ograničen, algoritam je pseudo-polinomne vremenske složenosti.

**Primjer 5.1.3.** *Konkretan primjer apsolutnog robusnog problema pakiranja naprtnjače dan je na 2 scenarija za naprtnjaču težinskog kapaciteta 8 i 4 predmeta:*

| $i$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $a_i$ | 3 | 6 | 2 | 4 |
| $v_i^1$ | 5 | 7 | 4 | 5 |
| $v_i^2$ | 5 | 4 | 2 | 3 |

*Kako bi se izračunala optimalna vrijednost* $g_0(8;0,0)$ *ona se prvo inicijalizira na* 0*, a zatim se računaju:*

$$g_0(5;3,5) = 3, g_0(2;7,4) = 4, g_0(6;4,2) = 2, g_0(3;7,7) = 7,$$

$$g_0(0;11,6) = 6, g_0(4;5,3) = 3, g_0(1;8,8) = 8, g_0(2;9,5) = 5$$

*Uz inicijalni uvijet na početku, uzimanjem maksimuma među parovima se u drugom koraku dobiva:*

$$g_1(8;0,0) = 3, g_1(2;7,4) = 4, g_1(6;4,2) = 7,$$
$$g_1(0;11,6) = 6, g_1(4;5,3) = 8, g_1(2;9,5) = 5$$

*Pa u narednim koracima:*

$$g_2(8;0,0) = 4, g_2(6;4,2) = 7, g_2(4;5,3) = 8, g_2(2;9,5) = 5$$
$$g_3(8;0,0) = 7, g_3(4;5,3) = 8$$
$$g_4(8;0,0) = 8$$

*Optimalno rješenje je* $g(8;0,0) = 8$ *za izbor prvog i četvrtog predmeta.*

# Poglavlje 6

# Implementacija algoritma robusnog najkraćeg puta

Za implementaciju je izabran jezik Ruby. Razlog tomu je što sintaksom podsjeća na engleski jezik i lako je razumjeti što se u algoritmima događa. Implementira se algoritam 3.1 koji je ranije demonstriran, ali ograničen na 4 scenarija. Algoritam rješava probleme i za manji broj scenarija, samo se ponavljaju neki scenariji više puta sve dok ih ne bude 4.

Prvi dio algoritma računa iznos najduljeg puta u grafu u svakom scenariju. Graf je zapisan kao trodimenzionalna lista bridova između vrhova. Vrijednost brida između vrhova $v_{k,i}$ i $v_{k+1,j}$ zapisuje se u obliku $c[i][j][k]$, a ako je zapisan negativan broj, onda brid ne postoji. Algoritmu se predaje niz takvih grafova koji predstavljaju različite scenarije koji se promatraju.

```ruby
def longest_path(c)
  n = c[0][0].length - 1
  m = c[0].length
  longest = Array.new(n, Array.new((n + 2), 0))
  for k in (n).downto(0)
    for i in 0...m
      for j in 0...m
        if c[i][j][k] > -1
          longest[i][k] = [c[i][j][k] + longest[j][k+1].to_i, longest[i][k]].max
        end
      end
    end
  end

  longest[0].max
end
```

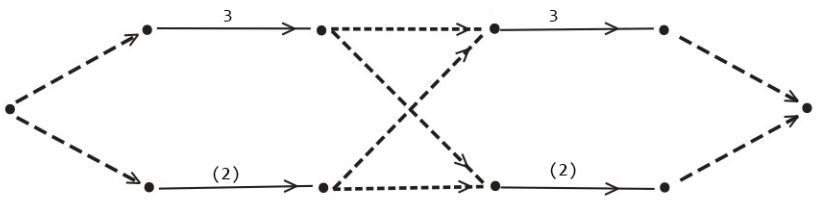
Slika 6.1: Implementacija funkcije koja računa najdulji put u jeziku Ruby

Algoritam nastavlja s inicijalizacijom polja vrijednosti i onda inicijalizacijom rekurzije. Polje se inicijalizira na $Infinity$ budući da se kasnije promatraju minimumi. To znači da ako algoritam vrati vrijednost $Inifinity$, onda nema puta od početnog do završnog vrha. Polje je tih dimenzija kako nikada indeksi ne bi postali preveliki. U najgorem slučaju je moguće da je samo jedan put, koji je ujedno i najdulji, pa bi time vrijednost $alpha1 + graph[s][i][j][k]$ postala dvostruko veća od vrijednosti $alpha1$ i tako za svaki scenarij.

```
L = Array.new()

m = graph[0][0][0].length - 1
n = graph[0][0].length
L[1] = longest_path(graph[0])
L[2] = longest_path(graph[1])
L[3] = longest_path(graph[2])
L[4] = longest_path(graph[3])

f = Array.new(m+3, Array.new(m+3, Array.new(2 * L[1], Array.new(2 * L[2], Array.new(2 * L[3], Array.new(2 * L[4], Float::INFINITY))))))


for alpha1 in 0...L[1] do
  for alpha2 in 0...L[2] do
    for alpha3 in 0...L[3] do
      for alpha4 in 0...L[4] do

        for i in 0..m+2 do
          for j in 0..m+2 do
            f[i][j][alpha1][alpha2][alpha3][alpha4] = [alpha1, alpha2, alpha3, alpha4].max
          end
        end

      end
    end
  end
end
array = []
for k in m.downto(0) do
  for i in 0...n do
    for alpha1 in 0...L[1] do
      for alpha2 in 0...L[2] do
        for alpha3 in 0...L[3] do
          for alpha4 in 0...L[4] do
            for j in 0...n
              array << f[k+1][j][alpha1 + graph[0][i][j][k]][alpha2 + graph[1][i][j][k]]
                       [alpha3 + graph[2][i][j][k]][alpha4 + graph[3][i][j][k]].to_f if graph[0][i][j][k] > -1
            end
          f[k][i][alpha1][alpha2][alpha3][alpha4] = array.min.to_f
          end
        end
      end
    end
  end
end
puts "-----------------"
puts f[0][0][0][0][0][0]
```

Slika 6.2: Implementacija algoritma u jeziku Ruby

Kod je testiran na slojevitim mrežama kakve su opisane u 3.1.2 budući da je za njih poznato rješenje. Obzirom da se radi o samo dva scenarija, u kodu će biti potrebno neke zapisati više puta.

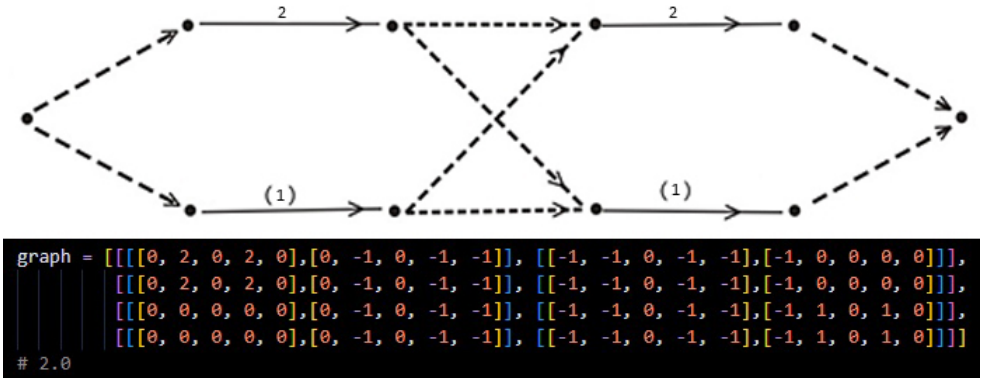



Slika 6.3: Graf na kojemu se testira implementacija (vrijednosti u zagradama iznose 0 u prvom scenariju, a 2 u drugom, analogno je s gornjim vrijednostima)

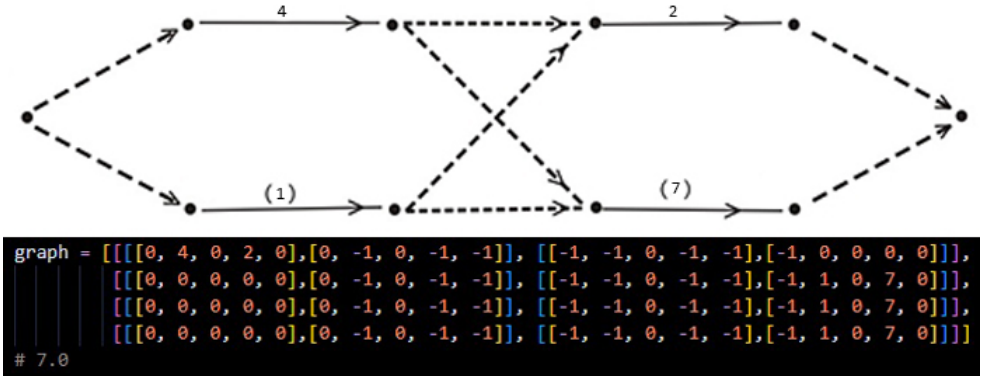
```
graph = [[[[0, 3, 0, 3, 0],[0, -1, 0, -1, -1]], [[-1, -1, 0, -1, -1],[-1, 0, 0, 0, 0]]],
         [[[0, 0, 0, 0, 0],[0, -1, 0, -1, -1]], [[-1, -1, 0, -1, -1],[-1, 2, 0, 2, 0]]],
         [[[0, 0, 0, 0, 0],[0, -1, 0, -1, -1]], [[-1, -1, 0, -1, -1],[-1, 2, 0, 2, 0]]],
         [[[0, 0, 0, 0, 0],[0, -1, 0, -1, -1]], [[-1, -1, 0, -1, -1],[-1, 2, 0, 2, 0]]]]
```

Slika 6.4: Zapis grafa u kodu

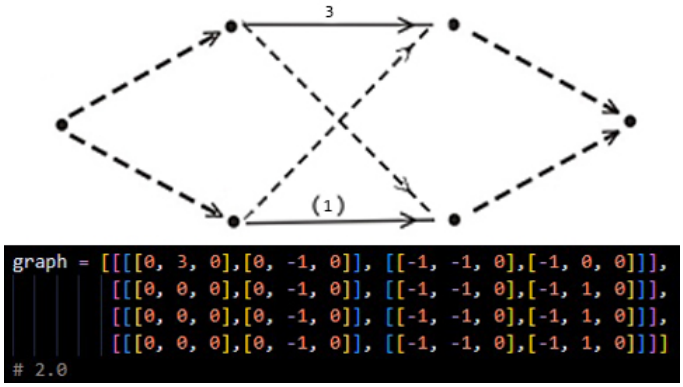
Na gornjem primjeru dobiven je rezultat 5, što je ujedno i polovica sume svih težina bridova. U nastavku su prikazani još neki testni primeri grafova s pripadajućim rezultatima.



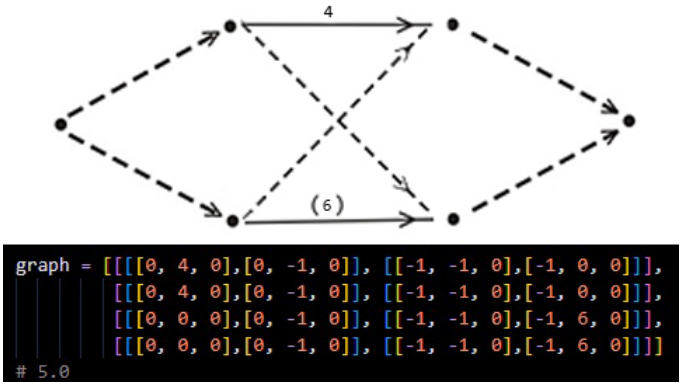

Slika 6.5: Graf sa 6 slojeva i manjim vrijednostima




Slika 6.6: Graf sa 6 slojeva




Slika 6.7: Graf s 4 sloja i manjim vrijednostima




Slika 6.8: Graf s 4 sloja

# Zaključak

Kada su u pitanju problemi na diskretnim podacima koji su neizvjesni, robusno-optimalno rješenje će osigurati da nikada nije izabran najgori slučaj. Dani su algoritmi za svega 3 takva problema, a od njih je implementiran samo jedan. Obzirom da su izabrani problemi koji postaju NP-teški u svojoj robusnoj varijanti, provjeravaju se neki "minimalni" uvjeti koji su dovoljni kako za problem više ne bi postojao algoritam koji ih rješava u polinomnom vremenu. Jedan od tih uvjeta je ograničen broj scenarija, a za implementaciju je izabran konkretan broj scenarija. Za svaki izabrani broj scenarija moguće je napisati novu implementaciju, a tako onda i za svaki izabrani problem i varijantu robusne optimizacije (apsolutnu, devijacijsku ili relativnu). U dokazima i pseudokodu algoritma je jasno da su u pitanju varijacije istog algoritma i da vremenska kompleksnost i logika ostaju iste, ali implementacije postaju kompliciranije svakom dodatnom promjenom.

# Bibliografija


[1] Hassene Aissi, Cristina Bazgan i Daniel Vanderpooten, *Min–max and min–max regret versions of combinatorial optimization problems: A survey*, European journal of operational research **197** (2009), br. 2.

[2] Stephen A Cook, *The complexity of theorem-proving procedures*, Proceedings of the third annual ACM symposium on Theory of computing, 1971.

[3] Edsger Wybe Dijkstra, *A note on two problems in connexion with graphs:(Numerische Mathematik, 1 (1959), p 269-271)*, (1959).

[4] Adam Kasperski i Paweł Zieliński, *Robust discrete optimization under discrete and interval uncertainty: A survey*, Robustness analysis in decision aiding, optimization, and analytics (2016).

[5] Panos Kouvelis i Gang Yu, *Robust discrete optimization and its applications*, sv. 14, Springer Science & Business Media, Berlin, 2013.

[6] Leonid Anatolevich Levin, *Universal sequential search problems*, Problemy peredachi informatsii **9** (1973), br. 3.

[7] Michael Sipser, *Introduction to the Theory of Computation*, ACM Sigact News **27** (1996), br. 1.

[8] Marko Špoljarec, *Efikasni algoritmi za rješavanje robusnih varijanti problema toka u mreži*, Disertacija, University of Zagreb. Faculty of Science. Department of Mathematics, 2018.

[9] Paolo Toth, *Dynamic programming algorithms for the zero-one knapsack problem*, Computing **25** (1980), br. 1.

[10] Mladen Vuković, *Složenost algoritama*, (2019), `https://www.pmf.unizg.hr/_download/repository/SA-skripta-2019.pdf`.

# Sažetak

U ovom radu opisane su vremenske složenosti nekih robusnih diskretnih problema optimizacije.
Nakon navođenja osnovnih pojmova, problema i definicija ispituju se robusne varijante problema najkraćeg puta, najmanjeg razapinjućeg stabla i pakiranja naprtnjače. Uvode se pojmovi klasa P i NP te P-teških i NP-teških problema.
Proučavaju se navedeni problemi s određenim ograničenjima u usporedbi sa svojim konvencionalnim varijantama. Definira se slojevita mreža i pokazuje se da su apsolutni i devijacijski robusni problemi najkraćeg puta slabo NP-teški čak i u slojevitim mrežama širine dva i sa samo dva scenarija. Nakon dokazivanja tih tvrdnji dani su i algoritmi koji rješavaju probleme na slojevitim mrežama u pseudo-polinomnom vremenu. Problemi su jako NP-teški kada broj scenarija nije ograničen. Nakon što su te tvrdnje dokazane, u slijedećem poglavlju se uvodi pojam rešetkastog grafa. Dokazuje se je apsolutni problem najmanjeg razapinjućeg stabla slabo NP-težak na poprilično restringiranom rešetkastom grafu sa samo dva scenarija. Dan je algoritam koji rješava problem na rešetkastom grafu u pseudo-polinomnom vremenu. Zatim se dokazuje da je apsolutni problem najmanjeg razapinjućeg stabla jako NP-težak kada je broj scenarija neograničen. Nadalje, dokazuje se da je apsolutni robusni problem naprtnjače jako NP-težak na neograničenom broju scenarija i dan je pripadajući algoritam.
Zadnje poglavlje sadrži konkretnu implementaciju algoritma koji rješava apsolutni robusni problem najkraćeg puta u jeziku Ruby s pripadajućim primjerima.

# Summary

In this paper, the time complexities of some robust discrete optimization problems are described.
After stating the basic concepts, problems and definitions, robust variants of the shortest path, smallest spanning tree and knapsack packing problems are examined. The concepts of classes P and NP as well as P-hard and NP-hard problems are introduced.
The aforementioned problems are studied with certain limitations compared to their conventional variants. A layered network is defined and it is shown that the absolute robust and robust deviation shortest path problems are weakly NP-hard even in layered networks of width two and with only two scenarios. After proving these statements, algorithms for the layered network problem for each are given. These problems are strongly NP-hard when the number of scenarios is not limited. After these claims have been proven, the next chapter introduces the concept of a grid graph. The absolute minimum spanning tree problem is proven to be weakly NP-hard on a fairly restricted grid graph with only two scenarios. An algorithm is given that solves the problem in pseudo-polynomial time for a grid graph and a bounded number of scenarios. Then it is proven that the absolute minimum spanning tree problem is strongly NP-hard when the number of scenarios is unbounded. The second to last chapter proves that the absolute robust knapsack problem is strongly NP-hard on an unlimited number of scenarios and the corresponding algorithm is given.
The last chapter contains a concrete implementation of an algorithm that solves the absolute robust shortest path problem in the Ruby programming language with associated examples.

# Životopis

Rođen sam 28. rujna 1996. godine u Zagrebu. Pohađao sam Osnovnu školu Žuti brijeg u zagrebačkoj Dubravi gdje sam odrastao i gdje živim. 2011. godine školovanje nastavljam upisom XV. gimnazije u Zagrebu. Uz osnovnu školu i većinu gimnazije sam se bavio karateom u kojemu sam dostigao crni pojas I. dan.
2015. godine upisao sam Preddiplomski sveučilišni studij Matematika na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta u Zagrebu, a 2019. godine, odmah po završetku tog studija sam upisao Diplomski sveučilišni studij Računarstvo i matematika na istom odsjeku.
Izuzev toga, od 2008. godine sam član izviđačkog pokreta, a od 2012. godine u njemu i aktivno volontiram kao jedan od voditelja u 28. Samostalnoj družini izviđača "Dubrava". Udruga je članica Saveza izviđača Hrvatske i Svjetskog skautskog pokreta te kroz njih pohađam i organiziram dodatne seminare, tečajeve i edukaciju. 2021. godine sam završio edukaciju za Wood Badge 2.

Osim dodatnih edukacija sviram gitaru, a 2020. godine sam bio jedan od nositelja i izvoditelja projekta "Izviđač brine i čuva: Mentalno zdravlje" kojeg su financirale Europske snage solidarnosti.