UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Finančna matematika – 1. stopnja

Sabrina Calcina in Jan Črne

Algoritmi in množice neodvisnosti za podatkovno vodene robustne probleme najkrajših poti

Finančni praktikum

Mentor: prof. dr. Sergio Cabello in asist. dr. Janoš Vidali

Ljubljana, 2020

Kazalo

Algoritmi in množice neodvisnosti za podatkovno vodene robustne probleme najkrajših poti

Povzetek

V prejktu smo se ukvarjali z robustnimi problemi najkrajših poti. To so problemi, katerih cilj je najti pot, ki optimizira najslabše delovanje v množici negotovosti. Množica negotovosti je množica, ki vsebuje vse scenarije cen povezav, zgeneriranih ali zabeleženih na podlagi opažanj. Predpostavka tega problema je, da so množice negotovosti podane s strani strokovnjakov, ki povedo obliko in velikost le te. Množice negotovosti sva v projektu naključno generirala. Meriteve uporabljene v članku, na katerem sloni najino delo, temeljijo na resničnih podatkih iz meritev prometa. Na podlagi teh podatkov, ki torej vsebujejo cene povezav nekega usmerjenega grafa, nato po premisleku izberemo primerno množico negotovosti, tako da primerjamo uspešnost dobljenih robustnih poti. Na podlagi že znanih eksperimentov o učinkovitosti elipsoidne negotovosti, se nato osredotočimo na elipsoidne množice negotovosti na katerih preiskušamo dva algoritma za iskanje najcenejše povezave med začetnim in končnim vozliščem.

1 Uvod

Za klasične probleme najkrajših poti v uličnih omrežjih so bile dosežene znatne pospešitve v primerjavi s standardnim Dijkstrovim algoritmom. Zahvala gre tehnikam novejših algoritmov, ki omogočajo uporabo informacij v realnem času, tudi v velikih omrežjih. Kljub temu je večina robustnih problemov z najkrajšimi oz. najcenejšimi potmi časovno zahtevna in optimizacija v realnem času ni na voljo. Za oblikovanje robustnega problema je tako treba imeti opis vseh možnih in ustreznih scenarijev, na katere naj bi se pripravili.

Prvi članek, ki sledi drugačni perspektivi problema najkrajše poti, je izšel leta 2017. Gre za robustno optimizacijo, ki temelji na podatkih, kjer je gradnja negotove množice na podlagi surovih opazovanj del robustnega problema optimizacije. Na podlagi realnih opazovanj mesta Chicago so tako izračunali ustrezne robustne rešitve in izvedli poglobljeno analizo njihove uspešnosti.

V delu se osredotočimo na primer elipsoidne negotovosti za katero preiskušamo algoritme.

2 Opis problema

- Za začetek imamo usmerjen graf $G = (V, A)$, kjer je V je množica vozljev, ter A množica povezav. Za vsako povezavo je znana njena ceno, ki bo v našem primeru čas, potreben za prehod te povezave.

- Cilj je najti pot v grafu, ki minimizira čas potreben za pot od začetnega do končnega ogliščа. Za razliko od nam do zdaj znanih primerov, kjer so bile cene povezav podane eksaktno, so v našem primeru te podane z množico opazovanj teh povezav.

- Na teh množicah lahko sedaj poiščemo takšno pot, da bomo zminimizirali časovno najugodnejšo pot v primeru najslabšega scenarija. Npr. želimo poiskati najkrajšo pot v mestu od ene točke do druge, v primeru, maksimalne zasedenosti cest. Torej takrat, ko za vožnjo po njih porebujemo največ časa.

- Osredotočili se bomo na izvajanje algoritma za iskanje teh poti na elipsoidinih množicah negotovosti. Le te ponudijo najbolše razmerje med maksimalno in povprečno potjo, ter ponudijo zadovoljivo časovno zahtevnost algoritma.

- Prvotno sva tako izvedla Naivni algoritem, kasneje pa še Izboljšan algoritem.

3 Množice negotovosti najkrajših poti

Naj bo $G = (V, A)$ graf, kjer z V označimo vozlišča grafa in z A poti. Za vsako pot $e \in A$ poznamo čas potovanja $c_e \geq 0$. Začetno oglišče označimo s s, končno pa s t. Cilj je najti najkrajšo pot, ki minimizira celoten čas potovanja, podan kot vsoto vseh časov na določeni povezavi, ki je del poti. Formalno to zapišemo kot

$$\min\{c^t x : x \in \chi\},$$

kjer je $\chi \subseteq \{0,1\}^n$ in $n = |A|$.

Predpostavimo tudi, da čas potovanja ni natančno znan, vendar je le ta podan kot množica R, kjer je $R = \{c^1, \ldots, c^N\}$.

Na podlagi teh podatkov generiramo množice negotovosti U, ki jih uporabljamo pri robustnih problemih najkrajših poti

$$\min\{\max_{c \in U} c^t x : x \in \chi\}.$$

Torej iščemo pot, ki minimizira najslabše možnosti cen, glede na vse cene v U.

Naj bo sedaj $\hat{c} = \frac{1}{N} * \sum_{i \in [N]} c^i$, kjer je $[N] = \{1, \ldots, N\}$. Elipsoidno negotovost za parameter λ definiramo kot

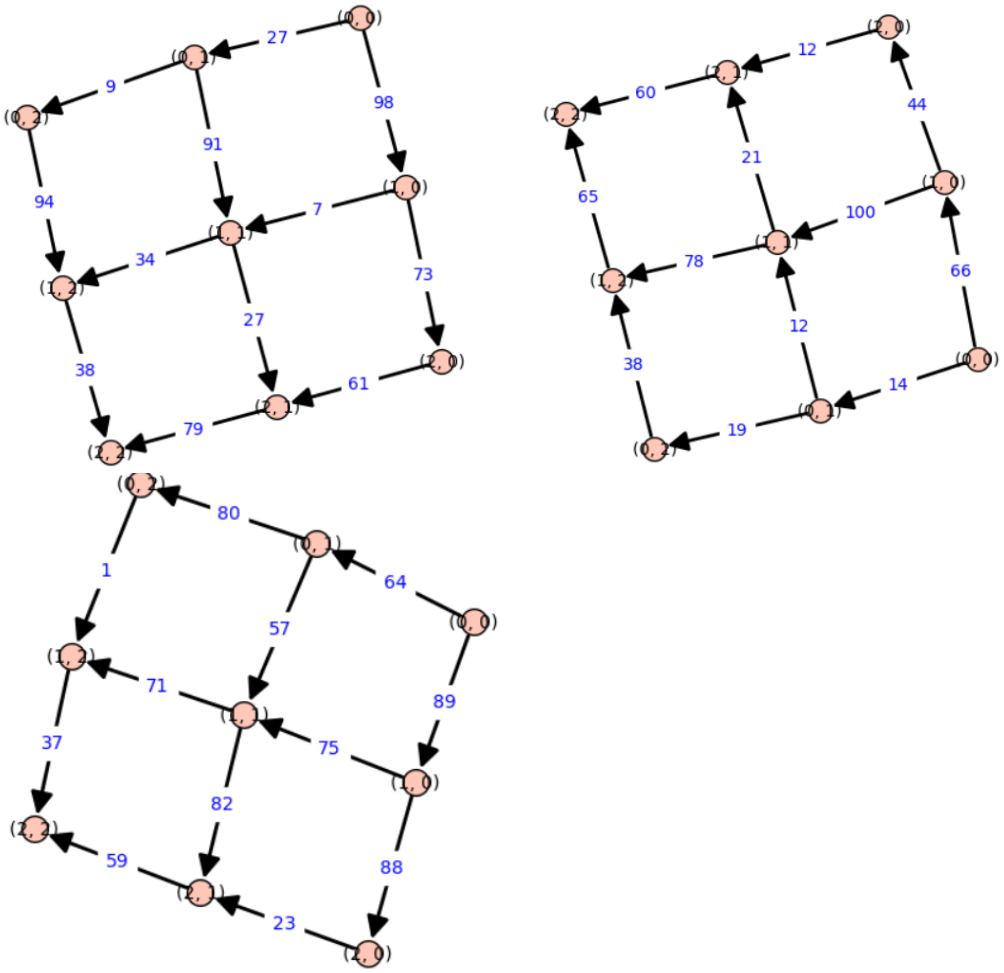
$$U^E = \{c : (c - \hat{c})^t \Sigma^{-1} (c - \hat{c} \leq \lambda\},$$

kjer je $\Sigma = \frac{1}{N} \sum_{i \in [N]} (c^i - \hat{c})(c^i - \hat{c})^t$.

Poleg elipsoidne negotovosti, poznamo tudi negotovost konveksnega trupa, intervalsko negotovost ter negotovost permotohula in druge.

4 Priprava

Najprej sva generirala množice negotovosti velikosti 50, torej za vsako povezavo v grafu je bilo danih 50 različnih cen oziroma opažanj. Cene so porazdeljene enakomerno, vrednosti so iz množice od $\{1, \ldots, 100\}$. Vsaki povezavi sva nato naračunala povprečno ceno, povprečja vseh teh povezav so zbrana v vektorju $\hat{c}$.



Slika 1: Prikaz na mrežatih grafih velikosti 3×3 za tri opažanja

V najini nalogi sva obravnavala dva algoritma , ki iščeta najcenejšo robustno rešitev. Algoritma sva prizkušala na usmerjenih mrežasti grafih, v najinem primeru na grafih z od 3×3, do 9×9 vozliščih. Za vsakega od

teh grafov sva iskala pot od začetnega vozlišča $(1,1)$ do skrajno robnega vozlišča s koordinatami (V,V). Za potrebo algoritmov, sva morala najti vse možne poti med tema dvema vozliščema, če je poti med vozliščema p, sva potem priredila vektorje iz množice $X = \{x^1, \ldots, x^p\}$, vektorji so sestavljeni iz elementov 1, v kolikor je povezava v poti uporabljena in 0, če je izpuščena. Velja omeniti še, da je število povezav mrežastih grafov in število poti med skrajnima vozliščema neposredno odvisno števila vozlišč grafa.

Pred pričetkom uporabe algoritmov sva pridobila še diagonalne elemente matrike Σ, ki je naračunana iz najinih množic negotovosti, torej predstavlja najino elipsoidono množico negotovosti.

5 Predstavitev optimizacijskega problema elipsoidnih množic negotovosti

Ker so elipsoidne množice negotovosti zelo stabilne in splošno zelo učinokovite, glede na ostale navedene negotovosti, smo se osredotočili na le te.

Sprva opišimo učinkovit algoritem za reševanje problema robustne najkrajše poti, če je množica negotovosti podana kot osem paralelni elipsoid. Formulacija tega problema je

$$\begin{aligned} \min \quad & \hat{c}^t x + z \\ \text{p.p.} \quad & z^2 \geq (x^t \Sigma x) \\ & x \in \chi, \end{aligned}$$

kjer Σ predstavlja diagonalo matrike, ki specificira obliko in velikost elipsoida.

Naj bo d diagonala matrike Σ. Ker velja, da je x binaren vektor in s tem $x_i^2 = x_i$, poleg tega pa velja, da je $(x^t \Sigma x)^2 = d^t x$, lahko problem zreduciramo na

$$\begin{aligned} \min \quad & \hat{c}^t x + \sqrt{d^t x} \\ \text{p.p.} \quad & x \in \chi. \end{aligned}$$

Problem lahko transformiramo na bikriterijski optimizacijski problem, podan kot

$$\begin{aligned} \min \quad & \begin{pmatrix} \hat{c}^t x \\ d^t x \end{pmatrix} \\ \text{p.p.} \quad & x \in \chi. \end{aligned}$$

Rešitev robustnega problema je v resnici tudi rešitev bikriterijskega optimizacijskega problema. Rešitvi zgornjega problema x^* rečemo *učinkoviti ekstrem*, če obstajata α_0 in α_1, velja $0 \leq \alpha_0 < \alpha_1 \leq 1$. Tako da za vse $\alpha \in [\alpha_0, \alpha_1]$ ne obstaja neka druga rešitev x', za katero bi veljalo $(\alpha c + (1-\alpha)d)^t x' < (\alpha c + (1-\alpha)d)^t x^*$.
To pomeni, da je dovolj, če rešimo naslednji problem

$$\begin{aligned} \min \quad & \alpha c + (1-\alpha)d)^t x \\ \text{p.p.} \quad & x \in \chi. \end{aligned}$$

Ker je rešitev lahko precej zahtevna, se omejimo na računanje podmnožic vseh učinkovitih ekstremnih rešitev problema. Med najdenimi rešitvami se nato izbere najboljša glede na robustno ciljno funkcijo.

Oba algoritma iščeta učinkovite oz. učinkovito ekstremno rešitev s pomočjo računanja optimizacije leksikografskega problema

$$\begin{aligned} \text{arglexmin} \quad & (\hat{c}^t x, d^t x) \\ \text{p.p.} \quad & x \in \chi. \end{aligned}$$

pri obeh je potrebno naračunati x_l in x_r, kjer eden minimizira skalarni produkt po $\hat{c}$, drugi pa po d, eden torej poišče najcenejšo pot po povprečjih povezav, drugi pa po diagonalnih vrednostih matrike.

Naivni algoritem učinkovite ekstremne rešitve išče rekurzivno, tako da vedno znova poišče najcenejšo pot, jo nato primerja z prejšnjo najcenejšo potjo in v kolikor pride do izbolšave doda v množico ekstremnih rešitev novo vrednost.
Izboljšani algoritem, pa se s pomočjo x_l in x_r omeji na neko območje iskanja rešitev, območje iskanja z vsako novo najdeno izboljšano rešitvijo zmanjša, ko je območja iskanja prazno ali zadosti majhno pa se algoritem ustavi ter vrne najugodnejšo pot.

6 Naivni algorotem in njegovi rezultati

Na začetku najinega dela, sva se reševanja problema lotila z uporabo naivnega algoritma. Ko sva naivni algoritem pognala na mrežnih grafih, je le ta vrnil množico učinkovitih ekstremnih rešitev, torej poti, katere dokaj dobro rešijo robustni problem najcenejših poti. Poleg tega, naju je zanimala še časovna zahtevnost algoritma in kolikokrat se je znotraj algoritma reševal problem najkrajših poti. Za vsak mrežast graf z $n \times n$ vozlišči, sva naredila

30 ponovitev, ter izračunala povprečno vrednost poganjanja. Rezultate sva za $n = 3, 4, 5, 6, 7, 8, 9$ predstavila v spodnji tabeli.

Naj bo Graf n, graf z $n \times n$ vozlišči, čas je podan v sekundah.

Graf 3	Graf 4	Graf 5	Graf 6	Graf 7	Graf 8	Graf 9
0.00078	0.00533	0.02949	0.24377	1.17852	7.36974	40.05113

Komputacijski čas je vračal pričakovane vrednosti in sicer, za grafe z večjimi vozlišči je potreboval več časa. Za grafe z 9×9 vozlišči je potreboval nekaj več kot 40 sekund, medtem ko je manjše grafe algoritem predelal zelo hitro. Komputacijski čas bova kasneje primerjala z komputacijskim časom izboljšanega algoritma.

Poleg tega, sva beležila tudi kolikokrat je za posamezno velikost grafa algoritem reševal problem najkrajših poti. Število reševanj je predstavljeno v spodnji tabeli.

Naj bo NP n, število ki nam pove, koliko krat je algoritem pregledal celotno množico x, za katero je potem našel najkrajšo možno pot glede na dan optimizacijski problem za grafe z $n \times n$ vozlišči.

NP 3	NP 4	NP 5	NP 6	NP 7	NP 8	NP 9
2.07	2.47	4.07	5.73	6.53	8.2	8.93

Za večje grafe je algoritem večkrat reševal problem, kar je torej podaljšalo komputacijski čas. Tudi rekurzivne klice bomo kasneje primerjali z izboljšanim algoritmom.

7 Izboljšani algoritem in njegovi rezultati

Na enakih grafih kot pri naivnem algoritmu, torej mrežastih, smo poskus ponovili na izboljšanem algoritmu. Naredili smo 30 ponovitev, ter izračunali povrpečno vrednost poganjanja. Rezultate za $n = 3, 4, 5, 6, 7, 8, 9$ smo predstavili v spodnji tabeli.

Naj bo Graf n, graf z $n \times n$ vozlišči, komputacijski čas je podan v sekundah.

Graf 3	Graf 4	Graf 5	Graf 6	Graf 7	Graf 8	Graf 9
0.11760	0.01872	0.02816	0.16980	0.64850	2.38989	17.06119

Tudi izboljšani algoritem je, pričakovano za večje grafe z več potmi, v povprečju potreboval dalj časa.

Poleg tega, smo ponovno beležili kolikokrat je za posamezno velikost grafa algoritem reševal problem najkrajših poti. Število reševanj je predstavljeno v spodnji tabeli.

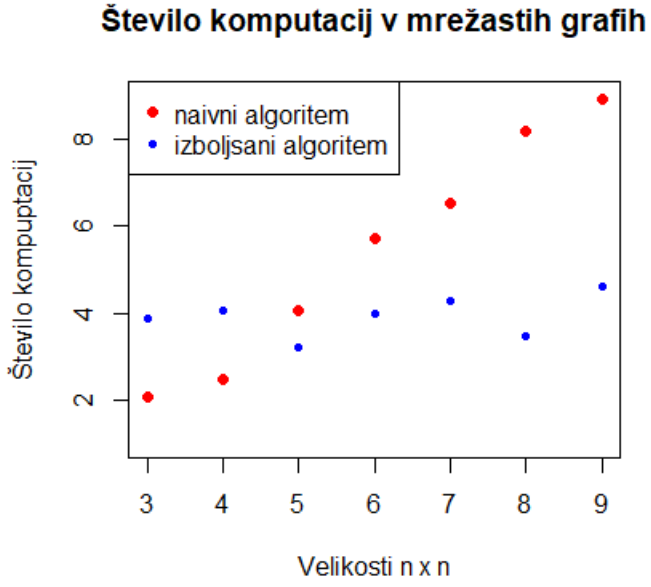
Naj bo NP n enako kot zgoraj, le da tokrat torej uporabljamo izboljšani algoritem.

NP 3	NP 4	NP 5	NP 6	NP 7	NP 8	NP 9
3.87	4.07	3.20	4.00	4.27	3.47	4.6

Izboljšani algoritem, zanimivo problema najkrajših poti ne računa nujno večkrat za grafe z več vozlišči, kar pomeni, da kot kaže, s svojimi pogoji že v prvih korakih določi dokaj dober približek najugodnejše poti za robusten optimizacijski problem.

8 Primerjava algoritmov

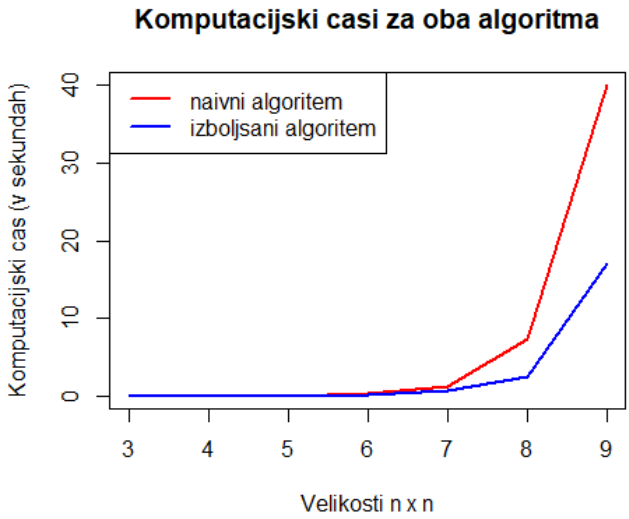
Opažanja smo predstavili na spodnjem grafu.



Slika 2: Število komutacijskih časov v mrežastih grafih

Razvidno je, da števila komputacij pri naivnem algoritmu naraščajo bistveno hitreje kot kot pri izboljšanem, kjer povečave skoraj ni bilo. Število komputacij pri 30 ponovitvah $9 \ldots 9$ grafov, je tako število komputacij v povprečju nekaj več kot 4, pri naivnem pa skoraj 9.

Komputacijske čase obeh algoritmov smo predstavili na spodnjem grafu.



Slika 3: Število komutacijskih časov v mrežastih grafih

Razvidno je, da izboljšani algoritem poti išče mnogo hitreje kot naivni. V iskanju robustnih rešitev, je časovno najzahtevnejše iskanje najkrajših poti. Izboljšani algoritem problem rešuje manjkrat, tako je pri istem številu vozlišč časovna zahtevnost nižja.