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 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 in razvijemo nov algoritem s katerim nato iščemo najcenejšo povezavo 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 in zagotavljanje hitrejšega algoritma.

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 definiramo kot

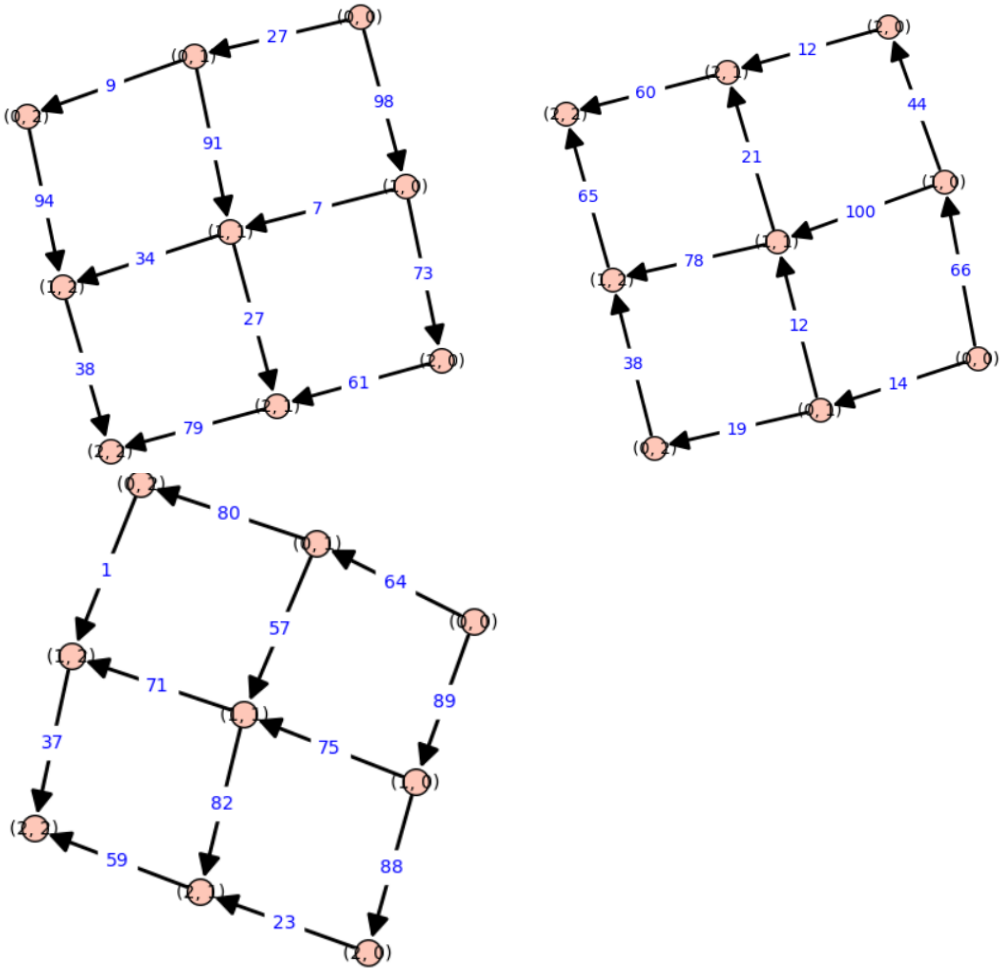
$$U^E = \{c : (c - \hat{c})^t \Sigma^{-1} (c - \hat{c} \leq \lambda\},$$

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.

4 Priprava

Najprej sva generirala množice negotovosti velikosti 50, torej za vsako povezavo v grafu je bilo danih 50 različnih cen oziroma opažanj. 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

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.

Pred pričetkom uporabe algoritmov sva pridobila še diagonalce matrike Σ, ki je naračunana iz najinih množic 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 osen 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. Rešitev zgornjega problema pa označimo z x^*, če obstajata α_0 in α_1, da je $0 \leq \alpha_0 < \alpha_1 \leq 1$. Tako za vse $\alpha \in [\alpha_0, \alpha_1]$ je rešitev

problema x', za katero velja $(\alpha c + (1-\alpha)d)^t x' < (\alpha c + (1-\alpha)d)^t x^*$.
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:

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

6 Naivni algorotem in njegovi rezultati

Na začetku najinega dela, sva se lotila problema z reševanjem naivnega algoritma. Ko sva naivni algoritem pognala na mrežnih grafih, sva dobila najkrajšo pot v grafu glede na vse možne cene na povezavah. Poleg tega, naju je zanimala še časovna zahtevnost algoritma, ter koliko rekurzij je algoritem izvedel. Sam algoritem sva pognala na 10 različnih grafih z $n \times n$ vozlišči, 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 pa podan v sekundah.

Graf 3	Graf 4	Graf 5	Graf 6	Graf 7	Graf 8	Graf 9
0.00262	0.00678	0.04248	0.28319	1.309803	7.54676	40.49303

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 izvedel zelo hitro. Komputacijski čas bova kasneje primerjala z komputacijskim časom izboljšanega algoritma.

Poleg tega, sva beležila tudi koliko krat je za posamezno velikost grafa algoritem rekurzivno klical. Število rekurzivnih klicev je predstavljeno v spodnji tabeli.

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

NP 3	NP 4	NP 5	NP 6	NP 7	NP 8	NP 9
2.4	3.2	4.6	5.8	6.8	7.8	8.2

Za večje grafe je tako algoritem izvedel več rekurzivnih klicov, kar je podaljšalo komputacijski čas, zato je pri veljih grafih komputacijski čas bistveno daljši. Tudi rekurzivne klice bomo kasneje primerjali z izboljšanih algoritmom.

7 Izboljšani algoritem in njegovi rezultati

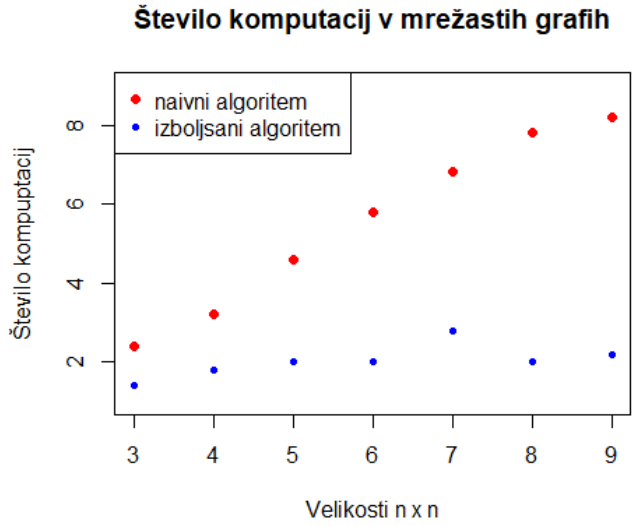
Naj bo Graf n, graf z $n \times n$ vozlišči, čas je pa podan v sekundah.

Graf 3	Graf 4	Graf 5	Graf 6	Graf 7	Graf 8	Graf 9
0.08271	0.22994	0.17297	0.56878	1.90194	32.74674	57.52192

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

NP 3	NP 4	NP 5	NP 6	NP 7	NP 8	NP 9
1.4	1.8	2.0	2.0	2.8	2.0	2.2

Primerjava komputacijskih časov med naivnim in izboljšanim algoritmom.



Slika 2: Število komutacijskih časov v mrežastih grafih