

Matching with low crossing numbers

Avtor:

* Jaka Basej

Ljubljana, 2021

# 

Kazalo:

[Uvod 3](#_Toc92213324)

[Celoštevilski linearni program 4](#_Toc92213325)

[Poimenovanje spremenjivk in zapis CLP 5](#_Toc92213326)

[Želimo: 5](#_Toc92213327)

[Pogoji: 5](#_Toc92213328)

[Celotni CLP: 6](#_Toc92213329)

[Komplikacije: 6](#_Toc92213330)

[Časovna zahtevnost: 7](#_Toc92213331)

# Uvod

Pri tej seminarski nalogi sem se lotil reševanja zahtevnega problema maksimalnega števila presekov premice z daljicami med pari točk. Program sem najprej napisal v pythonu z uporabo random kosa možnih razporeditev daljic. Nato sem nalogo resno izvedel in zapisal celoštevilski linearni program. Za konec pa dopisal program, ki dejansko zračuna število daljic za vse možne razporeditve daljic.

Problem v osnovi zgleda težak, ko se malo poglobiš vidiš, da je vprašanje zelo enostavno, ko pa začneš programirati rešitev se začnejo pa prikazovati težave. Pri linearnem programu rabim vpisat 2 vrst neznank z 4 integerji. Pri brout-force metodi pa zapisati vse možne razporeditve daljic v array. Obadva postopka sta zelo potratna s strani procesorske moči.

A picture containing clock, watch

Description automatically generated

Slika 1: Prikaz povezav in premice na točkah

# Celoštevilski linearni program

Problem bom zastavil s pomočjo celoštevilskega linearnega programiranja (CLP). Logika za reševanje problemov s CLP je zelo naravna, a v praksi računsko izjemno zahtevna, saj je CLP NP-težek proble in ni možnosti za izboljšanje unčikovitosti metode. Za probleme potrebujemo procesorsko moč, časovno zahtevnost bom prikazal v končnem poglavju.

Celoštevilski linearni program je definiran na naslednji način z podatki:

𝐴 ∈ Rm×n, 𝑐 ∈ Rn, 𝑏 ∈ Rm

in iščemo

𝑥 ∈ Zn,

kjer je dosežen

pri pogojih:

Pri reševanju problema bom pri pogojih privzel 𝐴𝑥 = 𝑏 in 1 ≥ 𝑥 ≥ 0, ker to zahteva moj problem.

# Poimenovanje spremenjivk in zapis CLP

V seminarski nalogi bom s A označil množico, ki vsebuje 2𝑛 točk

in . Povezava predstavlja usmerjeno povezavo iz točke v točko . Definiramo tudi

ykm = premica ki gre skozi točki

## Želimo:

## Pogoji:

Iz vsake točke gre lahko največ 1 daljica

pregstavlja daljico iz točke v točko . In vedno velja:

Imamo še splošne pogoje, ki omejujejo splošne spremenjivke:

## Celotni CLP:

Iščemo:

Pri pogojih:

# Komplikacije

Zečel sem iz uporabo MixedIntegerLinearProgram, a sem po dnevih pisanja videl, da sem se zapletel pri pisavi in sem začel od začetka. Tokrat sem si sam spisal vse funkcije in jih dodal v klas metoda.

Nasledna težava je prišla, ko sem začel pisati metodo, ki razporeja daljice med točkami. Nikakor mi ni uspelo napisati v obliki rekursije, in sem po 8 urah neprestanega kodiranja se razjezil, pobrisal vse in dodal v program najljučni izbor daljic. Ta del sem dogradil ko je bil projekt že končan.

Nasledni problem je bil ugotoviti ali se premica in daljica sekata. Hmalu sem ugotovil, da če si najprej zrišem na papir veliko hitreje končam nalogo. Na papir sem zastavil enačbo in postopek, tek ga prepisal v program python.

Ko ustvariš nov array oz ga pre imenuješ, se le doda isti pointer na array v pomnilniku. To sem se naučil pri PROG1 pozabil in to znanje spet osvežil.

Problemi so nastali, ko sem moral vse manjše metode združit v eno in, problemi so se znašali pri klicih, listi niso kompatibilni z množicami in podobno.

Vse probleme sem rešil, in zdaj sem rabil le še izpisat kar sem ugotovil. Tu je napočil problem. Jaz sem izbiral le naklučne kombinacije dalcij v polju in ne vseh. Ta marginalna napaka je vse moje rezultate razveljavila.

Svoj problem sem zapisal na papir in hmalu ugotovil, da rabim le izpisati vse permutacije v array, to sem si pomagal z googlom, našal sem funkcijo ki yielda vse elemente in jo spremenil, da vrača le te, katere jaz želim (vsaka dvojica mora imeti 1 element manjši od naslednjega, saj rabim deljico preverjat le v eno smer)

Kodo, sem moral popravit tako da je sedaj delovala z drugačnim konstruktorjem in dodal par parametrov za ekperimentacijo z problemom in dodal izpis časa izvajanja posamezne metode.

# Časovna zahtevnost

Časovna zahtevost narašča fakultativno oz. z izbolšavami .

Število razporedb daljic po točkah:

Čas zračuna permutacij daljic po n točkah:

Časovna zahtevnost izračuna presekov daljic z premico:

# Rezultat C

S pomočjo python programa, sem uspel priti do števila n=6. Poravil sem 15 ur komputiranja in za vsako število točk sem porabil 3 ponovitve.

Število presekov v vseh teh točkah je bila 1, vidimo, da to število v splošnem pada tudi v splošnem.

# Zaključek

Ugotovil sem, da je uporaba linearnih programov zelo pomembna in fakultetno naraščanje je strašanko hitrejše od kubičnega. Za N=10 Se razlikujeta za več kot 4 desetiško stopnjo.

Zanimivo odkritje je, da če postavim točke v krog se cel postopek izvede hitreje, čeprav sem porabil več časa, da sem same točke postavil v krog. Predvidevam, da je to zaradi bližine točk, in se decimalke hitreje izračuna.

Število C pri eksperimentiranju je odvisno tudi od šrevilo ponovitev, saj le občasno so točke postavljene tako, da se število maksimalnih presečišč poviša za 1 ali 2.