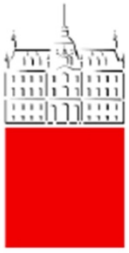
Univerza v Ljubljani
Fakulteta za matematiko in fiziko



Matching with low crossing numbers

Avtor:
Jaka Basej

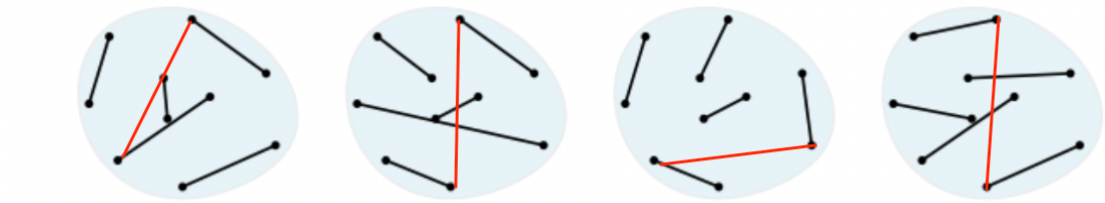
Ljubljana, 2021

Kazalo:

Uvod

V seminarski nalogi sem reševal zahtevni problem maksimalnega števila presekov premice z daljicami med pari točk. Program sem najprej napisal v Pythonu z uporabo naključnih možnih daljic po $2 * n$ točkah. Nato sem nalogo resno izvedel in zapisal celoštevilski linearni program. Dodal sem metodo, ki zračuna število pesekov premic z daljicami preko vseh možnih razporeditvah daljic.

Problem v osnovi zgleda težak, z malce truda pa vidiš, da je vprašanje zelo enostavno, le zapis rešitve je zakopliciran. Pri brout-force metodi sem imel težavo zapisati vse možne razporeditve daljic v array. Pri CLP pa sam zapis problema. Obadva postopka sta zelo potratna s strani procesorske moči. Le, da je CLP veliko bolj časovno efektiven.



Slika 1: Prikaz daljic 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 problem brez možnosti za izboljšanje učikovitosti. Za probleme potrebujemo procesorsko moč, časovno zahtevnost bom prikazal v končnem poglavju.

Celoštevilski linearni program je definiran na naslednji način z podatki:

$$A \in R^{m} * {}^{n}, c \in R^{n}, b \in R^{m}$$

in iščemo

$$x \in Z^{n},$$

kjer je dosežen

$$max\ c^{T}x$$

pri pogojih: $Ax \leq b$

$$x \geq 0$$

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

Poimenovanje spremenjivk in zapis CLP

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

$\alpha_i = (\alpha_i(1), \alpha_i(2)) \in R^2, i = 1,2,3, \ldots, 2n$ in $n \in N$. Povezava uv predstavlja usmerjeno povezavo iz točke $\alpha_i \in P$ v točko $\alpha_j \in P$. Definiramo tudi

ykm = premica ki gre skozi točki $\alpha_k \; in \; \alpha_m$

Y_{km} ={uv seka km}

$$\sum_{uv \in Y_{km}} X_{uv} \leq t$$

Želimo

$$Min\ t$$

Pogoji

Najprej moramo zagotoviti, da Iz vsake točke gre lahko največ 1 daljica in v vsaki točki se lahko največ ena daljica konča. Za lažji zapis programa uv zapišemo kot množico {u,v}

$$\sum_{v=0}^{2n-1} x_{uv} = 1, za \; \forall \; u = 0, 2, \ldots, 2n - 1.$$

x_{uv} predstavlja daljico na točkah v in u. In vedno velja:

$$x_{vu} = x_{uv}$$

, ker velja $\{u, v\} = \{v, u\}$

Nazadnje napišemo še splošne pogoje, ki omejujejo splošne spremenjivke:

$$0 \leq x_{uv} \leq 1 \quad \text{x} \in \text{N}^2$$

Celotni CLP

Iščemo:

$$Min\ t$$

Pri pogojih:

$$0 \leq x_{uv} \leq 1 \quad \text{x} \in \text{N}^2$$

$$\sum_{uv\ \in\ Y_{km}} X_{uv} \leq t$$

$$\sum_{v=0}^{2n-1} x_{uv} = 1, za \forall\ u = 0, 2, \ldots, 2n-1.$$

$$x_{vu} = x_{uv}$$, ker velja {u,v} = {v,u}

Komplikacije

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

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

Nasledni problem je ugotoviti ali se premica in daljica sekata.

Po še več majših napakah v kodi, sem moral združiti vse metode v eno oz. jih poklicati v pravem vrstnem redu.

Kodo, sem dopolnil z bolšo verzijo raporeditve daljic tako, da je sedaj delovala z drugačnim konstruktorjem in dodal sem še par parametrov za ekperimentacijo z problemom in dodal izpis časa izvajanja posamezne metode.

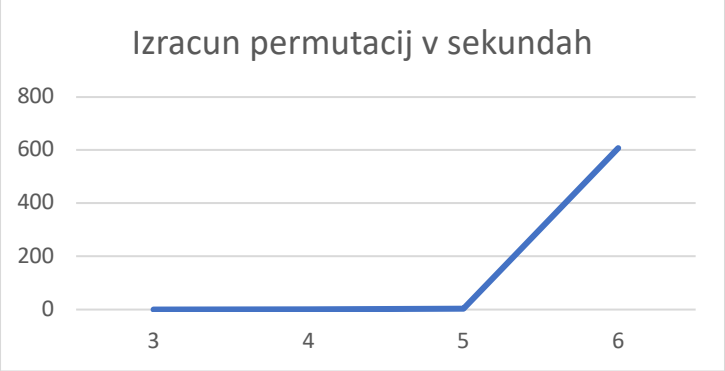
Postopek, je preveč časovno potraten, tako da sem spisal cel program še enkrat v obliki CLP.

Časovna zahtevnost prvotnega programa

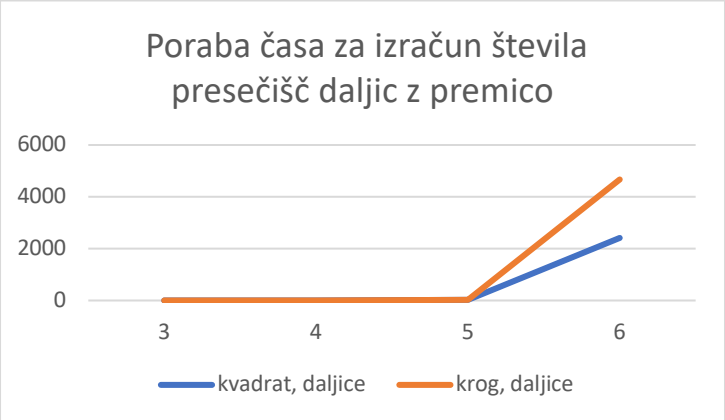
Časovna zahtevost narašča fakultativno!

Na spodnjem grafu sem prikazal naraščanje števila permutacij za $2n$ točk. Noro, Narašča veliko hitreje, kot sem se sprva zavedal.

Čas, ki ga program potrabuje, da izračuna vse permutacije narašča fakultativno. Za 12 točk, potrebuje že več kot 1 uro. CLP pa to reši v sekundi.

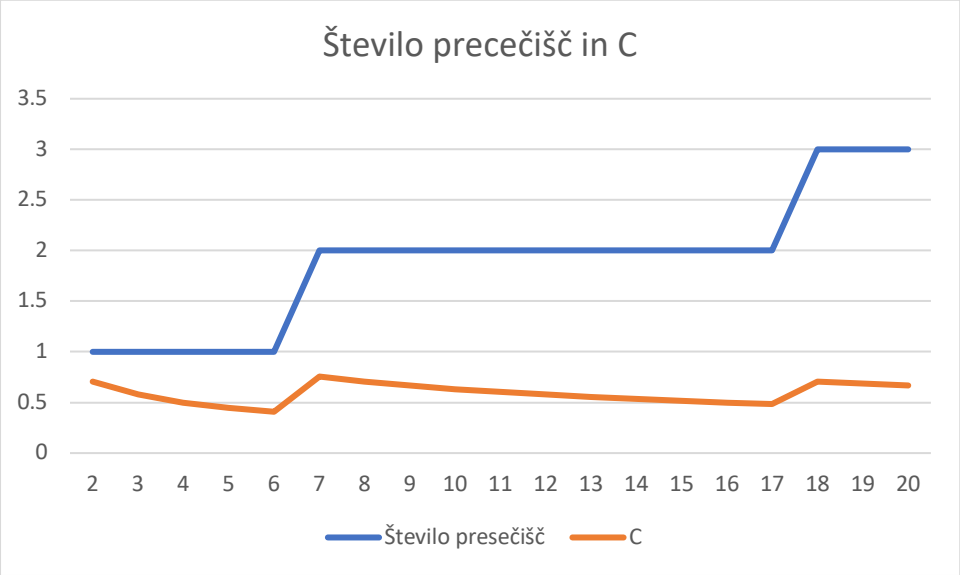


Čas, ki, ga program porabi, za izračuna število preseko daljic z premico, je visok, potrebujem boljši postopek in to je CLP.

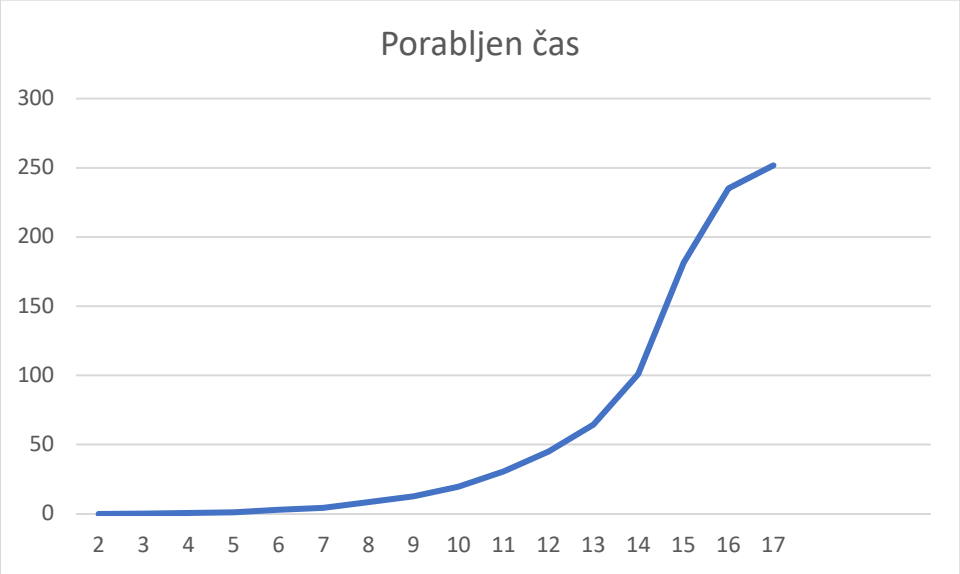


Rezultat C

S pomočjo CLP programa, sem uspel priti do števila $n = 20$. Prikazujem nihanje števila C.



CLP je veliko bolj efektiven, iz grafa lahko razberemo da čas porabe za izračun narašča skoraj kvadratno.



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, četudi je metoda randomizacije točk bolj kompleksna. Predvidevam, da se razdalja točk zmanjša in posledično se decimalke hitreje izračuna.

Število C pri eksperimentiranju je odvisno tudi od števila ponovitev, ker občasno so točke postavljene tako, da število maksimalnih presečišč ni povsod enako.

Izkaže se, da je C približno $0{,}71$.