na vrhu

Optimizacijske metode

3. Lokalna optimizacija

Vladimir Batagelj

Univerza v Ljubljani

FMF, matematika

Kazalo

Globalni in lokalni minimumi

Pogosto lahko v dani množici Ω definiramo relacijo *sosednosti* rešitev $S \subseteq \Omega \times \Omega$, za katero zahtevamo le refleksivnost.

Okolice. Kadar je $\Omega = \mathbb{R}^n$, za dani $\varepsilon > 0$, običajno definiramo relacijo $S(\varepsilon)$ s predpisom:

$$xS(\varepsilon)y \equiv d(x,y) \leq \varepsilon$$

kjer je d izbrana razdalja. Množica $S(\varepsilon, x)$ sosedov točke x je tedaj običajna ε–okolica točke x. Velja:

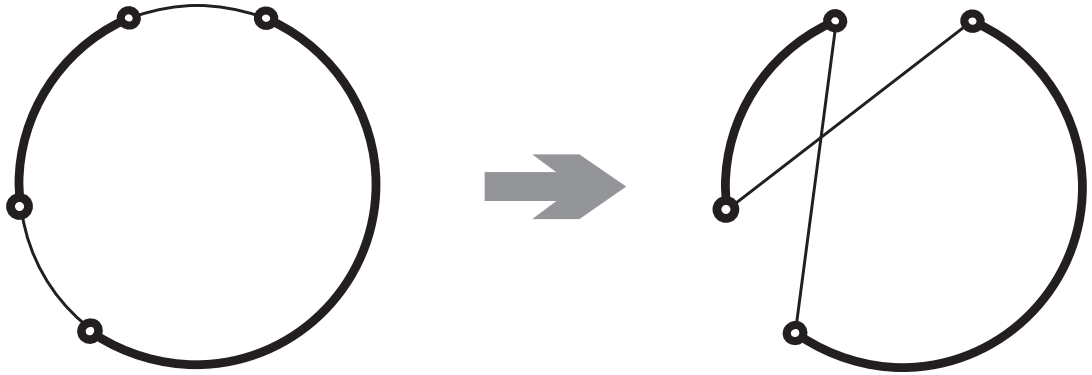
$$\varepsilon < \eta \Rightarrow S(\varepsilon) \subset S(\eta)$$

Lokalne transformacije

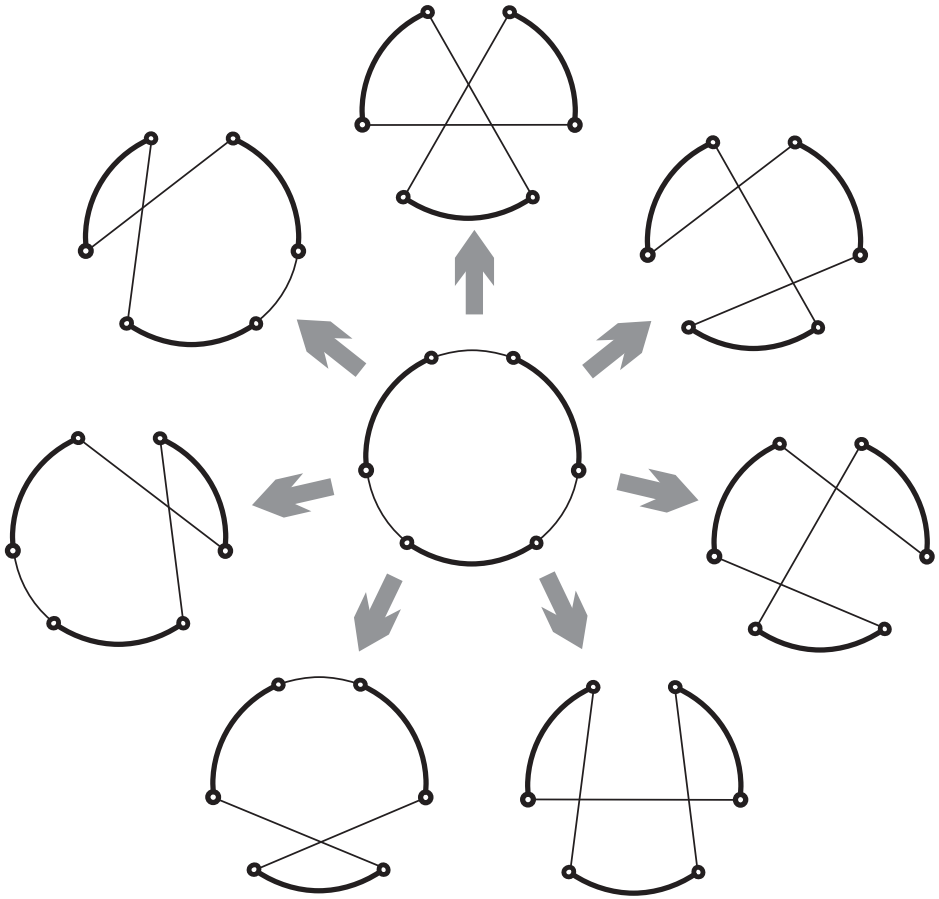
V diskretnih optimizacijskih nalogah običajno definiramo sosednost z *lokalnimi transformacijami*, ki prevedejo eno rešitev v drugo.

Dvojiški vektorji. V množici $\mathbb{B}^n$ dvojiških vektorjev dolžine n navadno definiramo lokalno transformacijo s spremembo vrednosti na i-tem mestu: $x'_i = 1 - x_i$.

Trgovski potnik. Pri problemu trgovskega potnika se najpogosteje uporabljajo sosednosti r-OPT, pri katerih so sosednje rešitve določene tako, da iz tekočega cikla odstranimo r povezav in jih nadomestimo z novimi, ki zopet sestavljajo cikel.



. . . Lokalne transformacije



Na slikah sta prikazani sosednosti 2-OPT in 3-OPT. Lin je s poskusi pokazal, da je sosednost 3-OPT veliko boljša kot sosednost 2-OPT, učinek sosednosti višjih redov pa ne upravičuje povečane porabe časa.

. . . Lokalne transformacije

Permutacije. V primeru, ko je $\Omega = S_n$ (permutacije n elementov), pa lahko postavimo:

$$\pi S \sigma \equiv \pi = \sigma \vee \exists p, q : \sigma = (pq)\pi$$

Vpeta drevesa. Vpeto drevo – izberemo tetivo, vključimo jo v novo drevo, iz pripadajočega cikla pa izločimo neko povezavo.

Lokalni minimumi

Element $x \in \Phi$ je *lokalni minumum* glede na S natanko takrat, ko je $x \in \mathrm{Min}(\Phi \cap S(x), P)$; ali drugače povedano, ko

$$\forall y \in \Phi \cap S(x) : P(x) \leq P(y)$$

Množico vseh lokalnih minimumov naloge (Φ, P, Min) glede na sosednost S označimo $\mathrm{LocMin}(\Phi, P, S)$.

Zato, da bi poudarili razliko, pravimo elementom množice $\mathrm{Min}(\Phi, P)$ tudi *globalni minimumi*. Očitno je vsak globalni minimum tudi lokalni.

. . . Lokalni minimumi

Poleg tega velja še:

IZREK 1 *Za vsako sosednost S je*

$$\mathrm{LocMin}(\Phi, P, \Phi \times \Phi) = \mathrm{Min}(\Phi, P) \subseteq \mathrm{LocMin}(\Phi, P, S)$$

in, če je $\emptyset \subset Q \subset S$, tudi

$$\mathrm{LocMin}(\Phi, P, S) \subseteq \mathrm{LocMin}(\Phi, P, Q)$$

Trditev je posledica lastnosti

$$\emptyset \subset \Psi \subset \Phi \ \wedge \ x \in \Psi \cap \mathrm{Min}(\Phi, P) \ \Rightarrow \ x \in \mathrm{Min}(\Psi, P)$$

Lokalna optimizacija

Relacija sosednosti nam ponuja za reševanje optimizacijskih nalog naslednji postopek *lokalne optimizacije*:

izberi $x \in \Phi$;
while $\exists y \in S(x) \cap \Phi : P(y) < P(x)$ **do** $x := y$;

Če se postopek izteče v končno korakih, konča v lokalnem minimumu.

V postopku lahko relacijo sosednosti tudi spreminjamo – npr. zmanjšujemo ε.

Postopek lokalne optimizacije

Pri razdelavi postopka lokalne optimizacije za dani optimizacijski problem moramo poiskati odgovore na naslednja vprašanja:

a. ***določitev sosednosti*** $S \subseteq \Omega \times \Omega$. Kot vemo, čim bogatejša je sosednost, tem verjetneje so lokalni minimumi tudi globalni. Po drugi strani pa pregledovanje obsežne soseščine zahteva precej časa. Pri izbiri sosednosti zato poskušamo uravnotežiti obe nasprotujoči si želji.

b. ***izbira začetne rešitve***. Lokalnim minimumom se poskušamo izogniti tako, da postopek lokalne optimizacije večkrat ponovimo pri različnih (naključnih) začetnih rešitvah in si zapomnimo najboljšo dobljeno rešitev. Izdelava 'poštenega' generatorja naključnih dopustnih rešitev je lahko včasih sama zase zahtevna naloga.

. . . Postopek lokalne optimizacije

Pri večjem številu ponovitev se pokaže kot zelo dober prikaz zgradbe prostora rešitev (glede na izbrano sosednost) tabela desetih trenutno najboljših dobljenih rešitev. Zaželjeno je tudi, da lahko postopku lokalne optimizacije kot začetno rešitev podtaknemo rešitev, ki si jo izmisli uporabnik, ali rešitev, ki jo dobimo s kakim približnim postopkom.

c. ***dokaz ustavljivosti postopka***. Zaporedje vrednosti rešitev, ki nastopijo pri lokalni optimizaciji, je padajoče. če pokažemo še, da je navzdol omejeno in se vsakič spremeni vsaj za $\delta > 0$, se postopek po končno korakih izteče. Na končni množici Φ je to vselej res.

. . . Postopek lokalne optimizacije

d. ***izbira naslednje rešitve***. Pri končnih soseščinah običajno uporabljamo kar pregled vseh sosedov. Včasih pa lahko za določitev ustrezne rešitveuporabimo lastnosti kriterijske funkcije in omejitev (npr. gradientni postopek). Pogosto obstaja v soseščini več rešitev z manjšo vrednostjo. V teh primerih najpogosteje izberemo ali

- prvo tako rešitev, ki jo najdemo; ali pa
- rešitev, ki najbolj zmanjša vrednost kriterijske funkcije (najhitrejši spust).

Poskusi kažejo, da glede končnih rezultatov ni značilnih razlik med obema pristopoma. Prvi porabi manj časa pri pregledovanju okolice, pa zato naredi več korakov . . .

. . . Postopek lokalne optimizacije

e. ***učinkovito preverjanje pogoja*** $P(y) < P(x)$. Pogosto je kriterijska funkcija P sestavljena iz prispevkov posameznih sestavin rešitve. Zato se izkaže, da se splača pogoj $P(y) < P(x)$ nadomestiti z enakovrednim pogojem $P(x) - P(y) > 0$ ali $P(x)/P(y) > 1$ ali kakim drugim, ker je izraz, ki nastopa v tem pogoju precej enostavnejši kot sama kriterijska funkcija – prispevki sestavin, ki pri lokalni transformaciji ne sodelujejo, se pokrajšajo.

. . . Postopek lokalne optimizacije

f. ***lokalna optimalnost rešitev***. Lokalna optimizacija nam v splošnem ne zagotavlja, da bomo dobili globalni minimum. V naslednjem razdelku bomo zvedeli, da je za konveksne naloge postopek lokalne optimizacije točen – dobljeni lokalni minimum je vselej globalni. V splošnem se moramo zadovoljiti z ugotovitvijo, da če je postopek generiranja naključnih začetnih rešitev vsaj nekoliko pošten (ustvari lahko vsako dopustno rešitev, čeprav ne nujno z enako verjetnostjo), s številom ponovitev postopka lokalne optimizacije narašča tudi verjetnost, da bo dobljeni najboljši lokalni minimum tudi globalni.

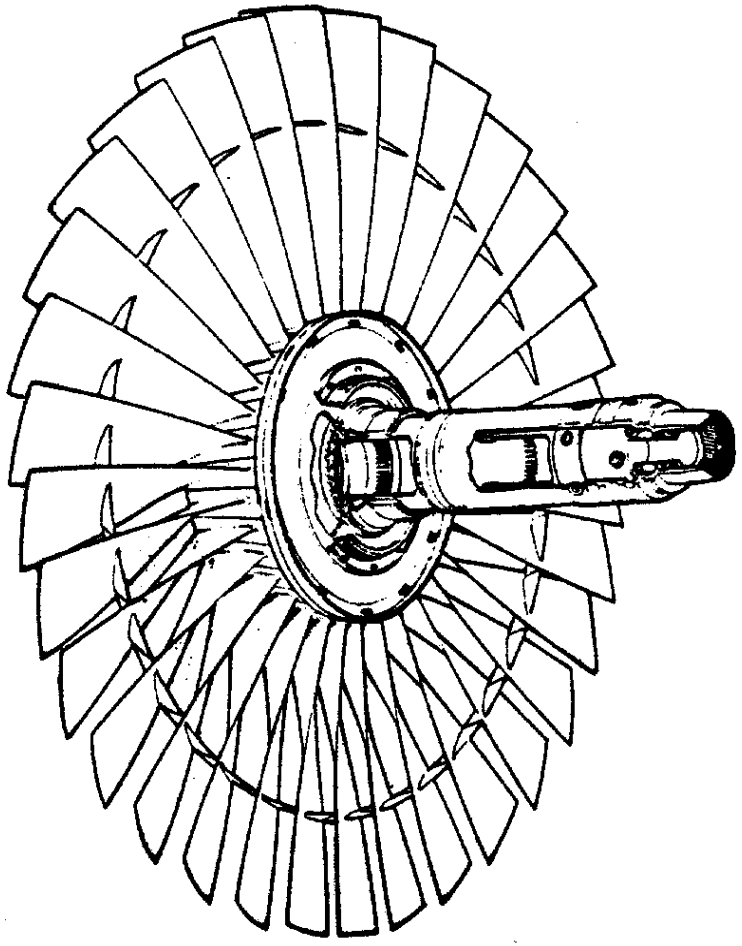
V poglavju o diskretni optimizaciji si bomo ogledali nekatere poskuse izpopolnitve postopka lokalne optimizacije: postopke ohlajanja (simulated annealing) in postopke prepovedanih smeri (tabu search).

. . . Postopek lokalne optimizacije

g. ***povezanost sosednosti*** S. Pri navadni lokalni optimizaciji je povezanost grafa (Ω, S) le lepotnega pomena, pri izpopolnjenih postopkih pa postane zelo zaželjena.

h. ***preverjanje postopka***. Ko postopek sprogramiramo, se postavi vprašanje, kako preveriti njegovo pravilnost in kakovost. Zato je potrebno pripraviti zbirko nalog z znanimi rešitvami. Vir takih nalog so lahko tudi teoretični rezultati o problemu.

Primer: problem uravnotežanja turbin



Veternica v letalskem motorju Rolls Royce RB211-535C

Pri izdelavi turbin lahko mase posameznih lopatic nihajo tudi za do pet odstotkov okrog povprečne vrednosti. Običajno turbino sestavlja 14 do 18 lopatic. Pri nameščanju lopatic na os namestimo težišča lopatic v isti ravnini enakomerno po krožnici. Lopatice želimo namestiti v takem vrstnem redu, da bo težišče turbine čim bliže osi vrtenja.

. . . Problem uravnotežanja turbin

Označimo z n število lopatic, z $m_1, m_2, \ldots, m_n$ mase lopatic in z $M = \sum m_i$ skupno maso lopatic. Razmestitev lopatic okrog osi opišemo s permutacijo σ – na p-to mesto postavimo lopatico $\sigma(p)$. Naj bo še

$$\varphi_i = (i-1)\frac{2\pi}{n}, \quad i = 1, 2, \ldots, n$$

kot, ki pripada i-temu mestu in r polmer krožnice. Tedaj je za razmestitev σ težišče $(\bar{x}_\sigma, \bar{y}_\sigma)$ določeno z izrazoma:

$$\bar{x}_\sigma = \frac{r}{M}\sum_{i=1}^{n} m_{\sigma(i)} \cos\varphi_i \qquad \bar{y}_\sigma = \frac{r}{M}\sum_{i=1}^{n} m_{\sigma(i)} \sin\varphi_i$$

odmik $d(\sigma)$ težišča od osi vrtenja pa z;

$$d(\sigma) = \sqrt{\bar{x}_\sigma^2 + \bar{y}_\sigma^2}$$

Nalogo uravnotežanja turbine lahko torej izrazimo kot optimizacijsko nalogo $(S_n, d(\sigma), \min)$.

. . . Problem uravnotežanja turbin

Krajši razmislek nam pove, da lahko pri razmeščanju eno maso "pribijemo"na izbrano mesto – enakovrednost rešitev glede na zasuk. Poleg tega tudi zrcalni permutaciji določata enakovredni rešitvi. Torej bi morali pri polnem preboru rešitev pregledati $\frac{1}{2}(n-1)!$ permutacij.

Recimo, da program pri polnem preboru (na super-računalniku) pregleda milijon rešitev na sekundo. Tedaj bi pri $n = 14$ tekel 8.6 ur; pri $n = 18$ pa že 49401 ure oziroma 5.6 let.

n	$\frac{1}{2}(n-1)!$
10	181440
11	1814400
12	19958400
13	239500800
14	3113510400
15	43589145600
16	653837184000
17	10461394944000
18	177843714048000
20	60822550204416000
25	3102242008666197196800000
30	442088099686985097727180800000

. . . Problem uravnotežanja turbin

Če dobimo **(a)** razmestitev η iz razmestitve σ tako, da v njej premenjamo masi na p-tem in q-tem mestu (transpozicija), sta koordinati novega težišča $(\bar{x}_\eta, \bar{y}_\eta)$ takole povezani **(e)** s težiščem razmestitve σ:

$$\bar{x}_\eta = \bar{x}_\sigma - \frac{r}{M}(m_{\sigma(p)} - m_{\sigma(q)})(\cos\varphi_p - \cos\varphi_q)$$

$$\bar{y}_\eta = \bar{y}_\sigma - \frac{r}{M}(m_{\sigma(p)} - m_{\sigma(q)})(\sin\varphi_p - \sin\varphi_q)$$

Za izbiro **(b)** začetne razmestitve uporabimo kar naslednji postopek za mešanje, ki je popolnoma pošten

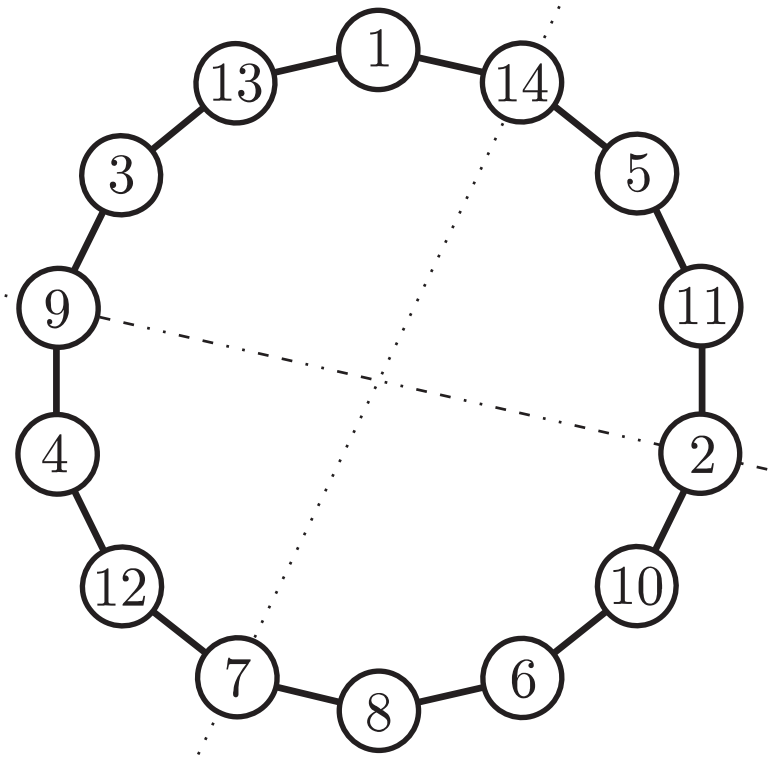
for $i := n$ **downto** 2 **do begin**
$j := 1 + \text{trunc}(i * \text{random}); t := \sigma[i]; \sigma[i] := \sigma[j]; \sigma[j] := t;$
end;

Za prvo permutacijo lahko postavimo kar $\sigma[i] = i$, $i = 1, \ldots, n$. Ker je vsaka permutacija produkt transpozicij, **(g)** je graf sosednosti povezan.

. . . Problem uravnotežanja turbin

$k_{max} := (n-1)(n-2)/2;\ k := k_{max};$
določi naključno začetno razmestitev σ;
izračunaj $\bar{x},\ \bar{y},\ d^2$;
search :
loop
for $p := 2$ **to** $n-1$ **do**
for $q := p+1$ **to** n **do**
$\tilde{x} := \bar{x} - \frac{r}{M}(m_{\sigma(p)} - m_{\sigma(q)})(\cos\varphi_p - \cos\varphi_q);$
$\tilde{y} := \bar{y} - \frac{r}{M}(m_{\sigma(p)} - m_{\sigma(q)})(\sin\varphi_p - \sin\varphi_q);$
$\tilde{d^2} := \tilde{x}^2 + \tilde{y}^2$
if $\tilde{d^2} < d^2$ **then**
$\bar{x} := \tilde{x};\ \bar{y} := \tilde{y};\ d^2 := \tilde{d^2};\ k := k_{max};$
$t := \sigma(p);\ \sigma(p) := \sigma(q);\ \sigma(q) := t;$
else
$k := k-1;$ **if** $k \leq 0$ **exit** *search*
endif
endfor
endfor
endloop

. . . Problem uravnotežanja turbin



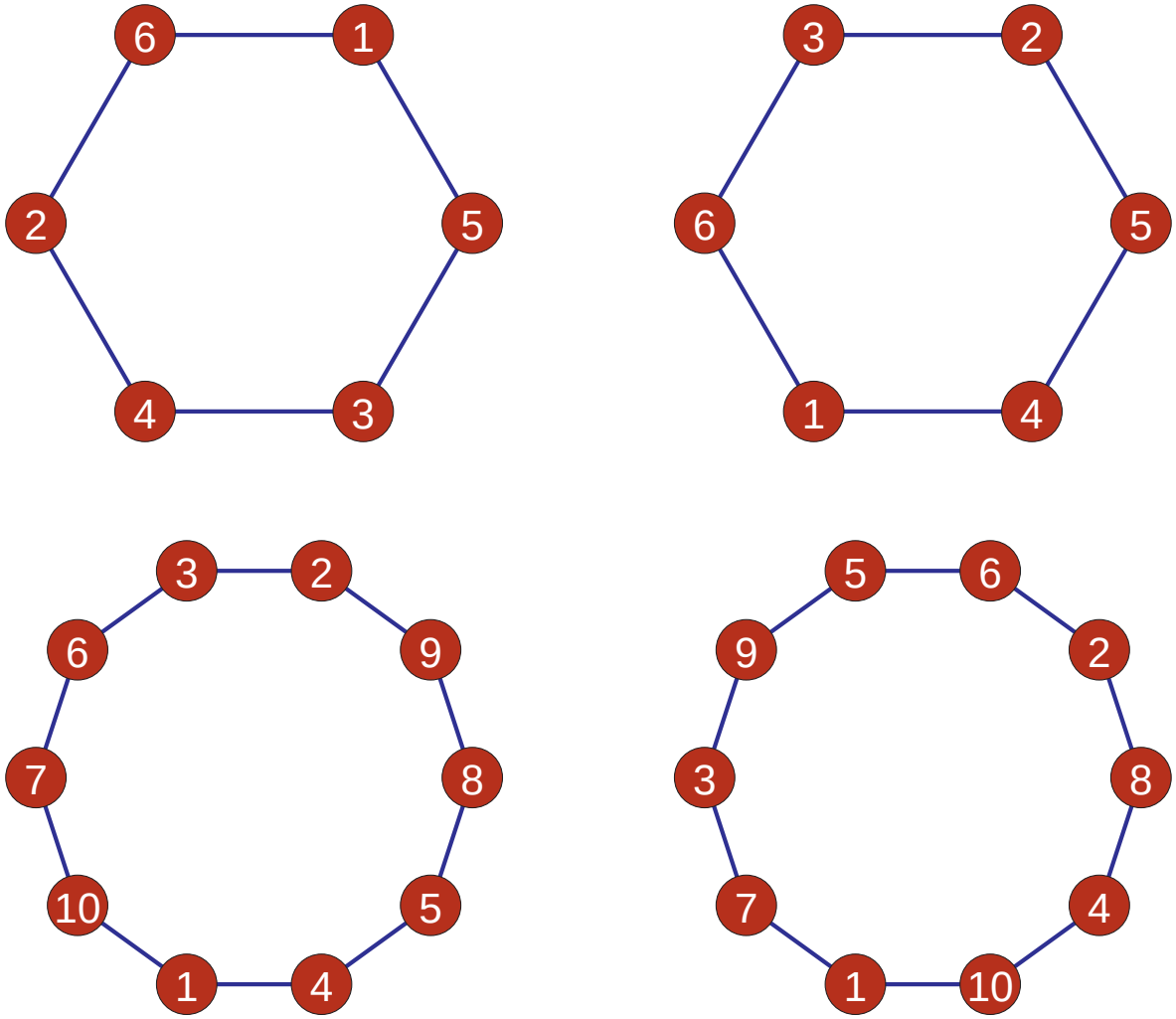
V postopku smo se odločili **(d)** za premik v prvo boljšo rešitev.

Pri preverjanju posopka **(h)** pride prav naslednji izrek o celoštevilskem problemu uravnotežanja turbin, pri katerem je $m_i = i$:

Celoštevilski problem uravnotežanja turbin ima uravnoteženo rešitev $d(\sigma) = 0$ *natanko takrat, ko* n *ni potenca nekega praštevila,* $n \neq p^k, k > 0$.

Na sliki je prikazana uravnotežena rešitev za $n = 14$.

. . . Problem uravnotežanja turbin



. . . Problem uravnotežanja turbin

Sled lokalne optimizacije za σ_a

$$\sigma_a = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 4 & 9 & 7 & 6 & 3 & 10 & 8 & 5 & 2 \end{pmatrix}$$

p	q	x	y	d
		−189.07613	30.50180	191.52060
2	3	−143.62158	−2.52286	143.64374
2	7	−114.20278	18.85115	115.74818
3	4	−80.49184	18.85115	82.66983
4	5	−62.31002	32.06101	70.07458
4	9	−51.07304	−2.52286	51.13531
5	10	7.76456	−45.27088	45.93192
6	7	−13.06995	18.85115	22.93882
6	8	−0.50662	1.55921	1.63945

$$\sigma_a^* = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 10 & 7 & 5 & 2 & 8 & 3 & 9 & 6 & 4 \end{pmatrix}$$

. . . Problem uravnotežanja turbin

Sled lokalne optimizacije za σ_b

$$\sigma_b = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 6 & 5 & 7 & 3 & 1 & 10 & 9 & 8 & 4 & 2 \end{pmatrix}$$

p	q	x	y	d
		−116.85547	−88.01890	146.29603
2	3	−98.67365	−101.22876	141.36390
2	6	−0.00000	−69.16775	69.16775
3	6	47.60062	−34.58387	58.83760
3	8	58.83760	0.00000	58.83760
4	5	40.65578	−13.20986	42.74802
4	10	20.32789	14.76908	25.12665
2	7	−9.09091	−6.60493	11.23698
4	5	−0.00000	0.00000	0.00000

$$\sigma_b^* = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 6 & 9 & 8 & 3 & 2 & 5 & 10 & 7 & 4 & 1 \end{pmatrix}$$