



Lokalna
optimizacija
za BM

V. Batagelj

Optimizacijski
problemi

Bločno
modeliranje

Posplošeni
bločni modeli

Lokalna
optimizacija

Razdelava lokalne optimizacije za bločno modeliranje

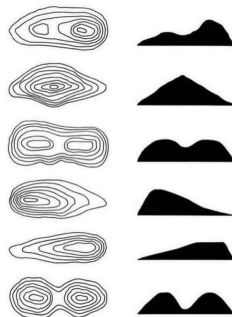
Vladimir Batagelj

IMFM Ljubljana, IAM UP Koper, and NRU HSE Moscow

1304. Sredin seminar
na Zoom, 12. maj 2021

- 1 Optimizacijski problemi
- 2 Bločno modeliranje
- 3 Posplošeni bločni modeli
- 4 Lokalna optimizacija

How to read contour lines on topographic maps



Vladimir Batagelj: vladimir.batagelj@fmf.uni-lj.si
Current version of slides (May 12, 2021 at 17:37): [slides PDF](#)
<https://github.com/bavla/Linked/blob/main/docs/>

Naj bo na množici Φ dana funkcija

$$P : \Phi \rightarrow \overline{\mathbb{R}}$$

kjer je $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty, -\infty\}$. Množici Φ bomo rekli *množica dopustnih rešitev*, funkciji P pa *namenska* ali *kriterijska* funkcija.

Pogosto pri določitvi množice dopustnih rešitev Φ izhajamo iz širše *množice rešitev* Ω , ki jo je enostavneje opisati. Množico dopustnih rešitev Φ tedaj sestavljajo tiste rešitve $X \in \Omega$, ki zadoščajo predikatu *dopustnosti* ali *omejitvam* $\Phi(X)$

$$\Phi = (\Omega, \Phi) = \{X \in \Omega : \Phi(X)\}$$

Običajno je funkcija P definirana na vsej Ω .

Postavimo

$$\text{Min}(\Phi, P) = \{X \in \Phi : \forall Y \in \Phi : P(Y) \geq P(X)\}$$

Za vsak par $X, Y \in \text{Min}(\Phi, P)$ velja $P(X) \geq P(Y)$ in $P(Y) \geq P(X)$; torej je $P(X) = P(Y)$. To pomeni, da imajo vsi elementi množice $\text{Min}(\Phi, P)$, če je ta neprazna, isto vrednost kriterijske funkcije P . Označimo jo z $\min(\Phi, P)$ in jo razširimo na primer, ko je množica $\text{Min}(\Phi, P)$ prazna, s predpisom:

$$\min(\Phi, P) = \begin{cases} \inf_{X \in \Phi} P(X) & \Phi \neq \emptyset \\ \infty & \Phi = \emptyset \end{cases}$$

Tedaj lahko zapišemo tudi

$$\text{Min}(\Phi, P) = \{X \in \Phi : P(X) = \min(\Phi, P)\}.$$

Ker velja $\forall X \in \Phi : P(X) \geq \min(\Phi, P)$, velja še

$$\text{Ext.1 } \forall X \in \Phi \setminus \text{Min}(\Phi, P) : P(X) > \min(\Phi, P)$$



Min in Max

Lokalna
optimizacija
za BM

V. Batagelj

Optimizacijski
problemi

Bločno
modeliranje

Posplošeni
bločni modeli

Lokalna
optimizacija

Na podoben način lahko vpeljemo tudi $\text{Max}(\Phi, P)$ in $\text{max}(\Phi, P)$; ali pa z uporabo zvez:

$$\text{Ext.2 } \text{Max}(\Phi, P) = \text{Min}(\Phi, -P)$$

$$\text{Ext.3 } \text{max}(\Phi, P) = -\text{min}(\Phi, -P)$$

Ti zvezi nam omogočata, da se omejimo samo na Min in min.

Za določitev *optimizacijske naloge* moramo, glede na značilnosti naloge, povedati še kdaj je naloga rešena. To povemo z množico *sprejemljivih* rešitev Σ – torej je naloga natančno določena s trojico (Φ, P, Σ) . Poglejmo si nekaj pogostejših oblik optimizacijskih nalog:

(Φ, P, Min) določi $\Sigma = \text{Min}(\Phi, P)$

$(\Phi, P, \in \text{Min})$ določi $X \in \text{Min}(\Phi, P)$

(Φ, P, min) določi $\text{min}(\Phi, P)$

(Φ, P, lim) določi zaporedje $(X_i : X_i \in \Phi, i \in \mathbb{N})$,
tako da velja $\lim_{i \rightarrow \infty} P(X_i) = \text{min}(\Phi, P)$

$(\Phi, P, \Delta < \varepsilon)$ določi $X \in \Phi$, tako da bo (z dano verjetnostjo)
 $\Delta = P(X) - \text{min}(\Phi, P) < \varepsilon$

$(\Phi, P, \in \Phi)$ določi $X \in \Phi$.

Zaradi zvez Ext.2 in Ext.3 je naloga (Φ, P, Max) enakovredna nalogi $(\Phi, -P, \text{Min})$; in naloga (Φ, P, max) nalogi $(\Phi, -P, \text{min})$.

Množica optimizacijskih nalog sestavlja *optimizacijski problem*. Običajno združimo v optimizacijski problem med seboj podobne naloge, ki imajo enako obliko in se razlikujejo le po podatkih. Nalogi, ki pripada optimizacijskemu problemu, pravimo tudi *primerek* problema.

Rešljivost – O nepraznosti množice $\text{Min}(\Phi, P)$

Lokalna
optimizacija
za BM

V. Batagelj

Optimizacijski
problemi

Bločno
modeliranje

Posplošeni
bločni modeli

Lokalna
optimizacija

O obstoju (globalnih) minimumov – *rešljivosti optimizacijskih nalog* v splošnem ni mogoče veliko povedati. Očitno je množica $\text{Min}(\Phi, P)$ neprazna, če je množica Φ končna in neprazna.

V tem primeru lahko množico $\text{Min}(\Phi, P)$, vsaj teoretično, določimo s *polnim preborom* množice dopustnih rešitev Φ :

```

 $P_{opt} := \infty; \text{Min} = \emptyset$ 
forall  $x \in \Phi$  do
     $P_x := P(x);$ 
    if  $P_x < P_{opt}$  then
         $P_{opt} := P_x; \text{Min} := \{x\}$ 
    elseif  $P_x = P_{opt}$  then
         $\text{Min} := \text{Min} \cup \{x\}$ 
    endif
endfor
    
```


IZREK

Naj bo P zvezna funkcija na kompaktni (zaprta in omejena) množici Φ , tedaj je $\text{Min}(\Phi, P) \neq \emptyset$.

IZREK

Naj bo Φ zaprta množica, $P : \Phi \rightarrow \mathbb{R}$ zvezna funkcija in naj bo za nek $x \in \Phi$ množica $\Phi(x) = \{y \in \Phi : P(y) \leq P(x)\}$ omejena. Tedaj je $\text{Min}(\Phi, P) \neq \emptyset$.

Večkrat lahko množico dopustnih rešitev Φ skrčimo na naslednji način. Recimo, da znamo vsaki rešitvi $x \in \Omega$ pripisati njeno *velikost* $m(x)$, $m : \Omega \rightarrow \mathbb{R}_0^+$. Pravimo, da je kriterijska funkcija *kotlasta*, če zanjo velja

$$\exists x_0 \in \Phi \exists c > 0 \forall x \in \Phi : (m(x) > c \Rightarrow P(x) > P(x_0))$$

Velja

IZREK

Naj bo kriterijska funkcija P naloge (Φ, P, Min) kotlasta in

$$\Psi = \{x \in \Phi : m(x) \leq c\} \neq \emptyset$$

tedaj je $\text{Min}(\Phi, P) = \text{Min}(\Psi, P)$.

Bločni modeli kot problem razvrščanja

Lokalna
optimizacija
za BM

V. Batagelj

Optimizacijski
problemi

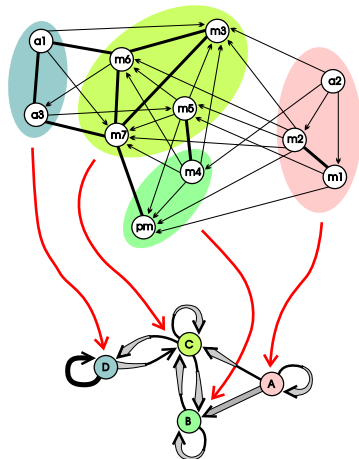
Bločno
modeliranje

Posplošeni
bločni modeli

Lokalna
optimizacija

Cilj *bločnega modeliranja* je skržiti obsežno, morda neskladno omrežje v manjše, razumljivejše omrežje, ki ustrezno povzema zgradbo izvirnega omrežja in ponuja njeno razlago.

Enote/vozlišča omrežja so pri bločnem modeliranju razvrščene v skupine glede na neko *smiselno* enakovrednost.



Skupina, razvrstitev, blok

Lokalna optimizacija za BM

V. Batagelj

Optimizacijski problemi

Bločno modeliranje

Posplošeni bločni modeli

Lokalna optimizacija

Glavni cilj bločnega modeliranja je določiti v danem omrežju $\mathcal{N} = (\mathcal{U}, R, w)$, $R \subseteq \mathcal{U} \times \mathcal{U}$, $w : R \rightarrow \mathbb{R}$ **skupine** (razrede) enot, ki imajo enake ali podobne strukturne značilnosti glede na R – enote iz posamezne skupine so enako ali podobno povezane z enotami drugih skupin. Skupine sestavljajo **razvrstitev** $\mathbf{C} = \{C_1, C_2, \dots, C_k\}$ ki je **razbitje** množice enot \mathcal{U} . Označimo z $V = \{1, 2, \dots, k\}$. Tedaj lahko razbitje \mathbf{C} opišemo s preslikavo $\mu : \mathcal{U} \rightarrow V$

$$\mu(u) = i \Leftrightarrow u \in C_i$$

Kot vemo, vsako razbitje določa neko enakovrednost (in obratno). Označimo s $\sim_{\mathbf{C}}$ enakovrednost določeno z razbitjem \mathbf{C} .

Razvrstitev \mathbf{C} razbije tudi relacijo R na **bloke**

$$R(C_i, C_j) = R \cap C_i \times C_j$$

Vsak tak blok sestavljajo enote iz skupin C_i in C_j in vse povezave, ki vodijo iz skupine C_i v skupino C_j . Če je $i = j$, bloku $R(C_i, C_i)$ rečemo **diagonalni** blok.

Primer: razvrstitev študentov / izposoja gradiv

Lokalna optimizacija za BM

V. Batagelj

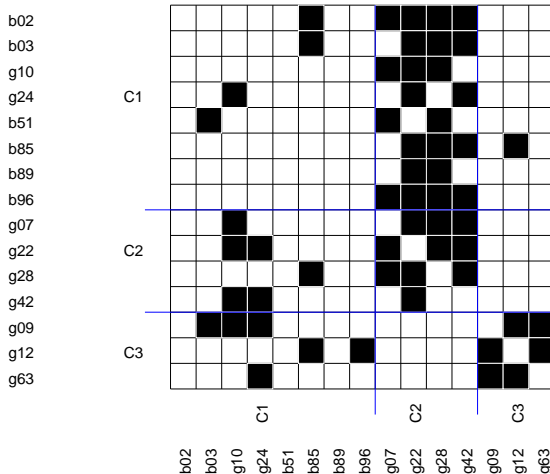
Optimizacijski problemi

Bločno modeliranje

Posplošeni bločni modeli

Lokalna optimizacija

Pajek - shadow [0.00,1.00]



Večina vrst enakovrednosti temelji na ne enem od naslednjih dveh pristopov/pogledov (Faust, 1988):

- enakovredni enoti sta na enak način povezani (imata enak vzorec povezanosti) do **istih** sosedov;
- enakovredni enoti sta na enak ali podoben način povezani do (lahko) **različnih** sosedov.

Primer prve vrste enakovrednosti je strukturna enakovrednost; primer druge vrste pa regularna enakovrednost.

Enoti sta enakovredni, če sta povezani z ostalim omrežjem na *enak* način (Lorrain in White, 1971). Natančneje:

Enoti u in v sta **strukturno enakovredni**, kar zapišemo $u \equiv v$, ntk. ki je permutacija (premena) $\pi = (u\ v)$ avtomorfizem relacije R (Borgatti in Everett, 1992).

Drugače povedano, enoti u in v sta strukturno enakovredni ntk.:

$$\begin{array}{ll} \text{s1.} & uRv \Leftrightarrow vRu \\ \text{s2.} & uRu \Leftrightarrow vRv \\ \text{s3.} & \forall z \in \mathcal{U} \setminus \{u, v\} : (uRz \Leftrightarrow vRz) \\ \text{s4.} & \forall z \in \mathcal{U} \setminus \{u, v\} : (zRu \Leftrightarrow zRv) \end{array}$$

Strukturno enakovredni enoti se razlikujeta le po oznaki – sta izmenljivi.

Za strukturno enakovrednost sta edina mogoča bloka prazni in polni blok (z morebitnim preklopom na diagonali za diagonalne bloke):

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

V dejanskih omrežjih je le malo enot strukturno enakovrednih. Skupni imenovalec vsem poskusom njene posplošitve je, da sta enoti enakovredni, če sta enako povezani z drugimi skupinami.

White in Reitz (1983): Enakovrednost \approx on \mathcal{U} je **regularna enakovrednost** na omrežju $\mathcal{N} = (\mathcal{U}, R)$ ntk. za vse enote $u, v, z \in \mathcal{U}$, iz $u \approx v$ izhaja tudi

$$R1. \quad uRz \Rightarrow \exists t \in \mathcal{U} : (vRt \wedge t \approx z)$$

$$R2. \quad zRu \Rightarrow \exists t \in \mathcal{U} : (tRv \wedge t \approx z)$$

Drug pogled na regularno enakovrednost temelji na barvanjih vozlišč (Everett, Borgatti 1996). Naj bo $b : \mathcal{U} \rightarrow B$ barvanje vozlišč omrežja. Enakovrednost \approx je **regularna** ntk. za vsak par $u, v \in \mathcal{U}$ velja

$$u \approx v \Leftrightarrow (b(u) = b(v)) \wedge (b(R(u)) = b(R(v)))$$

$$R(u) = \{v \in \mathcal{U} : uRv\}.$$

IZREK

Naj bo $\mathbf{C} = \{C_i\}$ razbitje, ki določa regularno enakovrednost \approx na omrežju $\mathcal{N} = (\mathcal{U}, R)$. Tedaj je vsak blok $R(C_i, C_j)$ ali prazen ali pa ima lastnost da obstaja vsaj ena 1 v vsaki njegovi vrstici in vsakem njegovem stolpcu. Obratno, če so za dano razbitje \mathbf{C} vsi bloki teh dveh vrst, je ustrezna enakovrednost regularna.

Bloki regularne enakovrednosti so prazni ali 1-pokriti.

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

1	0	1	0	0
0	0	1	0	1
0	1	0	0	0
1	0	1	1	0

Problem določitve razbitja enot omrežja glede na izbrano enakovrednost je poseben primer *problema razvrščanja*, ki ga lahko zastavimo kot optimizacijski problem (Φ, P, \min) :

Določi razbitje $\mathbf{C}^ \in \Phi$ za katerega velja*

$$P(\mathbf{C}^*) = \min_{\mathbf{C} \in \Phi} P(\mathbf{C})$$

pri čemer je Φ množica *dopustnih razbitij* in je $P : \Phi \rightarrow \mathbb{R}$ *kriterijska funkcija*, ki meri “napako” razbitja.

Ker je množica enot \mathcal{U} končna, je končna tudi množica dopustnih razbitij Φ ; in če je ta neprazna, je neprazna tudi množica optimalnih razbitij $\text{Min}(\Phi, P)$ – naloga je rešljiva.

Težava je, da je večina problemov BM težkih – znani točni algoritmi so eksponentne zahtevnosti – množica dopustnih razbitij je lahko zelo obsežna.

Najpogosteje se za reševanje problema BM uporabljata

- *posredni* pristop – kriterijsko funkcijo izrazimo na osnovi mere različnosti med vozlišči;
- *premi* pristop – kriterijska funkcija meri odstopanje dejanske razvrstitve vozlišč omrežja od ustreznega idealnega stanja za izbrano vrsto enakovrednosti.

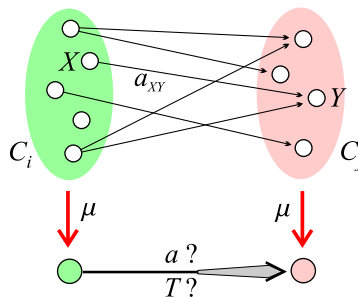
Običajno zahtevamo, da je izbrana kriterijska funkcija $P(\mathbf{C})$ *občutljiva* za izbrano vrsto enakovrednosti:

$$P(\mathbf{C}) = 0 \Leftrightarrow \mathbf{C} \text{ določa enakovrednost izbrane vrste.}$$

Pri reševanju se odpovemo zagotovljeni točnosti rešitev in se zadovoljimo z “dobrimi” rešitvami.

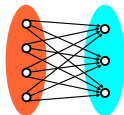
Bločni model je struktura, ki jo dobimo s skrčitvijo posamezne skupine $C_i \in \mathbf{C}$ v pripadajoče modelsko vozlišče $i \in V$ (position). Za natančno določitev BM moramo povedati še:

- kateri bloki ustvarijo usmerjeno povezavo v **modelnem grafu** in kateri ne,
- z določitvijo **vrste povezanosti** T povezave lahko ohranimo informacijo kako sta skupini povezani (posplošeno BM),
- modelni povezavi lahko pripišemo tudi **utež** a , ki povzema vrednosti na povezavah bloka.

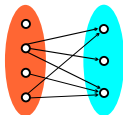


Modelni graf lahko predstavimo tudi z **modelno matriko** (image matrix).

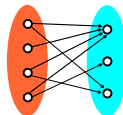
complete



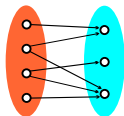
row-dominant



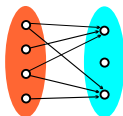
col-dominant



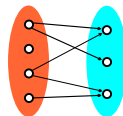
regular



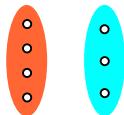
row-regular



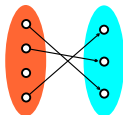
col-regular



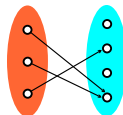
null



row-functional



col-functional



Vrste blokov in matrike

Lokalna optimizacija za BM

V. Batagelj

Optimizacijski problemi

Bločno modeliranje

Posplošeni bločni modeli

Lokalna optimizacija

	Y				
X	1	1	1	1	1
	1	1	1	1	1
	1	1	1	1	1
	1	1	1	1	1

complete

	Y				
X	0	1	0	0	0
	1	1	1	1	1
	0	0	0	0	0
	0	0	0	1	0

row-dominant

	Y				
X	0	0	1	0	0
	0	0	1	1	0
	1	1	1	0	0
	0	0	1	0	1

col-dominant

	Y				
X	0	1	0	0	0
	1	0	1	1	0
	0	0	1	0	1
	1	1	0	0	0

regular

	Y				
X	0	1	0	0	0
	0	1	1	0	0
	1	0	1	0	0
	0	1	0	0	1

row-regular

	Y				
X	0	1	0	1	0
	1	0	1	0	0
	1	1	0	1	1
	0	0	0	0	0

col-regular

	Y				
X	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0

null

	Y				
X	0	0	0	1	0
	0	0	1	0	0
	1	0	0	0	0
	0	0	0	1	0

row-functional

	Y				
X	1	0	0	0	0
	0	1	0	0	0
	0	0	1	0	0
	0	0	0	0	0
	0	0	0	1	0

col-functional

Prepoznavanje vrst blokov

Lokalna optimizacija za BM









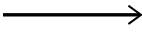

V. Batagelj

Optimizacijski problemi

Bločno modeliranje

Posplošeni bločni modeli

Lokalna optimizacija

null	nul	same 0 *	
complete	com	same 1 *	
regular	reg	1-pokrite vrstice in stolpci	
row-regular	rre	1-pokrite vrstice	
col-regular	cre	1-pokriti stolpci	
row-dominant	rdo	\exists vrstica samih 1 *	
col-dominant	cdo	\exists stolpec samih 1 *	
row-functional	rfn	$\exists!$ ena 1 v vsaki vrstici	
col-functional	cfn	$\exists!$ ena 1 v vsakem stolpcu	
non-null	one	\exists vsaj ena 1	

* izjema na diagonali

Vrste blokov in modelna matrika

Lokalna
optimizacija
za BM

V. Batagelj

Optimizacijski
problemi

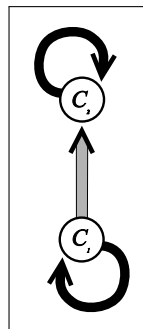
Bločno
modeliranje

Posplošeni
bločni modeli

Lokalna
optimizacija

1	1	1	1	1	1	0	0
1	1	1	1	0	1	0	1
1	1	1	1	0	0	1	0
1	1	1	1	1	0	0	0
0	0	0	0	0	1	1	1
0	0	0	0	1	0	1	1
0	0	0	0	1	1	0	1
0	0	0	0	1	1	1	0

	C_1	C_2
C_1	complete	regular
C_2	null	complete



Bločni model je urejena šesterka $\mathcal{M} = (V, K, \mathcal{T}, A, \tau, \alpha)$ kjer je:

- V je množica *modelskih vozlišč* (mest, skupin);
- $K \subseteq V \times V$ je množica *modelskih povezav*;
- \mathcal{T} je množica predikatov, ki opisujejo *vrste povezanosti* med skupinami omrežja. $\text{nul} \in \mathcal{T}$. Preslikava $\tau : K \rightarrow \mathcal{T} \setminus \{\text{nul}\}$ priredi posamezni modelski povezavi ustrezen predikat.
- A je množica *pravil povprečenja*. Preslikava $\alpha : K \rightarrow A$ določi, po katerem pravilu se izračuna vrednost modelske povezave.

(Surjektivna) preslikava $\mu : \mathcal{U} \rightarrow V$ določa bločni model \mathcal{M} omrežja \mathcal{N} ntk. zadošča zahtevama:

$$\forall (t, w) \in K : \tau(t, w)(C(t), C(w))$$

in

$$\forall (t, w) \in V \times V \setminus K : \text{nul}(C(t), C(w)).$$

Pri tem je $C(t) = \{u \in \mathcal{U} : \mu(u) = t\} = C_t$.

Naj bo \sim enakovrednost (ekvivalenčna relacija) na \mathcal{U} in $[u] = \{v \in \mathcal{U} : u \sim v\}$. Rečemo, da je \sim **usklajena** s \mathcal{T} na omrežju \mathcal{N} ntk.

$$\forall u, v \in \mathcal{U} \exists T \in \mathcal{T} : T([u], [v]).$$

Enostavno je preveriti, da usklajenost s $\mathcal{T} = \{\text{nul}, \text{reg}\}$ ustreza običajni definiciji regularne enakovrednosti (White and Reitz 1983). Podobno, usklajenost s $\mathcal{T} = \{\text{nul}, \text{com}\}$ ustreza definiciji strukturne enakovrednosti (Lorrain and White 1971).

Za usklajeno enakovrednost \sim preslikava $\mu: u \mapsto [u]$ določa bločni model z $V = \mathcal{U} / \sim$.

En izmed načinov izgradnje občutljive kriterijske funkcije za izbrano vrsto enakovrednosti je merjenje odstopanj dejanskih blokov določenih z danim razbitjem od najbližjih idealnih za izbrano vrsto enakovrednosti.

Za dano razbitje $\mathbf{C} = \{C_1, C_2, \dots, C_k\}$ naj bo $\mathcal{B}(C_i, C_j)$ množica idealnih blokov za izbrano vrsto enakovrednosti glede na blok $R(C_i, C_j)$. Tedaj lahko celotno napako razbitja \mathbf{C} izrazimo kot vsoto bločnih napak $Q(C_i, C_j)$

$$P(\mathbf{C}) = \sum_{C_i, C_j \in \mathbf{C}} Q(C_i, C_j) = \sum_{C_i, C_j \in \mathbf{C}} \min_{B \in \mathcal{B}(C_i, C_j)} d(R(C_i, C_j), B)$$

Izraz $d(R(C_i, C_j), B)$ meri odstopanje (napako) bloka $R(C_i, C_j)$ od idealnega bloka B . d izrazimo z uporabo značilnosti za prepoznavanje vrst povezanosti. Tudi mera d mora biti občutljiva za izbrano vrsto enakovrednosti.

Na primer, za strukturno enakovrednost lahko za nediagonalne bloke, izrazimo $d(R(C_u, C_v), B)$ z

$$d(R(C_i, C_j), B) = \gamma(B) \sum_{u \in C_i, v \in C_j} |r_{u,v} - b_{u,v}|.$$

kjer je $r_{u,v}$ vrednost v matriki relacije R in $b_{u,v}$ vrednost v ustreznem idealnem bloku B . Mera šteje število enic v ničelnem bloku ali število ničel v polnem bloku in je občutljiva za strukturno enakovrednost. Z utežjo $\gamma(B)$ lahko odstopanja od izbrane vrste blokov močnejše kaznujemo.

Pri določanju bločne napake določimo tudi vrsto povezanosti – za enoličnost je potrebno vrste povezanosti urediti.

Za prej naštete vrste povezanosti je mogoče ustvariti občutljive mere odstopanj.

Bločno napako – najmanjše odstopanje bloka $R(C_i, C_j)$ od ustreznega idealnega bloka B vrste T lahko razmeroma učinkovito določimo na osnovi značilnosti za prepoznavanje vrst povezanosti in izrazimo z naslednjimi količinami

- s_t – bločna vsota = # enic v bloku,
- n_r = $\text{card}(C_i)$ – # vrstic v bloku,
- n_c = $\text{card}(C_j)$ – # stolpcev v bloku,
- p_r – # neničelnih vrstic v bloku,
- p_c – # neničelnih stolpcev v bloku,
- m_r – največja vrstična vsota,
- m_c – največja stolpčna vsota,
- s_d – diagonalna vsota = # enic na diagonalni,
- d – diagonal napaka = $\min(s_d, n_r - s_d)$.

Število elementov bloka je $n_r n_c$.

... mere odstopanja za izbrano vrsto povezanosti

Lokalna optimizacija za BM

V. Batagelj

Optimizacijski problemi

Bločno modeliranje

Posplošeni bločni modeli

Lokalna optimizacija

Vrsta	$\delta(C_i, C_j; T)$	
null	$\begin{cases} s_t \\ s_t + d - s_d \end{cases}$	nediagonalni diagonalni
complete	$\begin{cases} n_r n_c - s_t \\ n_r n_c - s_t + d + s_d - n_r \end{cases}$	nediagonalni diagonalni
row-dominant	$\begin{cases} (n_c - m_r - 1)n_r \\ (n_c - m_r)n_r \end{cases}$	diagonalni, $s_d = 0$ sicer
col-dominant	$\begin{cases} (n_r - m_c - 1)n_c \\ (n_r - m_c)n_c \end{cases}$	diagonalni, $s_d = 0$ sicer
row-regular	$(n_r - p_r)n_c$	
col-regular	$(n_c - p_c)n_r$	
regular	$(n_c - p_c)n_r + (n_r - p_r)p_c$	
row-functional	$s_t - p_r + (n_r - p_r)n_c$	
col-functional	$s_t - p_c + (n_c - p_c)n_r$	
density γ	$\max(0, \gamma n_r n_c - s_t)$	

Dobljeni optimizacijski problem lahko rešimo z lokalno optimizacijo. Dobimo razbitje μ in vrste povezanosti τ . Pravila povprečenja so običajno tudi določena s funkcijo τ .

Rešitev lahko opišemo s tremi matrikami:

Matrika vrst povezanosti

$$\mathbf{T} = [\tau(i, j)]$$

Matrika bločnih napak

$$\mathbf{Q} = [Q(C_i, C_j)]$$

Matrika bločnih uteži

$$\mathbf{A} = [\alpha(i, j)(R(C_i, C_j))]$$

Predznačeno omrežje (signed network) $N = (\mathcal{U}, R, \sigma)$,
 $\sigma : R \rightarrow \{-1, 1\}$.

Imamo tri vrste blokov: ničelni, pozitivni in negativni. (Doreian, Mrvar)

Bločna napaka

$$Q(C_i, C_j) = \min\left(\gamma_z \sum_{u \in C_i, v \in C_j} |r_{u,v}|, \right. \\ \left. \gamma_p \sum_{u \in C_i, v \in C_j} \max(0, -r_{u,v}), \gamma_n \sum_{u \in C_i, v \in C_j} \max(0, r_{u,v})\right)$$



Utežena omrežja

Lokalna
optimizacija
za BM

V. Batagelj

Optimizacijski
problemi

Bločno
modeliranje

Posplošeni
bločni modeli

Lokalna
optimizacija

Uteženo omrežje $N = (\mathcal{U}, R, w)$, $w : R \rightarrow \mathbb{R}$.

Aleš Žiberna

Z *omejitvami* lahko izrazimo dodatno vedenje o nalogi:

- u pripada skupini C_i : $\mu(u) = i$
- u in v pripadata isti skupini: $\mu(u) = \mu(v)$
- u in v pripadata različnima skupinama: $\mu(u) \neq \mu(v)$
- skupina C_i vsebuje vsaj k enot: $\text{card}(C_i) \geq k$
- skupina C_i vsebuje največ k enot: $\text{card}(C_i) \leq k$

Določimo lahko tudi dovoljene vrste povezanosti za posamezne bloke
– *predoločeno* (prespecified) BM.

Primer: prazni poddiagonalni bloki določajo iskanje hierarhičnega BM.

V **večvrstnem** omrežju je množica enot razbita na nekaj vrst (mode) enot $(\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_s)$, $\bigcup_r \mathcal{U}_r = \mathcal{U}$. Razbitje lahko opišemo s funkcijo $m : \mathcal{U} \rightarrow 1..s$

$$m(u) = r \Leftrightarrow u \in \mathcal{U}_r$$

Bločno modeliranje lahko posplošimo na večvrstna omrežja z zahtevo - omejitvijo

$$\forall C_i \in \mathbf{C} \exists r \in 1..s : C_i \subseteq \mathcal{U}_r$$

oziroma

$$\mu(u) = \mu(v) \Rightarrow m(u) = m(v)$$

Povezana omrežja lahko zložimo v večvrstno omrežje (nekatera podomrežja so lahko prazna). Pri izgradnji kriterijske funkcije je smiselno dodati uteži k napakam posameznega podomrežja. Druga možnost bi bila večkriterijska optimizacija.

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:

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

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

Element $x \in \Phi$ je *lokalni minimum* glede na S natanko takrat, ko je $x \in \text{Min}(\Phi \cap S(x), P)$; 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 $\text{LocMin}(\Phi, P, S)$.

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

Poleg tega velja še:

IZREK

Za vsako sosednost S je

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

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

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

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 .

Pri razdelavi postopka lokalne optimizacije za dani optimizacijski problem moramo poiskati odgovore na več vprašanj.

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 sosesčine zahteva precej časa. Pri izbiri sosednosti zato poskušamo uravnotežiti obe nasprotujoči si želji.

Običajno se pri problemih razvrščanja uporabljata

- **prestavitev**: enoto prestavimo iz ene skupine v drugo; in
- **premena**: izberemo skupini in v vsaki izberemo po eno enoto ter ju prestavimo v drugo skupino.

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.

Na uspešnost postopka lokalne optimizacije močno vpliva "pokrajina", ki jo nad sosednostjo S poraja kriterijska funkcija P .



Pokrajina

Lokalna
optimizacija
za BM

V. Batagelj

Optimizacijski
problemi

Bločno
modeliranje

Posplošeni
bločni modeli

Lokalna
optimizacija



Pri večjem številu ponovitev se pokaže kot zelo dober prikaz zgradbe prostora rešitev (glede na izbrano sosednost) tabela desetih trenutno najboljših dobljenih rešitev (zaporedna številka poskusa, vrednost lokalnega minimuma). Ta seznam nam da občutek o kakovosti dobljene rešitve, zapletenosti problema in ga lahko uporabimo tudi pri odločanju o prekinitvi nadaljnjega iskanja.

Zaželeno 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 ustavljenosti 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.

d. izbira naslednje rešitve. Pri končnih soseščinah običajno uporabljamo kar pregled vseh sosedov. Včasih pa lahko za določitev ustrezne rešitve uporabimo 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 ...

Ker je prvo možnost nekoliko lažje sprogramirati, se običajno odločajo zanj. Pri tem bi lahko določili/spreminjali vrstni red iskanja (slučajno, hevristično).

e. učinkovito preverjanje pogoja $P(y) < P(x)$. Pogosto je kriterijska funkcija P sestavljena iz prispevkov posameznih sestavin rešitve. Zato se izkaže, da se spleta 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.

Pri bločnih modelih z večjim številom skupin to lahko precej pohitri postopek optimizacije.

f. lokalna optimalnost rešitev. Lokalna optimizacija nam v splošnem ne zagotavlja, da bomo dobili globalni minimum. Za konveksne naloge je 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.

Iz preteklih izkušenj vemo, da je pri BM koristno "kolobariti" med začetnimi razbitji z različnimi velikostmi skupin.

Obstajajo poskusi izpopolnitve postopka lokalne optimizacije: postopki ohlajanja (simulated annealing) in postopki prepovedanih smeri (tabu search).

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

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.

i. predstavitev omrežja. Pri programski izvedbi postopka lokalne optimizacije imamo več različnih možnosti. V pascalski izvedbi postopka, ki smo jo uporabljali za raziskave bločnega modeliranja v devetdesetih in je bila kasneje vključena v program Pajek, smo izbrali matrično predstavitev grafov/omrežij. Ta izbira je najbrž najustreznejša za manjše (do nekaj sto vozlišč) in razmeroma goste grafe.

Večji grafi/omrežja pa so običajno redki (imajo majhno povprečno stopnjo - pripadajoča matrika je pretežno zapolnjena z ničlami). Brezplodnemu računanju z ničlami se lahko izognemo tako, da izberemo varčnejšo predstavitev grafa/omrežja, ki vsebuje le obstoječe povezave.

Trenutno uporabljam za programiranje poskusnih programov za BM programski jezik Python si podpornimi knjižnicami Nets, NetsJSON in TQ.

j. Predanaliza. Pri delu z velikimi omrežji postane postopek kljub pohitritvam počasen. Zato je smiselno najprej analizirati zgradbo omrežja:

- če obstaja več komponent povezanosti, lahko problem rešimo za vsako komponento posebej in nato dobljene rešitve združimo v skupno rešitev
- v nekaterih velikih omrežjih je veliko vozlišč-listov. Pri nekaterih problemih lahko liste (in izolirana vozlišča) odstranimo, rešimo problem na skrčenem omrežju in dobljeno rešitev razširimo na začasno odstranjena vozlišča. Omejitev na k -sredice?

j. Omejen pregled okolice. Drug način pospešitve postopka je, da se odpovemo popolnemu preboru vseh možnosti. Najenostavneje je uporabiti slučajno izbiro premikov. Uporabimo lahko tudi ohlajanje (annealing) ali prepovedane smeri (tabu search), kjer dopuščamo (z določeno verjetnostjo) tudi premike v slabša vozlišča.

Naslednja možnost je, da na hitro in učinkovito ocenimo možne premike in dejansko preverimo le nekaj najbolj obetavnih. Seveda utegnejo biti tako dobljene rešitve nekoliko slabše – a jih morda lahko izboljšamo, če smo pripravljeni vložiti v reševanje več časa.



... Postopek lokalne optimizacije

Lokalna
optimizacija
za BM

V. Batagelj

Optimizacijski
problemi

Bločno
modeliranje

Posplošeni
bločni modeli

Lokalna
optimizacija

k. Uporaba prevajanih jezikov.

Preizkušam zmožnosti programskega jezika Julia, ki se vse bolj uveljavlja kot jezik za pripravo hitrih (prevedenih) programov za analizo podatkov in naj bi sčasoma nadomestil trenutno prevladajoča jezika R in Python. V bližnji prihodnosti, ko bom dovolj večji Julie, nameravam razviti podporo za delo z omrežji in tudi bločno modeliranje prestaviti v Julio.



Nekaj spletnih povezav

Lokalna
optimizacija
za BM

V. Batagelj

Optimizacijski
problemi

Bločno
modeliranje

Posplošeni
bločni modeli

Lokalna
optimizacija

[bavla/linked](#)

[Nets](#) – za delo s seznamsko predstavitvijo grafov/omrežij

[NetsJSON](#) – opis omrežij z vključenimi metapodatki in strukturiranimi vrednostmi

[TQ](#) – za delo časovnimi količinami in časovnimi omrežji
[Julia](#)