na vrhu

Optimizacijske metode

1. Optimizacijske naloge

Vladimir Batagelj

Univerza v Ljubljani

FMF, matematika

Kazalo

Optimizacijske naloge

Naj bo na množici Φ dana funkcija

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

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.

Minimalne rešitve

Postavimo

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

Za vsak par $X, Y \in \mathrm{Min}(\Phi, P)$ 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 $\mathrm{Min}(\Phi, P)$, če je ta neprazna, isto vrednost kriterijske funkcije P. Označimo jo z $\min(\Phi, P)$ in jo razširimo na primer, ko je množica $\mathrm{Min}(\Phi, P)$ 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 $\mathrm{Min}(\Phi, P) = \{X \in \Phi : P(X) = \min(\Phi, P)\}$. Ker velja $\forall X \in \Phi : P(X) \geq \min(\Phi, P)$, velja še

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

. . . Minimalne rešitve

V nadaljevanju nam bo večkrat prišla prav zveza

$$\inf_{x \in A \cup B} f(x) = \min(\inf_{x \in A} f(x), \inf_{x \in B} f(x))$$

iz katere izhaja za $\Phi, \Psi \subseteq \Omega$:

$$\min(\Phi \cup \Psi, P) = \min(\min(\Phi, P), \min(\Psi, P))$$

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

Ext.2 $\mathrm{Max}(\Phi, P) = \mathrm{Min}(\Phi, -P)$
Ext.3 $\max(\Phi, P) = -\min(\Phi, -P)$

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

Naštejmo nekaj njunih lastnosti.

Lastnosti minimalnih rešitev

Za $\Phi, \Psi, \Gamma \subseteq \Omega$ velja:

Ext.4 $\Psi \subseteq \Phi \;\Rightarrow\; \min(\Psi, P) \geq \min(\Phi, P)$

Ext.5 $\Psi \subseteq \Phi \;\wedge\; \Psi \cap \mathrm{Min}(\Phi, P) \neq \emptyset \;\Rightarrow$
$\min(\Psi, P) = \min(\Phi, P) \;\wedge\; \mathrm{Min}(\Psi, P) = \Psi \cap \mathrm{Min}(\Phi, P)$

Ext.6 $\Psi, \Gamma \subseteq \Phi \;\wedge\; \mathrm{Min}(\Psi, P) \cap \mathrm{Min}(\Gamma, P) \neq \emptyset \;\Rightarrow$
$\min(\Psi, P) = \min(\Gamma, P)$

Ext.7 $\Psi \subseteq \Phi \;\wedge\; \forall Y \in \Phi \setminus \Psi \, \exists X \in \Psi : P(X) < P(Y)$
$\Rightarrow \mathrm{Min}(\Psi, P) = \mathrm{Min}(\Phi, P)$

Ext.8 $\mathrm{Min}(\Psi, P) \neq \emptyset \;\wedge\; \mathrm{Min}(\Gamma, P) \neq \emptyset \;\Rightarrow$
$\mathrm{Min}(\Psi \cup \Gamma, P) = \mathrm{Min}(\mathrm{Min}(\Psi, P) \cup \mathrm{Min}(\Gamma, P), P) \neq \emptyset$

Ext.9 $\mathrm{Min}(\Phi, P) \neq \emptyset \;\Rightarrow\; \min(\mathrm{Min}(\Phi, P), P) = \min(\Phi, P)$

Ext.10 $\mathrm{Min}(\mathrm{Min}(\Phi, P), P) = \mathrm{Min}(\Phi, P)$

. . . Lastnosti minimalnih rešitev

Pogosto zadostuje že nekoliko šibkejša oblika lastnosti Ext.7:

$$\text{Ext.7'}\quad \Psi \subseteq \Phi \ \wedge\ \exists X \in \Psi\, \forall Y \in \Phi \setminus \Psi : P(X) < P(Y) \\ \Rightarrow \mathrm{Min}(\Psi, P) = \mathrm{Min}(\Phi, P)$$

Lastnost Ext.7 / Ext.7' nam omogoča zoženje iskanja minimalnih rešitev na podmnožico množice dopustnih rešitev – podmnožice, ki vsebujejo same neobetavne rešitve lahko odvržemo.

Lastnost Ext.8 pa omogoča uporabo načela *deli in vladaj* pri iskanju minimalnih rešitev. V ta namen jo zapišemo v obliki:

$$\text{Ext.8'}\quad \Phi = \bigcup_{i=1}^{k} \Phi_i \ \wedge\ \mathrm{Min}(\Phi_i, P) \neq \emptyset,\ i = 1, \ldots, k \ \Rightarrow \\ \mathrm{Min}(\Phi, P) = \mathrm{Min}(\bigcup_{i=1}^{k} \mathrm{Min}(\Phi_i, P), P) \neq \emptyset$$

Določitev $\mathrm{Min}(\bigcup_{i=1}^{k} \mathrm{Min}(\Phi_i, P), P)$ je preprosto: množico $\mathrm{Min}(\Phi, P)$ sestavljajo tiste izmed množic $\mathrm{Min}(\Phi_i, P)$, za katere je $\min(\Phi_i, P) = \min_{j \in 1..k} \min(\Phi_j, P)$.

Skupaj lastnosti Ext.7 in Ext.8' sestavljata osnovo postopkov *razveji in omeji* (branch & bound).

Dokazi

Ext.4 $\Psi \subseteq \Phi \Rightarrow \min(\Psi, P) \geq \min(\Phi, P)$

Velja (tudi v primeru, ko je $\Psi = \emptyset$):

$$\min(\Psi, P) = \inf_{X \in \Psi} P(X) \geq \inf_{X \in \Phi} P(X) = \min(\Phi, P)$$

Neenakost velja, ker je $\Psi \subseteq \Phi$.

Ext.5 $\Psi \subseteq \Phi \wedge \Psi \cap \mathrm{Min}(\Phi, P) \neq \emptyset \Rightarrow$
$\min(\Psi, P) = \min(\Phi, P) \wedge \mathrm{Min}(\Psi, P) = \Psi \cap \mathrm{Min}(\Phi, P)$

Iz predpostavk izhaja, da obstaja taka rešitev X^*, da je $X^* \in \mathrm{Min}(\Phi, P)$ in $X^* \in \Psi$. Zato je $P(X^*) = \min(\Phi, P)$ in $\mathrm{Min}(\Phi, P) = \{X \in \Phi : P(X) = P(X^*)\}$. Po Ext.4 je $\min(\Psi, P) \geq \min(\Phi, P) = P(X^*)$. Ker pa je $X^* \in \Psi$, je tudi $P(X^*) \geq \min(\Psi, P)$. Torej mora veljati enačaj

$$\min(\Psi, P) = \min(\Phi, P) = P(X^*)$$

in dalje

$$\mathrm{Min}(\Psi, P) = \{X \in \Psi : P(X) = P(X^*)\} =$$

$$= \{X \in \Phi : X \in \Psi \wedge P(X) = P(X^*)\} = \Psi \cap \mathrm{Min}(\Phi, P)$$

. . . Dokazi

Ext.6 $\Psi, \Gamma \subseteq \Phi \ \wedge \ \mathrm{Min}(\Psi, P) \cap \mathrm{Min}(\Gamma, P) \neq \emptyset \ \Rightarrow \min(\Psi, P) = \min(\Gamma, P)$

Po predpostavki obstaja taka rešitev $X^* \in \Phi$, ki je skupna obema množicama minimalnih rešitev. Ker je $X^* \in \mathrm{Min}(\Psi, P)$, je $P(X^*) = \min(\Psi, P)$; prav tako, ker je $X^* \in \mathrm{Min}(\Gamma, P)$, je $P(X^*) = \min(\Gamma, P)$. Torej je tudi $P(X^*) = \min(\Psi, P) = \min(\Gamma, P)$.

Ext.7 $\Psi \subseteq \Phi \ \wedge \ \forall Y \in \Phi \setminus \Psi \, \exists X \in \Psi : P(X) < P(Y) \Rightarrow \mathrm{Min}(\Psi, P) = \mathrm{Min}(\Phi, P)$

Pokazati moramo vzajemno vsebovanost obeh množic minimalnih rešitev. Torej:

a) $X^* \in \mathrm{Min}(\Phi, P) \Rightarrow X^* \in \mathrm{Min}(\Psi, P)$.
Iz predpostavke izhaja $\forall Y \in \Phi : P(Y) \geq P(X^*)$ in, ker je $\Psi \subseteq \Phi$, zato tudi $\forall Y \in \Psi : P(Y) \geq P(X^*)$, kar pomeni $X^* \in \mathrm{Min}(\Psi, P)$.

b) $X^* \in \mathrm{Min}(\Psi, P) \Rightarrow X^* \in \mathrm{Min}(\Phi, P)$
Iz predpostavke izhaja $\forall Z \in \Psi : P(Z) \geq P(X^*)$. Za rešitve, ki niso v Ψ pa po predpostavki trditve $\forall Y \in \Phi \setminus \Psi \exists Z \in \Psi : P(Y) > P(Z) \geq P(X^*)$. Združimo obe ugotovitvi in dobimo $\forall Y \in \Phi : P(Y) \geq P(X^*)$, kar pomeni $X^* \in \mathrm{Min}(\Phi, P)$.

. . . Dokazi

Ext.8 $\mathrm{Min}(\Psi, P) \neq \emptyset \wedge \mathrm{Min}(\Gamma, P) \neq \emptyset \Rightarrow$
$\mathrm{Min}(\Psi \cup \Gamma, P) = \mathrm{Min}(\mathrm{Min}(\Psi, P) \cup \mathrm{Min}(\Gamma, P), P) \neq \emptyset$

Naj bo $X^* \in \mathrm{Min}(\Psi, P)$. Tedaj je po Ext.1 $\forall Y \in \Psi \setminus (\Gamma \cup \mathrm{Min}(\Psi, P)) : P(Y) > P(X^*)$. Zato je po Ext.7

$$\mathrm{Min}(\Psi \cup \Gamma, P) = \mathrm{Min}(\Gamma \cup \mathrm{Min}(\Psi, P), P)$$

Ponovimo razmislek še za množico Γ pa dobimo

$$\mathrm{Min}(\Psi \cup \Gamma, P) = \mathrm{Min}(\mathrm{Min}(\Psi, P) \cup \mathrm{Min}(\Gamma, P), P)$$

Pokažimo še nepraznost množice $\mathrm{Min}(\Psi \cup \Gamma, P)$. Po predpodstavkah trditve obstajata rešitvi $X^* \in \Psi$ in $Y^* \in \Gamma$ tako da $\forall X \in \Psi : P(X) \geq P(X^*)$ in $\forall Y \in \Gamma : P(Y) \geq P(Y^*)$. Brez škode za splošnost lahko privzamemo, da je $P(Y^*) \geq P(X^*)$. Tedaj je res tudi $\forall Y \in \Gamma : P(Y) \geq P(X^*)$ oziroma $X^* \in \mathrm{Min}(\Psi \cup \Gamma, P)$.

. . . Dokazi

Ext.9 $\mathrm{Min}(\Phi, P) \neq \emptyset \;\Rightarrow\; \min(\mathrm{Min}(\Phi, P), P) = \min(\Phi, P)$

Po predpostavki obstaja rešitev $X^* \in \mathrm{Min}(\Phi, P)$. Tedaj je

$$\min(\mathrm{Min}(\Phi, P), P) = P(X^*) = \min(\Phi, P).$$

Ext.10 $\mathrm{Min}(\mathrm{Min}(\Phi, P), P) = \mathrm{Min}(\Phi, P)$

Če je $\mathrm{Min}(\Phi, P) = \emptyset$, trditev velja; sicer pa tudi

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

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

Druga enakost izhaja po Ext.9, tretja pa iz definicije množice $\mathrm{Min}(\Phi, P)$.

Sprejemljive rešitve

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 $\Sigma = \mathrm{Min}(\Phi, P)$
$(\Phi, P, \in \mathrm{Min})$	določi $X \in \mathrm{Min}(\Phi, P)$
(Φ, P, min)	določi $\mathrm{min}(\Phi, P)$
$(\Phi, P, \lim)$	določi zaporedje $(X_i : X_i \in \Phi, i \in \mathbb{N})$, tako da velja $\lim_{i \to \infty} P(X_i) = \mathrm{min}(\Phi, P)$
$(\Phi, P, \Delta < \varepsilon)$	določi $X \in \Phi$, tako da bo (z dano verjetnostjo) $\Delta = P(X) - \mathrm{min}(\Phi, P) < \varepsilon$
$(\Phi, P, \in \Phi)$	določi $X \in \Phi$.

. . . Sprejemljive rešitve

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

V nadaljnjem se bomo v glavnem usmerili na stroge naloge minimizacije (Φ, P, Min) in popuščali le, kadar bo to potrebno.

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.

Zgled: Kovanci

Poskusimo zapisati kot optimizacijsko nalogo naslednje vprašanje: Koliko največ enakih kovancev lahko postavimo na indeks tako, da se ne prekrivajo in v celoti leže na indeksu?

Recimo, da je polmer posameznega kovanca enak r in sta dolžini stranic indeksa a in b.

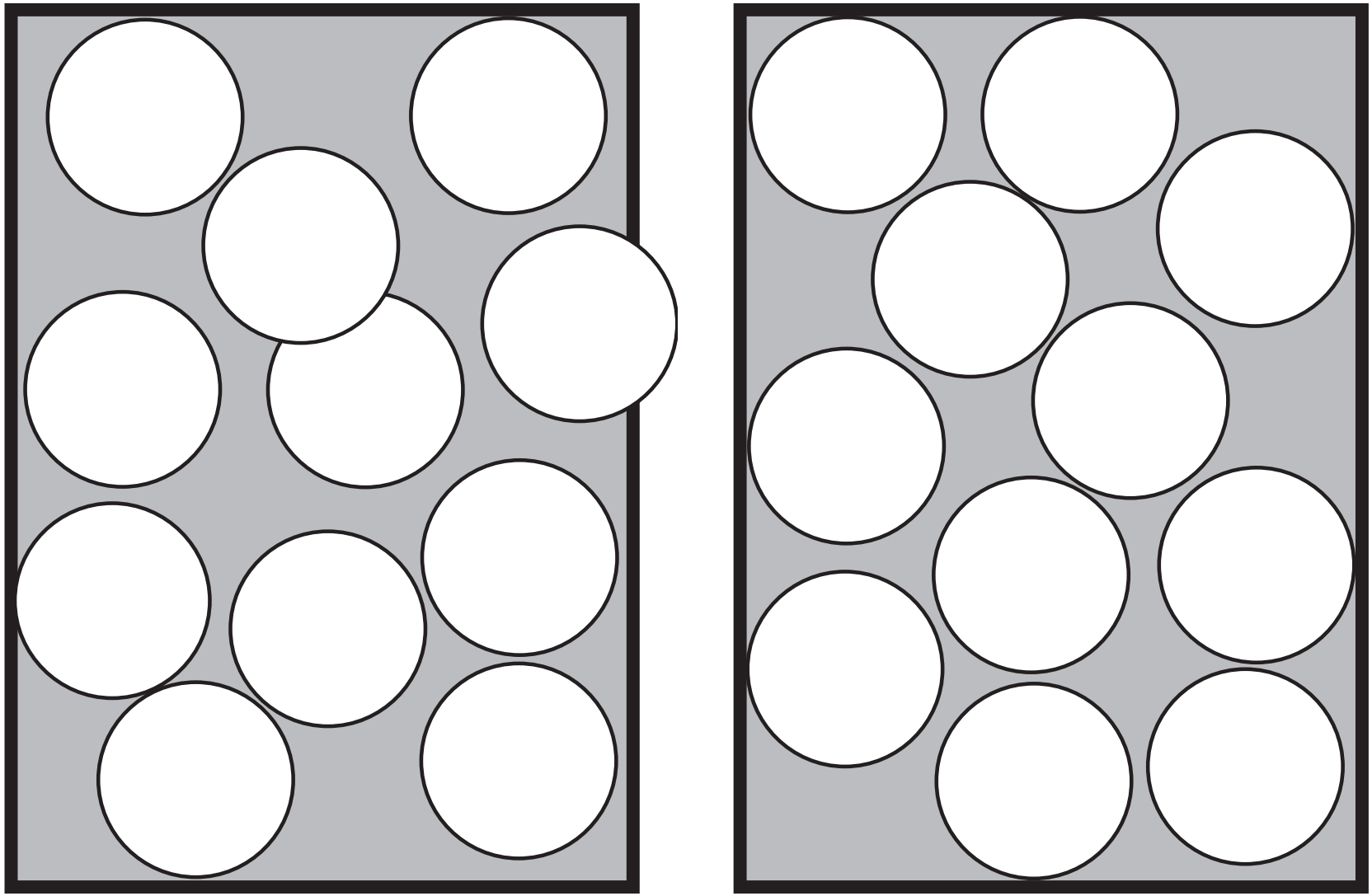
Kaj so rešitve? Vzemimo za rešitve vse možne razmestitve kovancev v ravnini. Kako opišemo posamezno razmestitev? To je množica postavitev posameznih kovancev. Kako povemo, kam smo postavili kovanec? Tako, da povemo kam smo postavili njegovo središče – njegovi koordinati (x, y).

Izhodišče koordinatnega sistema postavimo v sredino indeksa, tako da je stranica a vzporedna osi x. Torej imajo rešitve obliko

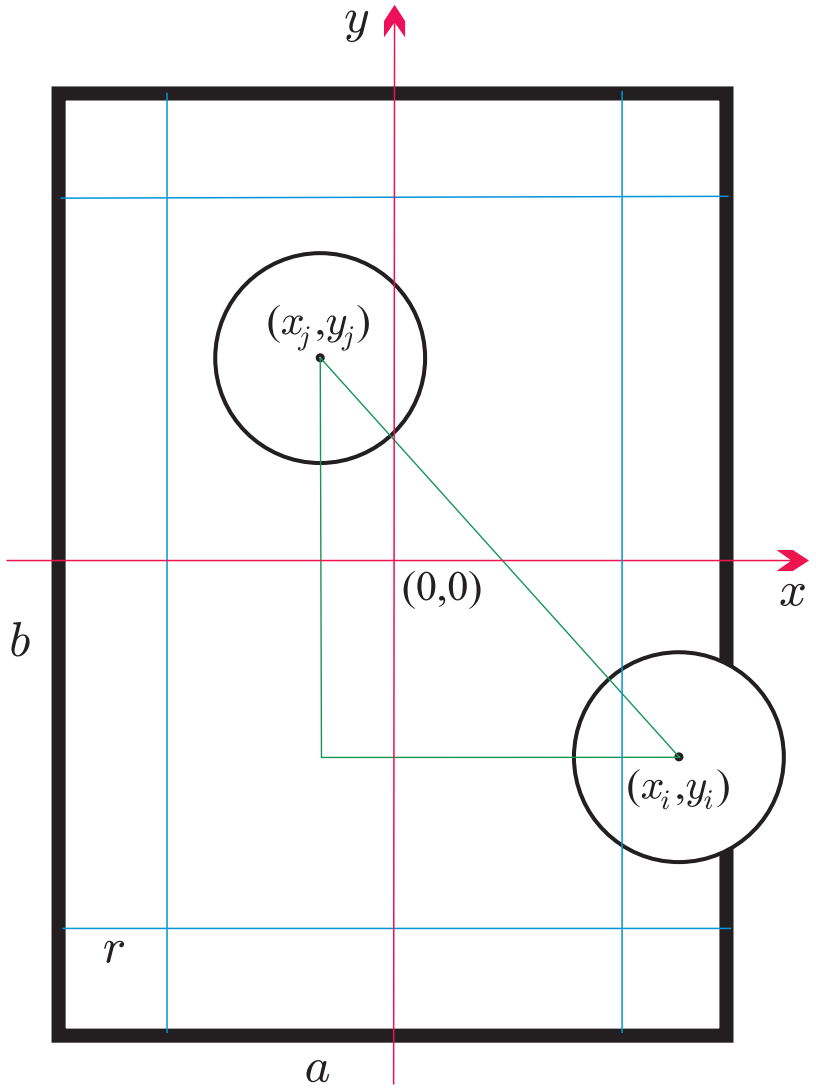
$$X = \{(x_1, y_1),\ (x_2, y_2), \ldots, (x_k, y_k)\}$$

kjer so $x_i, y_i \in \mathbb{R},\ i \in I = 1..k$.

. . . Kovanci



. . . Kovanci



. . . Kovanci

Seveda vse take rešitve ne določajo pravilnih razmestitev na indeksu – niso dopustne. Da bi bila rešitev $X = \{(x_i, y_i)\}_{i \in I}$ dopustna, mora zadoščati naslednjim omejitvam:

- vsak kovanec je v celoti na indeksu:

$$\forall i \in I : ((|x_i| \leq \frac{a}{2} - r) \wedge (|y_i| \leq \frac{b}{2} - r))$$

- različna kovanca se ne prekrivata:

$$\forall i, j \in I : (i \neq j \Rightarrow (x_i - x_j)^2 + (y_i - y_j)^2 \geq (2r)^2)$$

Konjunkcija obeh pogojev sestavlja predikat dopustnosti $\Phi(X)$.

Kaj pa je kriterijska funkcija? Zanima nas največje število kovancev – torej $P(X) = \operatorname{card} X$.

. . . Kovanci

Povzemimo: vprašanje o postavitvi čimvečjega števila kovancev na indeks lahko zapišemo kot optimizacijsko nalogo (Φ, P, Max), kjer je

$$\begin{aligned}\Phi \quad = \quad & \{\, \{(x_i, y_i)\}_{i \in I} : \forall i \in I : ((|x_i| \leq \frac{a}{2} - r) \wedge (|y_i| \leq \frac{b}{2} - r)) \ \wedge \\ & \forall i, j \in I : (i \neq j \Rightarrow (x_i - x_j)^2 + (y_i - y_j)^2 \geq (2r)^2)\}\end{aligned}$$

in je $P(X) = \mathrm{card}\, X$.

Na enak način bi zapisali tudi naloge o postavljanju: kovancev na knjigo, krožnikov na mizo, steklenic v zaboj – spreminjajo se le količine a, b in r. Vse naloge sestavljajo primerke istega problema. □

Seveda je natančen zapis optimizacijske naloge šele prvi, a zelo pomemben, korak pri njenem reševanju. Če ne drugega, znamo za posamezno rešitev presoditi ali je dopustna in za vsak par dopustnih rešitev odločiti, katera je boljša. Formalni zapis naloge je tudi izhodišče za načrtovanje postopka/programa za reševanje naloge.

Primeri problemov

Reševanje sistema enačb

Nalogi: reši sistem enačb

$$P_k(X) = 0, \qquad k \in I, \qquad X \in \mathbb{R}^n$$

lahko priredimo optimizacijsko nalogo $(\mathbb{R}^n, P, \mathrm{Min})$, kjer je

$$P(X) = \sum_{k \in I} P_k^2(X)$$

Sistem enačb je rešljiv natanko takrat, ko je $\min(\mathbb{R}^n, P) = 0$ in je množica $\mathrm{Min}(\mathbb{R}^n, P)$ neprazna. Množica rešitev sistema je tedaj kar $\mathrm{Min}(\mathbb{R}^n, P)$. Če je $\min(\mathbb{R}^n, P) > 0$, je sistem protisloven; elementi množice $\mathrm{Min}(\mathbb{R}^n, P)$; če je neprazna, pa nekako najbolje zadoščajo sicer protislovnim pogojem (enačbam).

. . . Reševanje sistema enačb

Fermat je (1637) na rob Diofantove knjige pripisal, da je našel preprost dokaz tega, da za $n > 2$ ne obstaja nobena rešitev enačbe

$$x^n + y^n = z^n$$

v naravnih številih, a da je žal na robu premalo prostora. Dokaza ni zabeležil tudi nikjer drugje. Od tedaj so matematiki več kot 350 let poskušali dokazati ali ovreči to trditev. To je končno leta 1995 uspelo Andrewu Wilesu – Fermatova trditev je izrek.

Po opisanem postopku lahko Fermatov problem zastavimo kot optimizacijsko nalogo $(\Phi, P, \min)$, kjer je

$$\Phi = \{(x, y, z, n) \in (\mathbb{N}^+)^4, n \geq 3\}$$

$$P(x, y, z, n) = (x^n + y^n - z^n)^2$$

Fermatov problem je enakovreden vprašanju: ali je $\min(\Phi, P) \neq 0$?

Linearno in nelinearno programiranje

Nalogo *linearnega programiranja* imenujemo nalogo (Φ, P, Min) , kjer je

$$\Phi = \{X \in \mathbb{R}^n : AX = b, X \geq 0\} \quad \text{in} \quad P(X) = c^T X$$

ter je A matrika reda $m \times n$ in sta $b \in \mathbb{R}^m, c \in \mathbb{R}^n$.

Kot naloge linearnega programiranja lahko zastavimo vrsto praktičnih optimizacijskih nalog. Zato in zaradi obstoja učinkovitih postopkov za njih reševanje, lahko štejemo linearno programiranje med najbolj rabljene optimizacijske metode.

Nalogo *nelinearnega programiranja* pa imenujmo nalogo (Φ, P, Min) , kjer je

$$\Phi = \{X \in \mathbb{R}^n : P_k(X) \leq 0, k \in I\}$$

in so $P : \mathbb{R}^n \to \mathbb{R}$ in $P_k : \mathbb{R}^n \to \mathbb{R}$, $k \in I$ dane funkcije. Običajno naložimo na funkcije P_k in P še dodatne zahteve, kot sta zveznost in konveksnost.

Izoperimetrična naloga

Izoperimetrična naloga zahteva, da določimo sklenjeno ravninsko krivuljo $\mathcal{K} : r(t) = (x(t), y(t)),\ a \leq t \leq b$ dolžine d, ki omejuje lik največje ploščine. Nalogo lahko zastavimo kot optimizacijsko nalogo (Φ, P, Max), kjer je

$$\Phi = \{\mathcal{K} \in \mathcal{C}^2 : \oint_{\mathcal{K}} \sqrt{\dot{r}\dot{r}}dt = d\}$$

in

$$P(\mathcal{K}) = \frac{1}{2} \oint_{\mathcal{K}} (x\dot{y} - y\dot{x})\, dt$$

Že stari Grki so vedeli, da so rešitve te naloge krožnice z obsegom d. Danes se s podobnimi nalogami ukvarja posebna veja matematične analize – *variacijski račun*.

Naloge diskretne optimizacije

Optimizacijska naloga je naloga *diskretne optimizacije*, če je moč množice Φ števna (končna ali števno neskončna). Med nalogami diskretne optimizacije so najpogostejše:

- celoštevilske: $\Phi \subseteq \mathbb{Z}^n$
- kombinatorične: Φ je podmnožica ene izmed množic kombinatoričnih konfiguracij (permutacije, kombinacije, razbitja, ...)
- naloge na grafih: Φ je podmnožica množice podgrafov danega grafa G.

Med naloge diskretne optimizacije sodijo tudi naslednje:

Minimalno vpeto drevo

Naj bo dan povezan graf $G = (V, E)$ in preslikava $v : E \to \mathbb{R}^+$, ki vsaki povezavi $e \in E$ pripiše njeno vrednost $v(e)$. Naloga o minimalnem vpetem drevesu imenujemo optimizacijsko nalogo (Φ, P, Min), kjer je

$$\Phi = \{H = (V, E') : H \text{ je povezan vpet podgraf grafa } G\}$$

in

$$P(H) = \sum_{e \in E'} v(e)$$

Nalogo imenujemo naloga o vpetem drevesu zato, ker velja

$$\mathrm{Min}(\Phi, P) \subseteq \{H : H \text{ je vpeto drevo grafa } G\}$$

Prirejanja

n ljudi mora opraviti n opravil. Stroški, ki jih imamo, če človek i opravi opravilo j, naj bodo a_{ij}. Ljudem moramo prirediti vsakemu po eno opravilo, tako da bodo skupni stroški najmanjši, pri čemer pa morajo biti opravljena vsa opravila.

Zastavljeni problem lahko prevedemo na optimizacijsko nalogo (S_n, P, Min), kjer je S_n množica vseh permutacij števil od 1 do n in

$$P(\pi) = \sum_{i=1}^{n} a_{i,\pi(i)}$$

Permutacijo π lahko opišemo tudi z dvojiško matriko X, določeno s predpisom

$$x_{ij} = \begin{cases} 1 & j = \pi(i) \\ 0 & \text{sicer} \end{cases}$$

. . . Prirejanja

Tedaj lahko nalogo o prirejanju zapišemo takole: (Φ, P, Min), kjer je $I = 1..n$,

$$\Phi = \{X \in \{0,1\}^{n^2} : \forall i,j \in I : \sum_{j \in I} x_{ij} = \sum_{i \in I} x_{ij} = 1\}$$

in

$$P(X) = \sum_{(i,j) \in I \times I} a_{ij} x_{ij}$$

Pogoji v opisu množice Φ zagotavljajo, da je X matrika neke permutacije. Iz tega zapisa naloge o prirejanju vidimo, da sodi med naloge (celoštevilskega) linearnega programiranja.

Trgovski potnik

Trgovec se je namenil, da bo obšel n mest tako, da se bo v vsakem mudil samo enkrat. Znane so razdalje d_{ij} med posameznimi mesti. Kako naj trgovec potuje, da bo opravil čim krajšo pot?

Pri formalizaciji naloge nadomestimo "zemljevid" z grafom povezanosti mest $G = (V, E)$. Točke tega grafa so mesta, povezave pa predstavljajo prometne zveze med mesti. Vsaki povezavi je pripisana še pripadajoča razdalja. Vsakemu trgovčevemu potovanju ustreza Hamiltonov cikel po grafu G. Tega lahko popišemo z urejeno n-terko točk π, sestavljeno po pravilu

$j = \pi(i)$ – točka j je v ciklu naslednik točke i

če naj π popisuje Hamiltonov cikel, mora biti ciklična permutacija.

. . . Trgovski potnik

Tako dobimo nalogo $(\Gamma_n, P, \mathrm{Min})$, kjer je Γ_n množica cikličnih permutacij števil od 1 do n in

$$P(\pi) = \sum_{i \in I} d_{i,\pi(i)}$$

Naloga o trgovskem potniku je torej podobna nalogi o prirejanju, vendar videz včasih vara. Za problem prirejanja obstaja učinkovit algoritem; problem trgovskega potnika pa sodi med "težke" probleme – domneva se, da zanj ni učinkovitega algoritma.