Univerza v Ljubljani
Fakulteta za matematiko in fiziko

Finančni praktikum

Algoritem za reševenje dvostopenjskega problema nahrbtnika z dinamičnim programiranjem

Jakob Zarnik, Tin Markon

Mentorja: prof. dr. Sergia Cabello Justo, asist. dr. Janoš Vidali

Ljubljana, 2020

Kazalo

Povzetek

V nalogi bova obravnavala reševanje dvostopenjskega problema nahrbtnika z dinamičnim programiranjem. Za izdelavo algoritma bova uporabila programski jezik Python.

1 Uvod

Dvostopenjski programi omogočajo modeliranje situacij, kjer glavni odločevalec, v nadaljevanju poimenovan investitor, optimizira svoja sredstva s tem, da neposredno upošteva odziv posrednika na njegovo odločitev o višini vložka. V primeru dvostopenjskega problema nahrbtnika (Bilevel Knapsack Problem), v nadaljevanju BKP, investitor določi prostornino nahrbtnika z namenom maksimizacije dobička, med tem ko se posrednik sooča z 0-1 problemom nahrbtnika s prostornino določeno s strani investitorja. BKP je ustrezen za modeliranje problema ≫ustreznega financiranja≪, kjer posameznik (tj. investitor), svoja sredstva razdeli med netvegano naložbo s fiksnim donosom (npr. varčevalni račun, državna obveznica) in bolj tvegano naložbo, preko posrednika kot je banka ali bančni posrednik (broker). Ta kupi delnice ali obveznice z namenom maksimizacije svojega dobička s tem, da upošteva omejitve finančnih sredstev (prostornina nahrbtnika) investitorja in ustvari donos z ustrezno izbiro investicij. Podobno uporabo modeliranja lahko opazimo na področju upravljanja s proizvodi, kjer se podjetje odloča koliko enot izdelkov naj proda samo in koliko preko posrednika.

BKP je mešan celoštevilski dvostopenjski problem predstavljen s strani Dempe in Richter, ki sta za rešitev predstavila "branch-and-bound"okvir (tj. razveji in omeji). V najini nalogi najprej razširiva potrebne in zadostne pogoje za obstoj optimalne rešitve. Nato predlagava enostaven in učinkovit algoritem dinamičnega programiranja za reševanje problema. V nasprotju s pristopom Dempe in Richter, kjer je beležen seznam nedominantnih rešitev, tukaj beležimo samo ciljne funkcijske vrednosti za oba, investitorja in posrednika, tekom dinamičnega procesa.

2 Formulacija in lastnosti problema

V narbtnik s prostornino oz kapaciteto y, ki jo določi investitor, vsakemu predmetu j določimo utež oz. volumen a_j, zaslužek posrednika c_j in zaslužek investitorja d_j. Ceno enote prostornine nahrbtnika označimo s t. Z danim y, posrednik izbere podmnožico predmetov, ki upošteva prostosrsko omejitev. To nam da dvostopenjski program

$$\text{BKP} = \begin{cases} \underset{y,x}{\text{Max}} f^1(y,x) = dx + ty \\ \text{s.t. } \underline{b} \leq y \leq \overline{b} \\ \underset{x}{\text{Max}} f^2(x) = cx \\ \text{s.t. } ax \leq y \text{ and } x \in \{0,1\}^n \end{cases}$$

kjer so a, c in d celoštevilske vrednosti in a, c, d, $\underline{b}$ in $\overline{b}$ nenegativne. V najini nalogi so z

$$\mathrm{S} = \{(x, y) \in \{0,1\}^n \times \left[\underline{b}, \overline{b}\right] : ax \leq y\}$$

označene omejitve, s

$$\mathrm{P}(y) = \{x \in \operatorname{Arg\,max}\{cx' : ax' \leq y, x' \in \{0,1\}^n\}\}$$

označimo posrednikovo racionalno izbiro množice (za fiksen y) in z

$$\mathrm{IR} = \{(x, y) | (x, y) \in S, x \in \mathrm{P}(y)\}$$

induktiven del, preko katerega investitor optimizira svojo funkcijo.

Dvostopenjski program obstaja v dveh različicah. Optimističen primer, ko racionalna množica ni singelton (enolična), posrednik izbere tisto rešitev, ki maksimizira zaslužek investitorja. Dobljena rešitev se imenuje močna rešitev. V pesimističnem primeru pa investitor predvideva, da kadar ima posrednik več enakovrednih možnosti izbire množice, izbere tisto, ki minimizira investitorjev zaslužek. Tako dobimo šibko rešitev.

Trditev 1 (Dempe in Richter)**.** *Če je cena enote prostornine (enota t) nepozitivna, potem obstaja optimalna rešitev BKP.*

Naslednja trditev povezuje ceno prostornine z investitorjevim razmerjem med zaslužkom in utežjo predmeta.

Trditev 2. *Naj bo* $\underline{b} = 0$ *in* $t < 0$*. Če je* $|t| > \max\limits_{1 \leq j \leq n} (\frac{d_j}{a_j})$*, potem je* $(y^*, x^*) = (0, 0_n)$ *optimalna rešitev.*

Dokaz. Naj bo (x, y) možna rešitev za dan BKP. Najprej z razširitvijo pogoja $ax \leq y$ s t $(t < 0)$ dobimo $(ta + d)x \geq ty + dx = f^1(y, x)$. Nato, ker je $ta_j + d_j < 0$ za $j = 1, \ldots, n$ in $x \in \{0,1\}^n$, sledi, da je $(ta + d)x \leq 0$, torej $f^1(y, x) \leq 0$. Vidimo, da ker je $(y^*, x^*) = (0, 0_n)$ možna rešitev danega BKP, v katerem je $f^1(y, x) = 0$, je ta rešitev tudi optimalna. □

Če je $\infty > \overline{b} \geq \sum_{i=1}^n a_i$ in $t > 0$, potem je optimalna rešitev trivialna: $x^* = (1, \ldots, 1)$ in $y^* = \overline{b}$. Če sta d in c kolinearna ($d = \alpha c$, kjer $\alpha > 0$) in $t \geq 0$, potem je reševanje BKP enako reševanju problema nahrbtnika s kapaciteto $\overline{b}$ za posrednika.

Definicija 1. *Diskreten dvostopenjski problem nahrbtnika (BKPd) je dvostopenjski problem nahrbtnika v katerem je spremenljivka, ki jo določi investitor diskretna.*

Trditev 3. *Če je* $t \leq 0$*, potem je vsaka optimalna rešitev* (y^*, x^*) *za BKPd, tudi optimalna rešitev za BKP.*

Če je $t > 0$ *in če optimalna rešitev za BKP obstaja, potem je optimalna tudi za BKPd.*

Dokaz. $t \leq 0$: Iz **Trditve 1** sledi, da optimalna rešitev (y^*, x^*) obstaja. Dodatno, IR (BKPd) $\subset$ IR (BKP) in iz Dempe in Richter sledi, da je y^* celo število.

$t > 0$: Direktno iz (ii) v Dempe in Richter. □

Iz **Trditve 3** sledi, da je reševanje BKP ekvivalentno reševanju BKPd, ko je t negativen. Če je t pozitiven in optimalna rešitev obstaja (glej Izrek 4 v Dempe in Richter), je ta dosežena v točki $(\overline{b}, x^*)$, kjer je $x^* \in P(\overline{b})$. Pomni, da optimalna rešitev BKPd vedno obstaja. Torej, če BKP ima optimalno rešitev, to lako dobimo z reševanjem zaporedja problemov nahrbtnika, ki vsebuje binarne spremenljivke, eno za vsak možno vrednost y. Algoritem, opisan v naslednjem poglavju, uporabi to lastnost.

3 Načrt za nadalnje delo

V nadaljevanju bova s programskim jezikom Python napisala program, ki s pomočjo dinamičnega programiranja izračuna optimalno rešitev (y^*, x^*) glede na naključno generirane podatke. S pomočjo primerov iz dokumenta bova preverila tudi, da program deluje pravilno. S pomočjo algoritma lahko izračunamo rešitev v primeru optimističnega primera, kot tudi pesimističnega. V najslabšem scenariju ima časovno zahtevnost $\theta(n\overline{b})$. Iz **Trditve 3** je razvidno, da je reševanje problema BKP ekvivalentno reševanju posrednikovega problema nahrbtnika za vsako celo število iz intervala $[\underline{b}, \overline{b}]$. Algoritem v svojem teku jemlje obe ciljni funkciji. To dosežemo z dvema fazama, ki sta podrobneje opisani spodaj.

Forward phase

Prva faza je sestavljena iz dveh zank: zunanja zanka s koraki $k \in [1, \ldots, n]$ in notranja zakna vezana na celoštevilsko kapaciteto nahrbtnika $y \in \left[\underline{b}, \overline{b}\right]$. Med to fazo sta generirani dve tabeli. Prva vsebuje optimalne vrednosti sledilca

$$f_k^2(y) = max \left\{ \sum_{j=1}^{k} c_j x_j : \sum_{j=1}^{k} a_j x_j \leq y, x \in \{0,1\}^k \right\}$$

druga pa optimalnevrednosti investitorja

$$\widetilde{f}_k^1(y) = max \left\{ \sum_{j=1}^{k} d_j x_j : x \in P(y) \right\}$$

na vsakem koraku k in za vsako kapaciteto y. Za graditev teh tabel z dinamicnim programiranjem se rekurzija izvede za vse vrednosti y med 0 in $\overline{b}$, za vsak predmet. Opomniti je treba, da funkcija $\widetilde{f}_k^1(y)$ ne upošteva ceno prostornine nahrbtnika, in je torej $\widetilde{f}_k^1(y) = f_k^1(y) + ty$.

for $k = 2, \ldots, n$ **and** $y = 0, \ldots \bar{b}$ **do**
  if $y < a_k$ **then**
    $f_k^2(y) = f_{k-1}^2(y)$ **and** $\widetilde{f}_k^1(y) = \widetilde{f}_{k-1}^1(y)$
  else
    $f_k^2(y) = max(f_{k-1}^2(y), f_{k-1}^2(y - a_k) + c_k)$
    if $f_{k-1}^2(y) \neq f_{k-1}^2(y - a_k) + c_k$ **then**
      $\widetilde{f}_k^1(y) = \widetilde{f}_{k-1}^1(y)$ **if** $f_k^2(y) = f_{k-1}^2(y)$
      $\widetilde{f}_k^1(y) = \widetilde{f}_{k-1}^1(y - a_k) + d_k$ **if** $f_k^2(y) = f_{k-1}^2(y - a_k) + c_k$
    else
      $\widetilde{f}_k^1(y) = max(\widetilde{f}_{k-1}^1(y), \widetilde{f}_{k-1}^1(y - a_k) + d_k)$ (Opt.)
      $\widetilde{f}_k^1(y) = max(\widetilde{f}_{k-1}^1(y), \widetilde{f}_{k-1}^1(y - a_k) + d_k)$ (Pes.)
    end if
  end if
end for

V prvem koraku (tj. $k = 1$) gledamo samo prvi predmet x_1. Optimalna rešitev investitorja in sledilca, za vsako kapaciteto y, je izračunana na sledeč način:

$$f_1^2(y) = \begin{cases} 0 \text{ for } y = 0, \ldots, a_1 - 1 \\ c_1 \text{ for } y = a_1, \ldots, \bar{b} \end{cases}$$

$$\widetilde{f}_1^1(y) = \begin{cases} 0 \text{ for } y = 0, \ldots, a_1 - 1 \\ d_1 \text{ for } y = a_1, \ldots, \bar{b} \end{cases}$$

Backtracking phase
Druga faza je uporabljena za iskanje optimalne rešitve (y^*, x^*), ki ustreza optimalni vrednosti določeni iz prejšnje faze. Optimalna kapaciteta y je generirana z n-tim stolpcem tabele investitiorja, kot je opisano v naslednji trditvi.

Trditev 4. *Naj bo* (x^*, y^*) *optimalna rešitev BKPd.*

- *Če je* $t \leq 0$*, potem* $\widetilde{f}_n^1(y^*) + ty^* = Max\left\{\widetilde{f}_n^1(y^*) + ty : y \in \{\underline{b}, \underline{b} + 1, \ldots, \bar{b}\}\right\}$
- *Če je* $t > 0$*, sta možnosti dve:* $(i) Max\left\{\widetilde{f}_n^1(y) + t(y+1)\right\} \leq \widetilde{f}_n^1(\bar{b}) + t\bar{b}$*, je* $(\bar{b}, x^*)$ *optimalna rešitev BKP, kjer je* $x^* \in P(y^*)$ *in* $y^* = \bar{b}$*; ali* (ii) *BKP nima optimalne rešitve.*

Iz optimalne rešitve vodje y^*, backtracking phase uporabi rekurzijo dinamičnega programiranja povezano z investitorjevim in sledilčevim problemom. Postopek v obliki psevdokode za to fazo je predstavljen spodaj.

Če se vodja sooča z enakovrednimi odločitvami, lahko spremenljivka x_k^* lahko zavzame vrednost 0 ali 1. Vrednost x_1^*, ki ni odvisna od rekurzije,

$y \leftarrow y^*$
for $f_{k-1}^2(y) \neq f_{k-1}^2(y - a_k) + c_k$ **do**
  if $y = 3$ **then**
    if $f_k^2(y) = f_{k-1}^2(y)$ **then**
      $x_k^* = 0$
    else
      $x_k^* = 1$ **and** $y \leftarrow y - a_k$
    end if
  else
    if $\widetilde{f}_k^1(y) = \widetilde{f}_{k-1}^1(y)$ **then**
      $x_k^* = 0$
    else
      $x_k^* = 1$ and $y \leftarrow y - a_k$
    end if
  end if
end for
if $f_1^2(y) = 0$ **then**
  $x_1^* = 0$
else
  $x_1^* = 1$
end if

je nastavljena na 0, če je $f_1^2(y) = 0$, in 1 v nasprotnem primeru. Če y^* ni enoličen, je za iskanje optimalne rešitve backtracking postopek uporabljen za vsak y^*.

Python koda za opisano fazo se nahaja v datoteki *BKP.py*, funkcija *backtracking_phase*. Vsebuje tudi komentarje za lažje raumenvanje funkcije.

Literatura

[1] L. Brotcorne, S. Hanafi, R. Mansi (2009). *A dynamic programming algorithm for the bilevel knapsack problem* Elsevier: Operations Research Letters 37, 215–218.

[2] S. Dempe, K. Richter (2000). *Bilevel programming with Knapsack constraint* European Newspaper of Operations Research 8, 93–107.