PRILAGOĐENI ZADACI IZ LINEARNOG PROGRAMIRANJA U SREDNJOŠKOLSKOM OBRAZOVANJU

JELENA MARKOVIĆ

ASISTENT NA MATEMATIČKOM FAKULTETU U BEOGRADU

PROFESOR U RAČUNARSKOJ GIMNAZIJI

JELENA_MARKOVIC@MATF.BG.AC.RS

JMARKOVIC@RG.EDU.RS

JELENA.MARKOVIC.10.11@GMAIL.COM

Sadržaj

O materijalu

Ovaj materijal kreiran je u okviru predavanja na Državnom seminaru za nastavnike 2022. godine

GLAVA 1

Uvod

Linearno programiranje je optimizaciona tehnika razvijena oko 1940. godine za potrebe federalne vlade. Primarni cilj joj je bio rešavanje problema alokacije resursa. Danas, primene zalaze u razne oblasti kao što su npr. poljoprivreda, medicina, menadžment, finansije, itd. Linearno programiranje koristimo svakodnevno npr. kada želimo da nađemo najkraći put od posla do kuće ili kada želimo da uz dostupne resurse obezbedimo maksimalni učinak našeg tima. Pa čak i kada planiramo dijetu.
Naravno, pre nego što se upustimo u dalja razmatranja, treba imati na umu da linearno programiranje nije tehnika koja može rešiti sve tipove problema. Stoga, treba da budemo u stanju da okarakterišemo probleme koje možemo rešiti.

Definicija 1.1. *Zadatak linearnog programiranja u standardnoj formi jeste pronaći vrednosti promenljivih* $x_1 \geq 0, x_2 \geq 0, ..., x_n \geq 0$ *i* $max(z)$ *tako da bude zadovoljeno:*

$$c_1 x_1 + c_2 x_2 + ... + c_n x_n = z(max)$$

$$a_{11} x_1 + a_{12} x_2 + ... + a_{1n} x_n \leq b_1$$

$$a_{21} x_1 + a_{22} x_2 + ... + a_{2n} x_n \leq b_2$$

$$...$$

$$a_{m1} x_1 + a_{m2} x_2 + ... + a_{mn} x_n \leq b_m$$

pri čemu su c_i, $a_i j$ *i* b_j *realni koeficijenti, a izraz koji je odredio z zovemo ciljnom funkcijom.*

Primetimo da je naziv linearno programiranje potekao iz toga što je ciljna funkcija linearna i što su ograničenja linearne (ne)jednakosti.

Oblik zadatka linearnog programiranja može da varira. Ono što je važno je da se uvek može svesti na standardnu formu. Razmotrimo sledeće slučajeve kada su narušena neka pravila standardne forme, i kako ih možemo korigovati da svedemo na standardnu formu:

- **Promenljiva nije nenegativna** - neka je x promenljiva koja nije nenegativna. Možemo je razložiti na druge dve x^+ i x^- koje jesu nenegativne, tako da važi $x = x^+ - x^-$ i ubaciti izmenu u sva ograničenja koja sadrže x.
- **Traži se minimiziranje a ne maksimiziranje ciljne funkcije** - u ovom slučaju dovoljno je iskoristiti činjenicu da važi $min(\sum_{i=1}^{n} c_i x_i) = -max(\sum_{i=1}^{n} c_i x_i)$.
- **Neka od ograničenja su oblika $\geq$ a ne $\leq$** - ako imamo neko ograničenje koje je oblika $a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \geq b_i$, dovoljno je da ga pomnožimo sa -1.
- **Neka od ograničenja su jednakosti** - svako ograničenje oblika $\sum_{j=1}^{n} a_{ij}x_j = b_i$ može se zameniti sa dva: $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ i $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$.

Definicija 1.2. *Vektor $x \in \mathbf{R}^n$ koji zadovoljava ograničenja zadatka linearnog programiranja je izvodljivo rešenje datog problema. Od svih izvodljivih rešenja, optimalno je ono koje minimizira odnosno maksimizira ciljnu funkciju (u zavisnosti od toga šta se traži).*

Nije teško videti da ako su obe strane jednog ili više ograničenja pomnožene nekom konstantom, optimalno rešenje novog problema identično je optimalnom rešenju datog. Ova tehnika se koristi da se postigne da svi koeficijenti linearnog programa budu približno iste veličine.
Interesantna činjenica je da postoji geometrijska interpretacija zadatka linearnog programiranja kao i njegovog rešenja. To možemo videti na sledećem primeru:

Primer 1.1

Neka je dat sledeći zadatak linearnog programiranja u standardnoj formi:

$$max(6x + 8y)$$

$$3x + y \leq 50$$

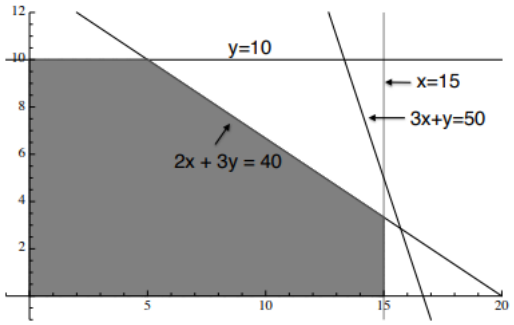
$$2x + 3y \leq 40$$

$$x \leq 15$$

$$y \leq 10$$

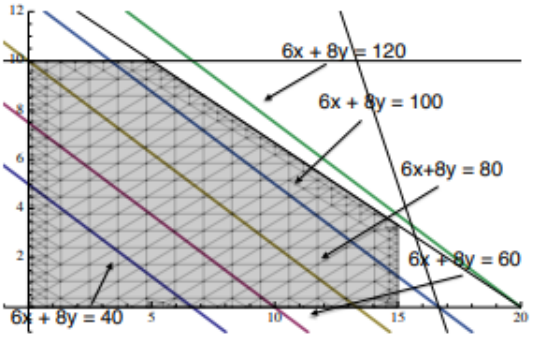
Geometrijski, ograničenja nam predstavljaju zatvorene poluravni. Mi tražimo presek tih poluravni, i oblast koja je presek nazivamo izvodljivom regijom. Kako presek može da bude konačan, beskonačan i prazan, tako i mi možemo da imamo beskonačno mnogo rešenja, konačno mnogo

ili da nemamo rešenje. Da bismo medju tim rešenjima pronašli optimalno (koje takođe ne mora da postoji), moramo da odgonetnemo geometriju ciljne funkcije.



Slika 1: Geometrijski prikaz primera 1.1

Pre nego što to učinimo, primetimo još i da je izvodljiva regija konveksna, jer nastaje kao presek konveksnih skupova. Na slici 1 je sivo išrafirana izvodljiva regija. Ovde je ona ograničena sa svih strana, i ono što je očigledno jeste da postoji rešenje. Ciljna funkcija je takođe linearna, u ovom slučaju je prava koju biramo iz skupa paralelnih pravih čije su jednačine oblika $6x+8y = a, a \in \mathbf{R}$. Intuitivno, krenuli bismo od nekog a za koje je prava $6x+8y = a$ blizu naše regije. Postepeno bismo menjali to a. Na slici 2 vidimo da sa porastom a raste i vrednost ciljne funkcije, u pravcu normalnom na prave $6x+8y = a$. Da bismo našli optimalno rešenje, važno je da znamo pravac u kome raste vrednost ciljne funkcije, ali i da ne napustimo izvodljivu regiju.



Slika 2: Traženje optimalnog rešenja - geometrijski prikaz

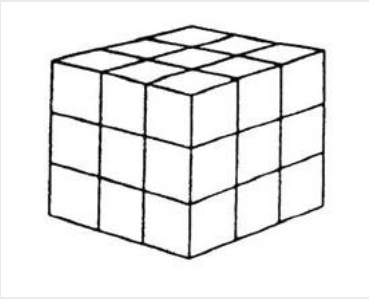
Ukoliko je izvodljiva regija ograničena, minimalna ili maksimalna vrednost ciljne funkcije (u zavisnosti od toga šta se traži), dostiže se u nekom od temena mnogougla koji je granica izvodljive regije. U suprotnom, optimalno rešenje ili ne postoji ili ako postoji, teme je otvorene mnogougao linije koja delimično ograničava izvodljivu regiju.

GLAVA 2

Neki zanimljivi zadaci

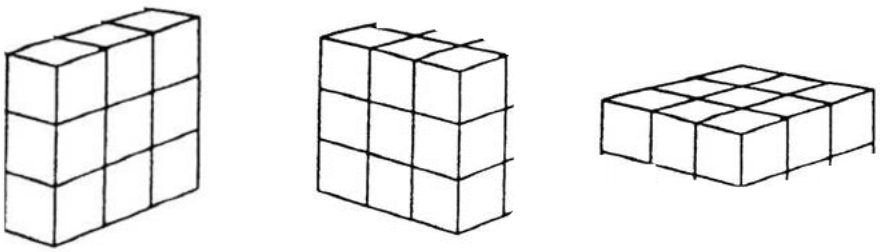
Primer 2.1

Dvadeset sedam ćelija je poređano u $3 \times 3 \times 3$ dimenzionalni niz kao na slici. Smatra se da tri ćelije leže u istoj liniji ako su na istoj horizontalnoj ili vertikalnoj liniji ili na istoj dijagonali. Dijagonale postoje na svakoj horizontalnoj i vertikalnoj sekciji a takođe spajaju naspramna temena kocke. Obojiti 13 ćelija u belu boju i 14 ćelija u crnu, tako da se minimizuje broj linija u kojima su sve ćelije obojene istom bojom.



Rešenje: Navedeni primer poseduje tipičan kombinatorni karakter koji se javlja u većini problema koji se rešavaju celobrojnim linearnim programiranjem. Očigledno je da ovde postoji gomila načina da se ćelije oboje. Ovakvi zadaci se obično ispostave teški da se rešavaju direktno celobrojnim linearnim programiranjem, već se najpre tretiraju nekim heuristikama kojima se teži da se izbaci što veći broj neizvodljivih rešenja. Ipak, naš cilj neće biti efikasnost, već usvajanje

načina za preformulisanje problema u zadatke linearnog programiranja.
Krenimo od određivanja promenljivih. Ono što je očigledno jeste da svakoj ćeliji moramo pridružiti po jednu promenljivu δ_j, $j = 1..27$ koja će nositi u sebi informaciju da li je ćelija koju predstavlja obojena belo ili crno. Možemo usvojiti konvenciju da ako je $\delta_j = 1$ to znači da je ćelija j crna, a ako je $\delta_j = 0$ da je ćelija j bela.
Manje očigledna je činjenica da za svaku liniju moramo imati promenljivu koja nju predstavlja. Ako pogledamo kocku iz različitih uglova, vidimo da se izdvajaju tri tipa slojeva, koje imamo date na slici 3. Svaki od tih tipova slojeva imamo po 3 u našoj kocki. U svakom imamo po dve dijagonale, što znači da ukupno imamo $3 * 3 * 2 = 18$ dijagonala. Sa druge strane, odabir ovih slojeva nam sugeriše da imamo dva tipa horizontalnih linija i jedan tip vertikalnih. Od svakog od tih tipova horizontalnih i vertikalnih linija imamo po 9 linija što znači da imamo ukupno 27 linija koje nisu dijagonale. Na kraju, ostaju nam još četiri dijagonale za koje važi da ne pripadaju istom sloju (one koje sadrže središnju ćeliju). Sledi da ukupno imamo 49 linija. Promenljive koje odgovaraju linijama označimo sa γ_i gde $i = 1..49$. Uvedimo konvenciju da je $\gamma_i = 1$ ako su sve ćelije u liniji i iste boje i 0 u suprotnom.



Slika 3: Pogledi na kocku iz različitih uglova

Da bismo osigurali da promenljive γ_i zaista predstavljaju ono što je izrečeno, nad njima uvodimo ograničenja koja su nam data u zadatku. Modelujemo ih tako da iz uslova $\gamma_i = 0$ sledi da je $\delta_{i_1} + \delta_{i_2} + \delta_{i_3} \geq 1$ i $\delta_{i_1} + \delta_{i_2} + \delta_{i_3} \leq 2$, gde su $\delta_{i_1}, \delta_{i_2}, \delta_{i_3}$ ćelije koje čine dijagonalu γ_i. Ovi uslovi nam osiguravaju da sigurno imamo bar jednu crnu i bar jednu belu u našem odabiru boja za ćelije posmatrane dijagonale. Veoma važna činjenica jeste da uslovi navedeni u ovom obliku nisu ograničenja linearnog programiranja. Da bi to postali, moraju biti izraženi kao:

$$\delta_{i_1} + \delta_{i_2} + \delta_{i_3} - \gamma_i \leq 2$$

$$\delta_{i_1} + \delta_{i_2} + \delta_{i_3} + \gamma_i \geq 1$$

Ove uslove moramo navesti za svako $i = 1..49$. Primetimo da oni ne garantuju da je $\gamma_i = 1$ ako su sve ćelije iste boje na liniji, ali to će biti osigurano minimizovanjem ciljne funkcije. Ostaje nam da dodamo uslov $\sum_{j=1}^{27} \delta_j = 14$ koji nam kaže da tražimo da 14 ćelija od 27 bude crno (što automatski znači da će ih 13 biti belo) i da proglasimo $\sum_{i=1}^{49} \gamma_i$ za našu ciljnu funkciju koju minimizujemo. Ovim je problem u potpunosti modelovan kao zadatak celobrojnog linearnog programiranja i može biti prepušten nekom alatu koji će ga rešiti za nas.

Primer 2.2

Postoji šest sela u zemlji Brutopiji. Plemić je dobio ovu zemlju kao feudalni posed od kralja. On treba da rasporedi vitezove u neka od sela (dozvoljeno je i u sva sela), tako da oni mogu da uguše pobune kmetova. Kada u nekom selu izbije pobuna, smatra se da je ona ugušena ako u nju za najviše 15 sati stigne vitez. Ako je vitez već u tom selu, smatra se da mu treba 0 sati da stigne u to selo. Svaki raspoređeni vitez prima istu platu. Kako je plemić cicija, on želi da minimizuje broj vitezova koje šalje. Podrazumevati da nakon što izbije jedna pobuna, druga neće izbiti barem 30 sati nakon toga, koliko je vitezu potrebno da se vrati u svoje selo.

From/To	1	2	3	4	5	6
1	0	10	20	30	30	20
2	10	0	25	35	20	10
3	20	25	0	15	30	20
4	30	35	15	0	15	25
5	30	20	30	15	0	14
6	20	10	20	25	14	0

Rešenje: Na osnovu tabele, možemo da zaključimo sledeće:

- Ako plemić postavi viteza u selo 1, taj vitez može da uguši pobune u selima 1 i 2.
- Ako plemić postavi viteza u selo 2, taj vitez može da uguši pobune u selima 1, 2 i 6.
- Ako plemić postavi viteza u selo 3, taj vitez može da uguši pobune u selima 3 i 4.
- Ako plemić postavi viteza u selo 4, taj vitez može da uguši pobune u selima 3, 4 i 5.
- Ako plemić postavi viteza u selo 5, taj vitez može da uguši pobune u selima 4, 5 i 6.
- Ako plemić postavi viteza u selo 6, taj vitez može da uguši pobune u selima 2, 5 i 6.

Promenljive su x_i, $i = 1..6$. Po konvenciji, $x_i = 1$ ako u selo i treba postaviti viteza, a 0 u suprotnom. Ciljna funkcija koju treba maksimizovati je data sa $x_1 + x_2 + x_3 + ... + x_6$.
Zadatak rešavamo promenom pristupa. Umesto da razmatramo šta se dešava kada plemić odluči da postavi viteza u neko selo, razmatraćemo šta će da se dogodi ako u tom selu nema viteza. Ako ne stavimo viteza ni u selo 1 ni u selo 2, pobuna u selu 1 ne može biti ugašena. Slično ako ne stavimo viteza u sela 1, 2, 6 istovremeno, onda pobuna u selu 2 ne može biti ugašena. Ako ne stavimo viteza u sela 3 i 4 istovremeno, onda pobuna u selu 3 ne može biti ugašena. Nastavljanjem ovog postupka, dolazimo do sledećih ograničenja:

$$x_1 + x_2 \geq 1$$

$$x_1 + x_2 + x_6 \geq 1$$

$$x_3 + x_4 \geq 1$$

$$x_3 + x_4 + x_5 \geq 1$$

$$x_4 + x_5 + x_6 \geq 1$$

$$x_2 + x_5 + x_6 \geq 1$$

U zadatku linearnog programiranja ne možemo direktno da kažemo da je neka promenljiva x jednaka 0 ili 1 već to modelujemo na sledeći način: dodajemo ograničenja $x \geq 0$ i $x \leq 1$ i oznaku da je x celobrojno.

Primer 2.3

Doktor Strejndžlav odlučio je da koristeći kućnu hemiju napravi neke specifične super dobre tečnosti za čišćenje za masovnu proizvodnju. U tabeli je prikazano šta mu je za koju tečnost potrebno i koliko može da zaradi prodajom jedne bočice takve tečnosti.

Tečnost	Sati rada u labi	Litara kućne hemije	Profit u evrima
A	3	4	6
B	2	3	4
C	6	4	7

Doktor Strejndžlav ima na raspolaganju 160 litara kućne hemije i 150 sati rada u laboratoriji. Ako doktor proizvodi tečnost X, za to treba da iznajmi mašinu za pravljenje tečnosti X. Iznajmljivanje mašine za tečnost A košta 200 evra, za tečnost B 150, a za tečnost C 100 evra. Koliko bočica koje tečnosti doktor treba da proizvede da bi maksimizovao profit?

Rešenje: Označimo sa x_A, x_B i x_C promenljive koje određuju broj bočica tečnosti A, B i C koje doktor treba da napravi. Dodatno, treba uvesti promenljive y_A, y_B i y_C koje služe kao indikatori da li će se tečnost odgovarajućeg tipa proizvoditi i da li treba iznajmiti mašinu za pravljenje te tečnosti. Ukoliko se neka tečnost ne proizvodi, onda je jasno da je iznajmljivanje mašine za pravljenje te tečnosti dodatan i nepotreban trošak. Ciljna funkcija u ovom slučaju odnosi se na profit i ona glasi:

$$6 * x_A + 4 * x_B + 7 * x_C - 200 * y_A - 150 * y_B - 100 * y_C$$

Ograničenja koja se odnose na sate rada u laboratoriji i količine dostupne hemije su sledeća:

$$3 * x_A + 2 * x_B + 6 * x_C \leq 160$$

$$4 * x_A + 3 * x_B + 4 * x_C \leq 150$$

Sada dolazimo do najtežeg dela ovog zadatka. Promenljive y treba dovesti u vezu sa x. Odnosno, treba da važi sledeće: Kada je $y_i = 1$ tada se tečnost i proizvodi ($i \in \{A, B, C\}$), a u suprotnom je $y_i = 0$. Naredni trik će nam rešiti problem. Odaberimo neki broj M za koji važi da je dovoljno veliki (veći od svih brojeva koji se javljaju u problemu) da ne utiče na prostor rešenja i uvedimo sledeća ograničenja:

$$x_A \leq y_A * M, x_B \leq y_B * M, x_C \leq y_C * M$$

Izvršimo sada analizu. Ako je, bez umanjenja na opštosti, $y_A = 0$, tada ograničenje $x_A \leq y_A * M$ postaje $x_A \leq 0$. Pošto je $x_A \geq 0$, odatle sledi da je $x_A = 0$. Ako je $y_A = 1$, ograničenje i dalje važi,

jer je M dovoljno veliki broj. Sa druge strane, ako je $x_A = 0$ šta god da biramo za y_A navedeno ograničenje će biti zadovoljeno, a kako je nama cilj maksimizacija ciljne funkcije, biće izabrano $y_A = 0$. Ako je $x_A = 1$, tada očigledno mora da važi $y_A = 1$. Ovim dolazimo do zaključka da su naša novouvedena ograničenja dovoljna da simuliraju vezu između ove dve grupe promenljivih, tj. važi $y_i = 1$ akko se tečnost i proizvodi ($i \in \{A, B, C\}$).

Primer 2.4

Na raspolaganju imamo 4 mašine. 4 posla treba da budu kompletirana. Svakoj mašini treba da dodelimo tačno jedan posao. Svaki posao treba da radi tačno jedna mašina. Za svaku mašinu nam je poznato koliko vremena joj treba da završi dati posao. Minimizovati ukupno vreme rada mašina.

Mašine\Posao	1	2	3	4
1	14	5	8	7
2	2	12	6	5
3	7	8	3	9
4	2	4	6	10

Ovaj problem je jedna verzija istorijski značajnog problema linearnog programiranja pod nazivom Problem dodele (engl. *Assignment problem*). U tom problemu cilj je dodeliti zadatke zaposlenima tako da svaki zaposleni radi tačno jedan zadatak i da svaki zadatak bude dodeljen tačno jednom zaposlenom. Priča koja je kasnije osmišljena da bi popularizovala problem je Problem braka (engl. *Marriage problem*). Ideja je bila povezati devojke za udaju sa neženjama.

Rešenje: Označimo sa x_{ij} promenljivu koja ima vrednost 1 ako je mašini i dodeljen posao j, a 0 u suprotnom. Ciljna funkcija koju minimizujemo data je sa:

$$14x_{11} + 5x_{12} + 8x_{13} + 7x_{14} + 2x_{21} + 12x_{22} + 6x_{23} + 5x_{24} +$$

$$+7x_{31} + 8x_{32} + 3x_{33} + 9x_{34} + +2x_{41} + 4x_{42} + 6x_{43} + 10x_{44}$$

Ograničenja koja su nam data su sada u obliku jednakosti. Imamo dva tipa ograničenja: svakoj mašini je dodeljen tačno jedan posao, i svaki posao radi tačno jedna mašina. Prvi tip ograničenja može se zapisati kao $x_{i1} + x_{i2} + x_{i3} + x_{i4} = 1$ za $i = 1..4$, dok se drugi tip ograničenja može zapisati kao $x_{1j} + x_{2j} + x_{3j} + x_{4j} = 1$ za $j = 1..4$.

Primer 2.5

Brodograditelj pravi 3 tipa brodova: čamce, kanue i kajake. Prodaje ih za 200, 150 i 100 dolara redom. Brodovi zahtevaju aluminijum koga imamo na raspolaganju 630 kg. Takođe, u firmi imamo dva tima, na koja možemo da podelimo rad i vreme koliko im treba da izrade jedan brod datog tipa. Prvi tim može da radi ukupno 110 sati, a drugi 50. Koliko brodova koje vrste treba da izradi koji tim da bismo maksimizovali zaradu?

	čamac	kanu	kajak
aluminijum	30kg	15kg	10kg
Tim 1	2h	3h	2h
Tim 2	1h	1h	1h

Rešenje: Ovaj zadatak je drugačiji od drugih zbog toga što nije dovoljno uvesti promenljive x, y i z koje nam govore koliko čamaca, kanua i kajaka treba da napravimo da bismo maksimizovali zaradu. Umesto toga je potrebno podeliti te promenljive na dve, koje nam govore koliko koji tim treba da napravi čamaca, kanua i kajaka. Neka je x_1 broj čamaca koje treba da napravi tim 1, a x_2 broj čamaca koje treba da napravi tim 2. Slično, uvedimo i y_1, y_2 za kanue i z_1, z_2 za kajake. Ciljna funkcija glasi:

$$200 * x_1 + 200 * x_2 + 150 * y_1 + 150 * y_2 + 100 * z_1 + 100 * z_2$$

Ograničenje koje se tiče aluminijuma je:

$$x_1 * 30 + x_2 * 30 + y_1 * 15 + y_2 * 15 + z_1 * 10 + z_2 * 10 \leq 630$$

Ograničenja koja se odnose na to koliko koji tim sati može da radi:

$$x_1 * 2 + y_1 * 3 + z_1 * 2 \leq 110$$

$$x_2 + y_2 + z_2 \leq 60$$

GLAVA 3

Istorijski važni zadaci linearnog programiranja

3.1 Stiglerov problem dijete

Godine 1945. nobelovac Džordž Stigler napisao je članak "Cena egzistencije" na naizgled veoma jednostavnu temu - kako prehraniti vojnika sa što manje novca. "Stiglerov problem dijete" postao je klasični problem linearnog programiranja, oblasti računarstva koja je kasnije otkrivena. Stigler je sam uspeo da pronađe rešenje vrlo blisko optimalnom za problem malih dimenzija, metodom različitom od linearnog programiranja.
Osnovna ideja samog problema je jednostavna: izabrati od svakog tipa hrane koji je na raspolaganju količinu koju treba uneti da bi se zadovoljile nutricione potrebe, a da se pritom osiguramo da je cena dijete razumna. Poznavajući osnovne pojmove linearnog programiranja koje smo do sada videli, možemo da naslutimo da bi ovaj problem mogao da se razvija na dva načina. U prvom bi moglo da nam se traži da minimizujemo cenu dijete a da nutriciona ograničenja ostanu zadovoljena, a u drugom da maksimizujemo nutricioni sadržaj dijete a da ostanemo u okvirima predviđenog budžeta. Druga verzija problema ipak nema praktičnog smisla, iz razloga što bismo u tom slučaju imali poteškoće pri osmišljavanju nutricionih ograničenja, s obzirom da nutritivni sastojci mogu da budu izraženi u različitim jedinicama što nam onemogućava da ih prosto saberemo.
Formalno, sam problem se može zapisati na sledeći način:

- **promenljive:**
  - x_j kvantitet (količina izražena u nekim jedinicama) hrane j u dijeti
- **koeficijenti:**
  - a_{ij} količina nutritivnog sastojka i u hrani j

- c_j cena jedinice količine hrane j
- $\overline{b_i}(\underline{b_i})$ najveći (najmanji) prihvatljivi unos nutritivnog sastojka i u dijeti

- **ograničenja:**
  - $\underline{b_i} \leq \sum_j a_{ij} x_j \leq \overline{b_i}, \forall i$
  - $x_j \geq 0, \forall j$

- **ciljna funkcija:**
  - $\sum_j c_j x_j$ da se minimizuje

Više informacija o samom problemu i njegova implementacija u alatu OR-Tools može se naći na linku https://developers.google.cn/optimization/lp/stigler_diet#c++_1.

3.2 Problem sečenja (engl. *Cutting stock problem*)

Problem sečenja i njegove varijacije opisani su u ranim danima linearnog programiranja. Formulisani su 1961. godine. Najjednostavnije rečeno, dati su materijali dostupni u različitim oblicima i veličinama koje je dozvoljeno seći da se dobiju određeni oblici zadatih veličina koji se potom nekako uklapaju u cilju minimizacije određenog cilja. Razlikujemo jednodimenzione, dvodimenzione i trodimenzione probleme.

Primer 3.1

Pretpostavimo da se rolne ukrasnog papira prave uniformne širine od 100 inča i da mogu da se naprave narudžbine za rolne širine 14 inča, 31, 36 i 45 inča. Kompanija je dobila sledeće narudžbine: 211 rolni ukrasnog papira širine 14 inča, 395 rolni širine 31 inča, 610 rolni širine 36 inča i 97 rolni širine 45 inča. Rolna od 100 inča može da se iseče na dve od 45 inča, pri čemu otpada materjal širine 10 inča, ali takođe može da se iseče na jednu od 45 inča, jednu od 31 i jednu od 14 inča. Svaka kombinacija sečenja rolne od 100 inča naziva se šablon. Za ovaj primer postoji 37 različitih šablona. Odrediti koliko kojih šablona sečenja treba primeniti, tako da se minimizuje količina materijala koji otpada.

Rešenje: Neka je I skup širina, a J skup različitih šablona. Naš zadatak linearnog programiranja dat je sa:

- **promenljive:**
  - x_j broj rolni koje se seku u šablon j

- **koeficijenti:**
  - a_{ij} broj rolni širine i u šablonu j
  - b_i broj naručenih rolni širine i
  - c_j širina materijala koji otpada pri šablonu j

- **ciljna funkcija:**
  - $\sum_{j \in J} c_j x_j$ da se minimizuje
- **ograničenja:**
  - $\sum_{j \in J} a_{ij} x_j \geq b_i, \forall i \in I$

3.3 Raspored za zaposlene (engl. *Employee Scheduling*)

Problem se odnosi na pravljenje rasporeda za zaposlene na poslovima gde postoje smene, kao što su npr. vozači autobusa, medicinske sestre, itd. Zapravo, ovaj problem je prvi deo velikog zadatka pravljenja rasporeda smena za zaposlene, i on ne podrazumeva dodelu smena konkretnom zaposlenom. Takođe, nisu uzete u obzir posebne sposobnosti zaposlenih.
Ako želimo da formalno zapišemo problem, pretpostavimo da su nam dati neki vremenski intervali numerisani sa $j = 1, 2, ..., n$ i da je najmanji broj zaposlenih koji u intervalu j moraju biti prisutni b_j. Označimo promenljivu koja se odnosi na broj zaposlenih u intervalu j sa x_j. Uvedimo konvenciju $x_{n+1} = x_1$ i $b_{n+1} = b_1$ kako bismo olakšali zapis. Kakva ćemo ograničenja uvesti, dalje zavisi od toga kako se dužine intervala odnose prema dužini smena. Npr. ako su vremenski intervali dužine 4h, i svaka smena traje 8h, onda se za zaposlenog koji dolazi na posao na početku intervala j smatra da je prisutan u vremenskim intervalima j i $j+1$. Pod ovim uslovom, problem bi se formalno zapisao kao:

- **promenljive:**
  - x_j broj zaposlenih prisutnih u vremenskom intervalu j
- **koeficijenti:**
  - b_j najmanji broj zaposlenih koji mora biti prisutan u vremenskom intervalu j
- **ciljna funkcija:**
  - $\sum_j x_j$ da se minimizuje
- **ograničenja:**
  - $x_j + x_{j+1} \geq b_{j+1}, \forall j = 1..n$
  - $x_j \geq 0$

Matrica koeficijenata za ograničenja ima specijalnu strukturu. To je tkzv. Hofman-Kruskalova matrica:

$$a_{ij} = \begin{cases} 1, & \text{if i=j} \\ 1, & \text{if i=j+1} \\ 1, & \text{if i=1 and j=n} \\ 0, & \text{u suprotnom} \end{cases}$$

Pravljenje rasporeda smena može da se radi za jedan dan, više dana, nedelja, mesec dana, itd.

Primer 3.2

Napraviti raspored za smene medicinskih sestara za jedan dan, prema tabeli:

	Midnight – 4 a.m.	4 a.m. – 8 a.m.	8 a.m. – 12 noon	12 noon – 4 p.m.	4 p.m. – 8 p.m.	8 p.m. – midnight
Time slot #	1	2	3	4	5	6
Nurses needed	6	8	11	9	18	11

Slika 4: Vremenenski intervali u toku radnog dana jedne bolnice sa informacijama o potrebnom osoblju

3.4 Problemi mešanja (engl. *Blending problems*)

Godine 1952. napisan je rad o mešanju goriva za avione sa dodatnim hemikalijama koje bi poboljšale svojstva ovih goriva. Cilj je bio maksimizovati profit.
Opšta postavka problema je sledeća. Imamo na raspolaganju m tipova sirovih materijala i njihovim mešanjem može da se dobije n tipova konačnih proizvoda. Količine sirovih materijala su ograničene i označene su sa s_i. Pretpostavka je da se materijali mešaju linearno, tj. ako pomešamo α jedinica materijala A i β jedinica materijala B imaćemo mešavinu C čija su svojstva proporcionalna količini materijala A i B koje smo mešali. Na primer, ako pomešamo 3 galona vode čija je temperatura 80 stepeni celzijusa, i 2 galona vode čija je temperatura 100 stepeni celzijusa, dobićemo 5 galona vode čija je temperatura $(3*80+2*100)/5 = 88$ stepeni celzijusa. Za svaki konačan proizvod j data je količina tog proizvoda d_j koju na kraju treba da imamo. Cena jedinice količine i-tog sirovog materijala je c_i, dok je cena po kojoj prodajemo jedinicu količine j-tog konačnog proizvoda p_j. U cilju kontrole kvaliteta mešavina, neka pravila moraju da se ispoštuju, a to je da najmanji procenat sirovog materijala i u konačnom proizvodu j mora da bude $\underline{a_{ij}}$, a najveći $\overline{a_{ij}}$. Npr. ako je $\underline{a_{ij}} = 0.4$ i $\overline{a_{ij}} = 0.75$, to znači da je procenat sirovog materijala i u konačnom proizvodu j između 40 i 75 posto. Naš cilj je maksimizacija zarade. Opisani zadatak linearnog programiranja dat je sa:

- **promenljive:**
  - x_{ij} koliko jedinica sirovog materijala i učestvuje u finalnom proizvodu j
- **koeficijenti:**
  - c_i cena jedne jedinice količine sirovog materijala i
  - p_j cena po kojoj prodajemo jednu jedinicu količine konačnog proizvoda j
  - s_i koliko jedinica količine sirovog materijala i imamo na raspolaganju
  - d_j koliko jedinica količine konačnog proizvoda treba da dobijemo mešanjem
  - $\underline{a_{ij}}$ koliko najmanje procenata sirovog materijala i smemo da imamo u konačnom proizvodu j

- $\overline{a_{ij}}$ koliko najviše procenata sirovog materijala i smemo da imamo u konačnom proizvodu j

- **ciljna funkcija:**
  - $\sum_j p_j \sum_i x_{ij} - \sum_i c_i \sum_j x_{ij}$ da se maksimizuje

- **ograničenja:**
  - $\sum_j x_{ij} \leq s_i, \forall i$
  - $\sum_i x_{ij} = d_j, \forall j$
  - $x_{ij} \geq \underline{a_{ij}} \sum_k x_{kj}, \forall i, j$
  - $x_{ij} \leq \overline{a_{ij}} \sum_k x_{kj}, \forall i, j$
  - $x_{ij} \geq 0, \forall i, j$

Primer 3.3

Kompanija se suočava sa problemom mešanja tri tipa sirovih materijala kako bi se dobila dva tipa konačnih proizvoda. Podaci su dati u tabeli.

Sirovi materijali/proizvodi	1 2	Dostupne količine (u tonama)	Cena po jedinici (u dolarima)
1 2 3	[.4, .6] [.5, .6] [.1, .2] [.1, .4] [.2, .5] [.2, .3]	2000 1000 500	1.00 1.50 3.00
Potrebna količina (u tonama)	600 700		
Prodajna cena (u dolarima)	10 8		

3.5 Problem trgovačkog putnika (engl. *Travelling salesman problem*)

Problem trgovačkog putnika (nadalje u tekstu TSP) je jedan od najpoznatijih NP-teških problema u računarstvu. On glasi:"Neka je data lista gradova i njihova međusobna rastojanja, koja je najkraća moguća putanja da posetimo svaki grad tačno jedan i da se vratimo u grad od koga smo krenuli?"
TSP problem se može formulisati kao problem celobrojnog linearnog programiranja. Postoji mnogo načina na koji je to urađeno, a mi ćemo se ovde osvrnuti na Miler-Taker-Zemlinovu formulaciju (MTZ). Pretpostavka je da su gradovi poređani u niz, i da je prvi odabran kao početni grad.

- **promenljive:**
  - x_{ij} vrednost ove promenljive je 1 ako postoji direktan put od i-tog do j-tog grada u odabaranoj putanji, u suprotnom je 0
  - u_i rang čvora u odabranoj putanji

- **koeficijenti:**

- c_{ij} rastojanje od grada i do grada j

- **ciljna funkcija:**
  - $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$ da se minimizuje

- **ograničenja:**
  - $\sum_{i=1}^{n} x_{ij} = 1$ u svaki grad j stižemo iz tačno jednog grada
  - $\sum_{j=1}^{n} x_{ij} = 1$ iz svakog grada i idemo u tačno jedan grad
  - $u_i + x_{ij} \leq u_j + (n-1)*(1-x_{ij}), 2 \leq j \leq n$ ova ograničenja se zovu i eliminacija podruta. Govore nam sledeću jednostavnu činjenicu, a to je da ako je $x_{ij} = 1$, to povlači da je $u_j = u_i + 1$. Takva ograničenja nisu linearna, pa su zato preformulisana u oblik koji je ovde dat.

GLAVA 4

Alat OR-Tools

Alat OR-Tools je softver otvorenog koda pogodan za matematičku optimizaciju, napravljen od strane kompanije Google. Poseduje podršku za rešavanje problema rutiranja, mrežnog protoka, zadataka celobrojnog i linearnog programiranja, kao i programiranja ograničenja.
Nakon što modelujemo problem, možemo izabrati rešavač koji želimo da upotrebimo. To može da bude neki komercijalni rešavač (Gurobi, CPLEX,...) ili neki rešavač otvorenog koda (GLPK SCIP, GLOP,...).
Modelovanje zadataka linearnog programiranja možemo da uradimo uz pomoć 4 programska jezika i to su: Python, Java, C++ i C#. U zavisnosti od toga koji jezik želimo da koristimo i na kom operativnom sistemu radimo razlikuju se i uputstva za instalaciju. O instalaciji može se više pročitati na linku `https://developers.google.com/optimization/install`.
Autor ovog materijala koristi Ubunutu 16.04 kao svoj operativni sistem, ali se smernice ovde date mogu lako primeniti i na druge operativne sisteme. Nakon instalacije, može se uočiti folder pod nazivom *examples*. U tom folderu treba napraviti odabrati folder *cpp* (pod uslovom da želite da radite u C++-u, kao što je autor ovde izabrao) i unutar tog foldera treba napraviti svoj fajl u kome se radi modeliranje zadatka i pozivanje odgovarajućeg rešavača. U samom Makefile-u treba dodati u delu *EXE* liniju $(BIN\_DIR)/1\$E$ ako se fajl u kome smo izvršili modeliranje zove 1.cc. Nakon što to uradimo u folder *bin* (koji je u istom direktorijumu kao *examples*) će se izgenerisati izvršni fajl pod nazivom 1.

4.1 Ugrađene mogućnosti

Najvažnija klasa koja je za nas dostupna u okviru biblioteke *linear_solver.h* jeste *MPSolver*. Ona je omotač za naš rešavač i u okviru nje ne samo da možemo podesiti koji rešavač želimo da ko-

ristimo, već i koliko vremena želimo da dozvolimo rešavaču da traži rešenje.
Klase koje nam pomažu da definišemo promenljive, ograničenja i ciljnu funkciju su redom: *MPVariable*, *MPConstraint* i *MPSolver*. Promenljivu generišemo pozivanjem metode *MakeNumVar* ukoliko je to realna promenljiva, i *MakeIntVar* ako je to celobrojna promenljiva. Oba metoda zahtevaju tri argumenta: donje ograničenje za promenljivu, gornje ograničenje za promenljivu i naziv pod kojim će biti ispisano rešenje. Ukoliko neka promenljiva nije ograničena, na raspolaganju nam je specijalna konstanta za beskonačnost, koju dobijamo pozivom metode *infinity* nad objektom klase *MPSolver*.
Ograničenja generišemo pozivom metode *MakeRowConstraint* nad objektom klase *MPSolver*. Ova metoda prima dva argumenta, donje i gornje ograničenje za linearni izraz kojim je ograničenje određeno. Pozivom metode *SetCoefficient* nad objektom tipa *MPConstraint* koja prima dva argumenta, objekat tipa *MPVariable* i broj, može se podesiti koliko je koeficijent uz tu promenljivu u linearnom izrazu kojim je ograničenje određeno.
Veoma slično se generiše cilj nad kojim možemo podesiti da li želimo maksimizaciju (pozivom metoda *setMaximization*) ili minimizaciju (pozivom metoda *SetMinimization*).
Pozivanjem metoda *Solve* nad objektom tipa *MPSolver* poziva se rešavač da reši problem. Nakon toga, možemo dobiti informacije o tome da li je pronađeno optimalno rešenje u zadatom periodu, kako glasi to rešenje, koliko vremena je bilo potrebno da se ono nađe, koliko čvorova u stablu pretrage smo obišli, itd.
Postoje i neke složenije ugrađene mogućnosti, od kojih ćemo neke pomenuti na predavanje, a druge ostaju za samostalno istraživanje.

GLAVA 5

Literatura

- [1] Eiselt H.A., Sandblom C.L.: *Linear programming and its applications*, Springer-Verlag, Berlin Heidelberg, 2007.
- [2] https://stemez.com/subjects/science/1HOperationsReseach/1HOperationsReseach.php
- [3] Dantzig G., Thapa M.: *Linear programming - Introduction*, Springer, 1997.
- [4] Tadeusz, S.: *A note on the Miller-Tucker-Zemlin model for the asymmetric traveling salesman problem*, Bulletin of the Polish Academy of Sciences Technical Science, 2016.
- [5] https://developers.google.com/optimization
- [6] https://math.libretexts.org/Courses/Community_College_of_Denver
- [7] Kolman B., Beck R.: *Elementary Linear Programming with Applications*, Academic Press, 1995.