Univerza v Mariboru
Fakulteta za naravoslovje in matematiko

Linearni problem prevoza

Eva Zmazek

Maribor, 2019

1 Opredelitev problema linearnega prevoza

2 Problem linearnega prevoza

Iz skladišč želimo prepeljati večje količine izdelka na različne lokacije (trgovine). Imamo n skladišč in k trgovin. Skladišča imajo vsaka svojo zalogo izdelka, vsaka trgovina pa želi prejeti določeno količino izdelka. Za prevoz izdelka iz i-tega skladišča v j-to trgovino imamo v naprej določeno ceno prevoza. Izračunati je potrbno, kolikšen je minimalen strošek prevoza, če želimo izpolniti vse zahteve trgovin. Problem je **linearen**, če strošek prevoza iz skladišča v trgovino ni odvisen od količine, ko jo prevozimo, torej je cena prevoza na izdelek konstanta.

Definirajmo:

- $sour(i)$ ... **zaloga** izdelka v i-tem skladišču
- $dest(j)$ ... **zahteva** j-te trgovine
- $cost(i, j)$ ... **cena** prevoza enega izdelka iz i-tega skladišča v j-to trgovino
- $x_{i,j}$ ... **količina prevoženega izdelka** iz i-tega skladišča v j-to trgovino (pri konkretnem prevozu)

Glede na zgornje oznake lahko problem linearnega prevoza zapišemo kot:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{k} cost(i, j) \cdot x_{i,j}$$

Pri reševanju problema moramo upoštevati še naslednje pogoje:

- $\sum_{j=1}^{k} x_{i,j} \leq sour(i);\ i = 1, 2, \ldots, n$
  (iz i-tega skladišča ne moremo odpeljati večje kotličine, kot je zaloga skladišča)
- $\sum_{i=1}^{n} x_{i,j} \geq dest(j);\ j = 1, 2, \ldots, k$
  (v j-to trgovino moramo pripeljati najmanj zahtevano količino trgovine)
- $x_{i,j} \geq 0;\ i = 1, 2, \ldots n$ in $j = 1, 2, \ldots, k$
  (prevažamo vedno iz skladišča v trgovino in ne obratno)

2.1 Uravnotežen problem linearnega prevoza

Če velja $\sum_{i=1}^{k} sour(i) = \sum_{j=1}^{n}$, pravimo da je problem linearnega prevoza **uravnotežen**.

Posledično veljata tudi naslednji enakosti:

- $\sum_{j=1}^{k} x_{i,j} = sour(i);\ i = 1, 2, \ldots, n$
- $\sum_{i=1}^{n} x_{i,j} = dest(j);\ j = 1, 2, \ldots, k$

Znotraj tega projekta bomo obravnavali samo uravnotežene problem linearnega prevoza.

2.2 Celoštevilski uravnotežen problem linearnega prevoza

Če za vsako skladišče i velja, da je $sour(i)$ naravno število in za vsako trgovino j velja, da je $dest(j)$ naravno število v uravnoteženem problemu linearnega prevoza, potem je tudi optimalna rešitev uravnoteženega problema linearnega prevoza celoštevilska (to pomeni, da so vrednosti $x_{i,j}$ naravna števila. Velja tudi, da je število pozitivnih (torej različnih od 0) vrednosti $x_{i,j}$ največ $k + n - 1$.

3 Klasični genetski algoritem

Da je algoritem klasični pomeni, da so kromosomi pradstavljeni z dvojiškim zapisom. To pomeni, da so možne rešitve predstavljene z vektorjem

$$(v_1, v_2, v_3, \ldots, v_p);\ p = n \cdot k,$$

pri čemer je komponenta vektorja v_i dvojišči vektor

$$(w_0^i, w_1^i, \ldots, w_s^i),$$

ki predstavlja vrednost $x_{j,m}$ za $j = \lfloor \frac{i-1}{k+1} \rfloor$ in $m = (i-1) mod (k+1)$.

Za vsako dopustno rešitev tega problema mora veljati:

- $v_q \geq 0$ za $q = 1, 2, \ldots, k \cdot n$,
- $\sum_{i=c\cdot k+1}^{c\cdot k+k} v_i = sour(c+1)$ za $c = 0, 1, \ldots, n-1$,
- $\sum_{j=m,\ \text{korak}\ k}^{k\cdot n} v_j = dest(m)$ za $m = 1, 2, \ldots, k$.

3.1 Evaluation function

Evaluacijska funkcija tega problema je skupna cena prevoza, pri čemer so izpolnjeni vsi zgornji pogoji.

$$eval((v_1, v_2, v_3, \ldots, v_p)) = \sum_{i=1}^{p} v_i \cdot cost(j, m),$$

kjer je $j = \lfloor \frac{i-1}{k+1} \rfloor$ in $m = (i-1) mod (k+1)$.

3.2 Genetic operators

Mutacija in križanje za klasični genetični algoritem naletita na mnoge probleme, ki privedejo do ugotovitve, da ta implementacija ni najbolj primerna.

4 GEN1 - Izboljšava vektorske reprezentacije in njena implementacija

V tem poglavju bomo opisali implementacijo prvega genetskega algoritma, ki ga bomo označevali z oznako $GEN1$.

4.1 Inicializacija-inicialization

Če želimo dobiti začetno populacijo, moramo zgenerirati nekaj dopustnih rešitev, ki zadoščajo zgornjim pogojem. S funckijo *inicialization*.
Rešitve podajamo v obliki permitacije, kjer permutacija določa zaporedje izbire celice v matriki, po kateres se potem izračuna inicializacija rešitve.

4.2 Genetic operators

Za konstrukcijo nove generacije uporabimo eno izmed možnosti inverzije (obrata), mutacije ali križanja. Inverzija permutacijo, ki predstavlja rešitev samo obrne (torej izbiramo v obratnem vrstnem redu, mutacije zamenja dva elementa permutacije, križanje pa izbere del prvega starša (prve permutacije) ter preostali del zapolni z mankajočimi elementi v zaporedju, kot ga določa drugi starš (druga permutacija).

5 GEN-2

V tem algoritmu rešitve predstavljamo v obliki matrike, kar se zdi bolj naravno glede na problem.
Za vsako dopustno rešitev tega problema mora veljati:

- $v_{i,j} \geq 0$ za $i = 1, 2, \ldots, k;\ j = 1, 2, \ldots, n,$
- $\sum_{i=1}^{k} v_{i,j} = dest(j)$ za $j = 1, \ldots, n,$

- $\sum_{j=1}^{n} v_{i,j} = sour(i)$ za $i = 1, \ldots, k$.

5.1 Evaluation function

$$eval(v_{i,j}) = \sum_{i=1}^{k} \sum_{j=1}^{n} v_{i,j} \cdot cost(i,j)$$

5.2 Genetic operators

5.2.1 Mutation