Neke inačice problema usmjeravanja vozila

Nekić, Lovro

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:493921

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Lovro Nekić

NEKE INAČICE PROBLEMA USMJERAVANJA VOZILA

Diplomski rad

Zagreb, srpanj, 2024.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Općeniti problem usmjeravanja vozila *(engl. Vehicle routing problem, VRP)* možemo zadati na sljedeći način: za dani broj vozila za prijevoz i zahtjeve za dostavu (ili transport), potrebno je odrediti rute vozila tako da zadovoljimo sve zahtjeve uz minimalni trošak. Odnosno, za svako vozilo želimo odrediti rutu (tj. koje će dostave izvršiti u kojem poretku) tako da su svi zahtjevi ispunjeni i da je trošak minimalan. Kao ulazne podatke stoga očekujemo informacije o vozilima (broj, početna točka), točke dostave te pripadne troškove prijevoza između svake dvije točke, te eventualno zaradu generiranu svakom dostavom, iz čega formiramo povezan (ili čak jako povezan) graf kao vizualnu ilustraciju problema, i nadalje, tražimo najbolje rješenje, odnosno skup ruta koji minimizira funkciju troška za zadani problem.

VRP je, naravno, poopćenje problema trgovačkog putnika *(engl. Travelling salesman problem, TSP)* za koji znamo da je NP-težak problem, jer za n gradova, postoji $(n-1)!$ mogućih ruta, npr. već za $n = 21$ imamo 5.11×10^{19} mogućnosti, pa je kombinatorno traženje rješenja neefikasno i zahtijeva optimizacijske metode.

Problem usmjeravanja vozila ima široku primjenu u industriji, naročito u logistici, npr. cisterne dovoze gorivo na benzinske crpke, kooperant s nekim brojem kamiona različitih kapaciteta dovozi građevinski materijal na gradilište, helikopteri različitog dometa i kapaciteta dovoze potrepštine na zabačene otoke, itd. Zbog kompleksnosti i obujma samog problema, traženje optimalnog rješenja preko matematičke optimizacije, pohlepnih algoritama ili kombinatorno mogu biti skupa, pa se VRP uglavnom rješava heuristikama. Praksom se pokazalo da industrijski algoritmi za rješavanje VRP mogu smanjiti trošak za $5\% - 30\%$[1].

[1] Ova tvrdnja se nalazi u [6], dok se većina ovog rada temelji na [3, str. 291 - 346]

Poglavlje 1

Problem trgovačkog putnika

1.1 Zapis problema unutar teorije grafova

Promotrimo trgovačkog putnika koji kreće od svoje kuće, i tokom radnog dana mora obići određen broj gradova te se na kraju radnog dana vratiti kući. Neka je, uključujući i grad iz kojeg kreće, $n \in \mathbb{N}$ ukupan broj gradova koje trgovački putnik mora obići. Logično je zahtijevati da iz svakog grada može doći do svakog drugog, pa svaka dva grada imaju cestovnu ili zračnu udaljenost između sebe (ovisno o tome kako trgovački putnik putuje). Naravno, udaljenost od grada A do grada B je identična udaljenosti od grada B do grada A. Dakle, ako gradove zamišljamo kao vrhove grafa, ceste između gradova kao bridove, te udaljenosti kao težine tih bridova, dobivamo neusmjeren (jako) povezan težinski graf.

Napomena 1.1.1. *Ovako dobiveni graf ne mora nužno biti neusmjeren (jako) povezan. Zaista, istaknimo neke moguće situacije:*

- *Ako zamislimo da je neka jednosmjerna dionica ceste zatvorena zbog radova, tada iz grada A možemo doći u grad B, ali iz grada B ne možemo doći u grad A. Također, možemo imati da je put iz grada A u grad B kraći od puta iz grada B u grad A, npr. zbog radova na cesti. U ovom slučaju imamo usmjeren povezan graf (ne nužno jako povezan), gdje možemo imati više bridova između dva ista vrha (različite ceste) koji imaju različite težine*

- *Ako zamislimo da trgovački putnik putuje zrakom, ne moramo nužno imati letove iz grada A u sve preostale gradove, tj. možda ćemo morati presjesti na drugi let u gradu C, prije nego sletimo u grad B. Tada imamo neusmjeren povezan graf. Ako baš želimo jako povezan, tj. potpun graf, dodajemo nepostojeće bridove koji nam trebaju za potpunost, na način da na njih stavimo ogromne težine (tako da trgovački putnik nikada neće ići tim bridovima)*

Pretpostavimo da je naš graf neusmjeren jako povezan težinski graf, odnosno, neusmjeren potpun težinski graf. Neka trgovački putnik kreće iz nekog vrha m. Želimo pronaći Hamiltonov ciklus najmanje težine uz uvjet polaska iz vrha m. Hamiltonov ciklus je šetnja po grafu koja posjećuje sve vrhove grafa točno jednom, a kreće i završava u istom vrhu. Težina Hamiltonovog ciklusa je zbroj težina svih bridova po kojima je šetnja tekla. Naravno, Hamiltonov ciklus se sastoji od točno onoliko bridova koliko imamo i vrhova u grafu.

Najintuitivniji način rješavanja ovog problema je dakako *brute-forcing*, odnosno, provjeravanje težina svih Hamiltonovih ciklusa te zatim određivanje najmanjeg. No, to je NP-težak problem jer za n gradova razlikujemo $(n-1)!$ Hamiltonovih ciklusa, odnosno, ako ne računamo inverze ciklusa, imamo $\frac{(n-1)!}{2}$ Hamiltonovih ciklusa za provjeriti. Dakle, kombinatornim rješavanjem nećemo doći daleko za velike n, no nemamo neki znatno bolji algoritam koji daje egzaktno rješenje.

No, heuristički možemo dobiti Hamiltonov ciklus koji je najviše $\frac{3}{2}$ teži od optimalnog na sljedeći način:

1. Neka je ω težinska funkcija nad bridovima tako da vrijedi:

$$\forall x, y, z \in V, \quad \omega(\{x, z\}) \leq \omega(\{x, y\}) + \omega(\{y, z\})$$

2. Iz početnog grafa pronalazimo minimalno razapinjuće stablo S

3. Vrhove neparnog stupnja iz S (ima ih parno, $2k$, jer iz leme o rukovanju znamo da je u svakom grafu suma stupnjeva vrhova paran broj, pa onda vrhova neparnog stupnja mora biti parno) sparimo sa k bridova M na način da je M najmanje moguće težine

4. Graf $S + M$ (svi vrhovi parnog stupnja) ima Eulerovu turu, pa od Eulerove ture konstruiramo Hamiltonov ciklus na način da izbacimo bridove koji nas dovode do vrhova koje smo već posjetili, te na tim mjestima biramo brid koji nas dovodi do sljedećeg vrha kojeg nismo posjetili

Dokaz ove tvrdnje može se pronaći u [1, str. 81, 82].

1.2 Problem trgovačkog putnika kao optimizacijska zadaća

Uvodimo sljedeće oznake:

- Neka je N skup svih vrhova koje moramo posjetiti
- Neka je n kardinalnost skupa N, tj. $n = |N|$

- Neka je c_{ij} trošak prilikom putovanja od vrha i do vrha j
- Neka je

$$x_{ij} = \begin{cases} 1 & \text{ako trgovački putnik putuje od } i \text{ do } j,\ i \neq j \\ 0 & \text{inače} \end{cases}$$

Budući da želimo naći Hamiltonov ciklus, prirodno se postavlja nužni zahtjev da svaki vrh mora imati neki drugi vrh u koji može otići, te neki vrh iz kojeg je mogao doći. Naravno, ako imamo potpun graf, to je trivijalno ispunjeno, ali, kako smo već komentirali, to ne mora biti slučaj. Dodatno, tako svaki vrh posjećujemo samo jednom, za svaki vrh imamo točno jedan vrh iz kojeg je putnik došao, i točno jedan u kojeg odlazi. Time imamo sljedeću minimizacijsku zadaću:

$$\sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \to \min \tag{1.1}$$

$$\forall i \in N, \quad \sum_{\substack{j \in N \\ j \neq i}} x_{ij} = 1 \tag{1.2}$$

$$\forall j \in N, \quad \sum_{\substack{i \in N \\ i \neq j}} x_{ij} = 1 \tag{1.3}$$

$$x_{ij} \in \{0, 1\} \tag{1.4}$$

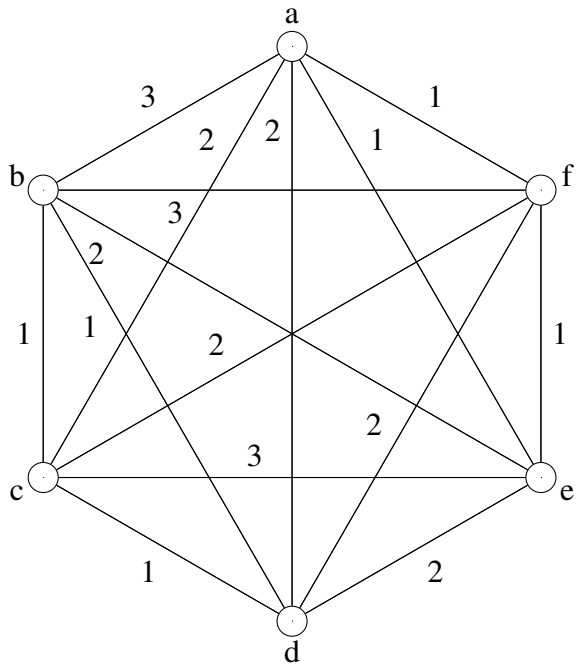
Pri čemu (1.1) predstavlja minimizaciju funkcije troška, (1.2) uvjet koji osigurava da svaki vrh ima točno jedan vrh u koji putnik odlazi, dok smo s (1.3) zadali da svaki vrh ima točno jedan vrh iz kojeg putnik dolazi.

Ovakva formulacija optimizacijske zadaće za problem trgovačkog putnika daje nužne ali ne i dovoljne uvjete, odnosno, optimalno rješenje ne mora dati Hamiltonov ciklus. Za ilustraciju, pogledajmo primjer potpunog težinskog grafa K_6 (vidi sliku 1.1).

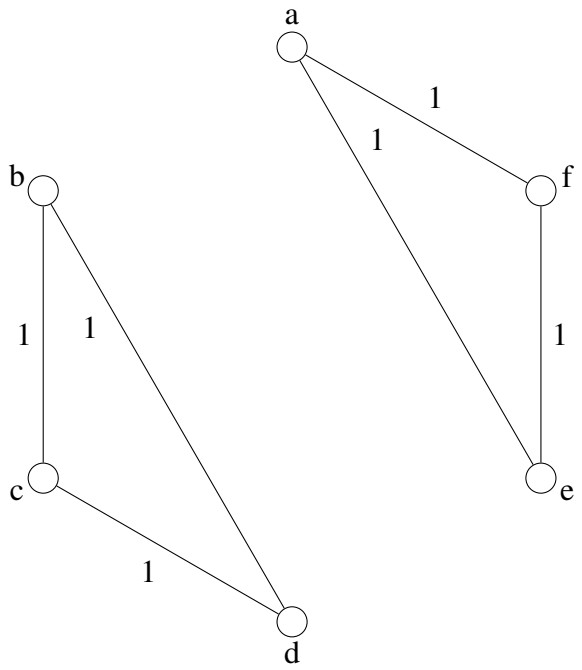
Ako uzmemo da je $x_{bc} = x_{cd} = x_{db} = x_{ef} = x_{fa} = x_{ae} = 1$, te svi ostali 0, dobivamo rješenje minimizacijske zadaće, jer iz svakog vrha i izlazi točno jedan brid, te u svaki vrh j ulazi točno jedan brid, odnosno, zadovoljeni su uvjeti (1.2) i (1.3). Uvjet (1.4) je trivijalno zadovoljen. Dodatno, funkcija cilja postiže optimalnu vrijednost 6, jer su pripadne težine c_{ij} za $x_{ij} \neq 0$ sve jednake 1. No, ovo rješenje ne daje Hamiltonov ciklus (vidi sliku 1.2) pa onda nije niti rješenje problema trgovačkog putnika.

Problem se, naravno, javlja jer iako imamo lokalne Hamiltonove cikluse, nemamo globalni Hamiltonov ciklus, odnosno, imamo Hamiltonove cikluse na podgrafovima. Pojave podciklusa možemo eliminirati na način da ne dozvolimo da se „krug zatvori”, odnosno, u našem konkretnom primjeru, imamo podciklus $b - c - d - b$, dakle, možemo dodati uvjet da u toj konkretnoj konfiguraciji imamo najviše 2 brida, tj. da vrijedi:

$$x_{bc} + x_{cb} + x_{cd} + x_{dc} + x_{db} + x_{bd} \leq 2$$



Slika 1.1: Primjer težinskog K_6 grafa



Slika 1.2: Rješenje koje zadovoljava minimizacijsku zadaću

jer u tom slučaju ne možemo dobiti podciklus na vrhovima b, c, d. Analogno bi napravili i za podciklus $e - f - a - e$. No, time nismo osigurali da nam se opet ne stvore podciklusi različiti od ova dva. Moramo dakle osigurati da se za sve moguće konfiguracije vrhova „krug ne zatvori", odnosno, za svaki podskup vrhova, moramo onemogućiti stvaranje podciklusa. To nam daje sljedeći uvjet:

$$\forall S \subset N, S \neq \emptyset, \quad \sum_{i \in S} \sum_{\substack{j \in S \\ i \neq j}} x_{ij} \leq |S| - 1 \tag{1.5}$$

Uvjeti (1.1)-(1.5) nam daju Dantzig-Fulkerson-Johnsonovu (DFJ) formulaciju problema trgovačkog putnika u terminima optimizacije. Uvjet (1.5) je zapravo $2^n - 2$ zasebnih uvjeta (jer gledamo sve podskupove od N osim praznog skupa i cijelog N, dakle imamo $2^n - 2$ mogućnosti jer N sadrži n elemenata), dok s (1.2) i (1.3) prepoznajemo $2n$ uvjeta tipa jednakosti. U zbroju dakle imamo $2^n + 2n - 2$ uvjeta na parametre x_{ij}, a to je bolje od $\frac{(n-1)!}{2}$ za velike n. Uz pretpostavku da je vrijeme izvršavanja isto za kombinatorno i optimizacijsko rješavanje, tj. da se jedan Hamiltonov ciklus na kombinatorni način pronađe u istom vremenu kao i uvažavanje jednog od ovih uvjeta, već za $n = 7$ je optimizacijski pristup bolji, jer je $(\frac{(n-1)!}{2} - 2^n - 2n + 2)|_{n=7} = 220$.

Još jedan pristup za eliminaciju podciklusa je Miller-Tucker-Zemlinova (MTZ) formulacija koja koristi tzv. veliki-M uvjet za određivanje prioriteta vrhova, odnosno red u kojem se vrhovi posjećuju. Uz iste oznake kao i prije, dodatno uvodimo i novu:

$$u_i \geq 0, \text{označava red vrha } i \neq 1 \text{ u nizu vrhova koji se posjećuju}$$

Sada imamo minimizacijski problem:

$$\sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \to \min \tag{1.6}$$

$$\forall i \in N, \quad \sum_{\substack{j \in N \\ j \neq i}} x_{ij} = 1 \tag{1.7}$$

$$\forall j \in N, \quad \sum_{\substack{i \in N \\ i \neq j}} x_{ij} = 1 \tag{1.8}$$

$$\forall (i, j) \in N \times N, i \neq 1, j \neq 1, i \neq j, \quad u_i - u_j + n x_{ij} \leq n - 1 \tag{1.9}$$

$$x_{ij} \in \{0, 1\}, u_i \geq 0 \tag{1.10}$$

Uvjeti (1.6)-(1.10) daju Miller-Tucker-Zemlinovu formulaciju problema trgovačkog putnika, gdje je uvjet (1.9) veliki-M uvjet. Promotrimo pobliže veliki-M uvjet:

- Ako je $x_{ij} = 1$, tj. trgovački putnik ide iz vrha i u vrh j, tada uvrštavanjem u (1.9) dobivamo $u_i + 1 \leq u_j$

- Ako je $x_{ij} = 0$, tada imamo $u_i - u_j \leq n - 1$, a to vrijedi uvijek

Drugim riječima, ako iz i idemo u j, red od vrha j je za 1 veći od reda od vrha i, tj. ako varijable u_i zamišljamo kao svojevrsni *queue*, u_j je sljedeći na redu nakon u_i ako je $x_{ij} = 1$ (stoga u ovom slučaju zapravo imamo strogu jednakost, tj. $u_i + 1 = u_j$). Obratno, ako je $x_{ij} = 0$, tada je razlika u redu od vrha i i vrha j manja ili jednaka $n - 1$, odnosno, ako ih zamišljamo kao *queue*, vrh j je na redu nakon vrha i za najviše $n - 1$ koraka, a to je naravno istina, jer ako nije prva sljedeća na redu, može biti druga sljedeća, ako ne treća, itd. do najviše $n - 1$, jer imamo n vrhova, a na kraju se moramo vratiti u početni.

Prednost Miller-Tucker-Zemlinove formulacije je polinomijalna složenost jer imamo $2n$ uvjeta iz (1.7) i (1.8), te $(n - 1)(n - 2)$ uvjeta iz (1.9) pa je to sveukupno $n^2 - n + 2$ uvjeta. Ako opet pretpostavimo isto vremensko izvršavanje operacija, već za $n = 6$ je ova formulacija uspješnija jer $(\frac{(n-1)!}{2} - n^2 + n - 2)|_{n=6} = 28$.

Za kraj ovog odjeljka, spomenimo još i formulaciju mrežnog toka *(engl. Network flow based formulation)*. U ovoj fomulaciji, podcikluse eliminiramo na način da imamo konstantni tok kroz vrhove Hamiltonovog ciklusa, odnosno, u početni vrh „upumpamo" $n - 1$ jedinica toka, te u svakom vrhu „potrošimo" jednu jedinicu toka, uz uvjet da je tok dozvoljen jedino kroz vrhove Hamiltonovog ciklusa (vidi sliku 1.3).

Formalno, moramo imati uvjet očuvanja toka koji svakom vrhu dopušta samo jednu jedinicu toka, a zatim velikim-M uvjetom osiguravamo da tok ne prolazi bridovima koji nisu u Hamiltonovom ciklusu. Stoga nam za ovu formulaciju treba još i sljedeća varijabla: $y_{ij} \geq 0$, označava tok kroz brid $(i, j) \in E, i \neq j$. Minimizacijski problem u formulaciji mrežnog toka sada glasi:

$$\sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \to \min \tag{1.11}$$

$$\forall i \in N, \quad \sum_{\substack{j \in N \\ j \neq i}} x_{ij} = 1 \tag{1.12}$$

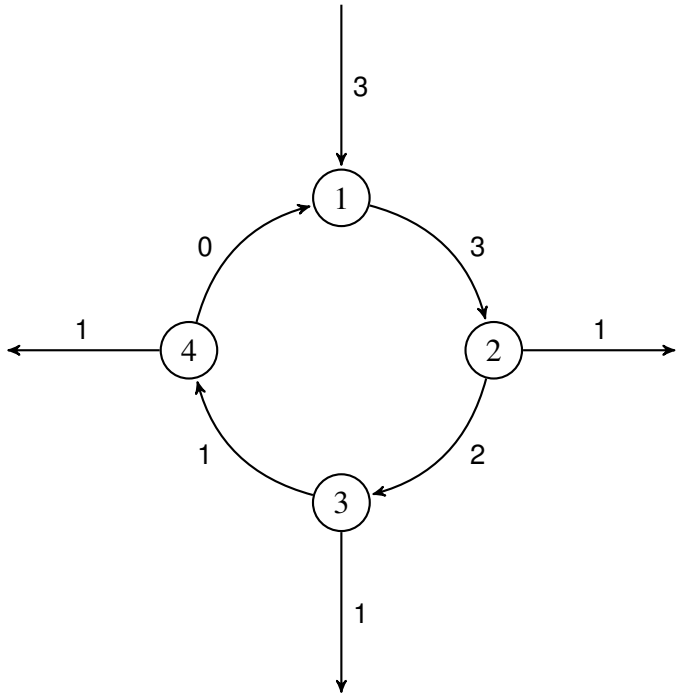
$$\forall j \in N, \quad \sum_{\substack{i \in N \\ i \neq j}} x_{ij} = 1 \tag{1.13}$$

$$\sum_{\substack{j \in N \\ j \neq 1}} y_{1j} = n - 1 \tag{1.14}$$

$$\forall j \in N, j \neq 1, \quad \sum_{\substack{i \in N \\ i \neq j}} y_{ij} - \sum_{\substack{k \in N \\ k \neq j}} y_{jk} = 1 \tag{1.15}$$

$$\forall (i, j) \in N \times N, i \neq j, \quad y_{ij} \leq (n - 1) x_{ij} \tag{1.16}$$

$$x_{ij} \in \{0, 1\}, y_{ij} \geq 0 \tag{1.17}$$



Slika 1.3: Primjer mrežnog toka

Uvjeti (1.11)-(1.13) su identični kao i u prethodnim formulacijama. (1.14) je uvjet koji „upumpava" $n-1$ jedinicu toka u početni vrh grafa, dok uvjet (1.15) osigurava očuvanje toka u ostalim vrhovima, tj. razlika količine pritoka u vrh i količine odtoka iz vrha jednaka je 1. Uvjet (1.16) je veliki-M uvjet koji osigurava da tok prolazi isključivo kroz vrhove u Hamiltonovom ciklusu (odnosno strogo zadanim redosljedom kojim ciklus obilazi vrhove). Naime, ako je $x_{ij} = 0$ tada je i $y_{ij} = 0$ pa nema toka ako iz i ne prelazimo u j, a ako je $x_{ij} = 1$, tada je $y_{ij} \leq n-1$, dakle, ako iz i idemo u j, tada je količina toka najviše $n-1$, ovisno naravno u kojem se vrhu nalazimo.

Složenost ove formulacije je polinomijalna, jer imamo $2n-2$ uvjeta iz (1.12) i (1.13), n uvjeta iz (1.14) i (1.15), te $n(n-1)$ uvjeta iz (1.16), a to u zbroju daje n^2+2n uvjeta. Dakle, ova formulacija je po složenosti malo lošija od MTZ formulacije.

1.3 Problem trgovačkog putnika s vremenskim ograničenjima

Promotrimo jednu varijaciju problema trgovačkog putnika. Do sada smo se bazirali na klasični problem trgovačkog putnika, gdje smo samo uzimali u obzir udaljenosti među vrhovima, no, u stvarnosti nemamo samo prostorne nego i vremenske zahtjeve. Stoga je prirodnije da promatramo modificirani problem u kojem imamo i vremenske uvjete, npr. ako imamo kamion koji dostavlja robu na neke lokacije tokom dana, on ih mora dostaviti u nekom određenom periodu, jer svaka od lokacija ima svoje radno vrijeme, kao i sam vozač kamiona.

Formalno, neka je $G = (N, E)$ usmjereni graf, gdje svaki vrh $i \in N$ ima vremenski rok $[R_i, D_i]$, gdje je R_i *release time* odnosno najranije vrijeme kada možemo posjetiti vrh i, te D_i *deadline*, odnosno najkasnije vrijeme kada možemo posjetiti vrh i. Svaki brid $(i, j) \in E$ ima svoju težinu c_{ij} i vrijeme putovanja θ_{ij}. Ovaj se model bazira na Miller-Tucker-Zemlinovoj formulaciji, gdje varijable za određivanje reda u_i zamjenjujemo varijablama za određivanje termina, u kontekstu vremena. Koristimo sljedeću notaciju:

- Neka je N skup svih vrhova koje moramo posjetiti
- Neka je E skup svih dostupnih bridova kojima možemo prolaziti
- Neka je n kardinalnost skupa N, tj. $n = |N|$
- Neka je c_{ij} trošak prilikom putovanja od vrha i do vrha j
- Neka je θ_{ij} vrijeme potrebno da bismo došli od vrha i do vrha j
- Neka je R_i najranije vrijeme kada možemo posjetiti vrh i
- Neka je D_i najkasnije vrijeme kada možemo posjetiti vrh j
- Neka je M_{ij} neki veliki pozitivni broj pridružen bridu $(i, j) \in E$
- Neka je
$$x_{ij} = \begin{cases} 1 & \text{ako trgovački putnik putuje od } i \text{ do } j,\ i \neq j \\ 0 & \text{inače} \end{cases}$$
- Neka je $s_i \geq 0$ vrijeme za posjetiti vrh i

Sada imamo minimizacijski problem:

$$\sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \to \min \tag{1.18}$$

$$\forall i \in N, \quad \sum_{\substack{j \in N \\ j \neq i}} x_{ij} = 1 \tag{1.19}$$

$$\forall j \in N, \quad \sum_{\substack{i \in N \\ i \neq j}} x_{ij} = 1 \tag{1.20}$$

$$s_1 = 0 \tag{1.21}$$

$$\forall (i, j) \in N \times N, j \neq 1, i \neq j, \quad s_i + \theta_{ij} - (1 - x_{ij})M_{ij} \leq s_j \tag{1.22}$$

$$\forall i \in N, \quad R_i \leq s_i \leq D_i \tag{1.23}$$

$$x_{ij} \in \{0, 1\}, s_i \geq 0 \tag{1.24}$$

Uvjeti (1.18)-(1.20) su standardni i identični kao i u Miller-Tucker-Zemlinovoj formulaciji. Uvjet (1.21) nam određuje da je vrijeme u početnoj točki, tj. vrhu 0. Uvjet (1.23) modelira vremenska ograničenja koja smo zahtijevali, odnosno da posjet vrhu i bude između vremena R_i i D_i. Uvjet (1.22) je modificirani veliki-M uvjet za kojeg imamo:

- ako je $x_{ij} = 1$ tada iz vrha i idemo u vrh j, pa uvrštavanjem imamo $s_i + \theta_{ij} \leq s_j$
- ako je $x_{ij} = 0$ imamo $s_i + \theta_{ij} - M_{ij} \leq s_j$

Odnosno, ako iz vrha i idemo u vrh j, tada vrijeme za posjetiti vrh j ne smije biti veće od vremena za posjetiti vrh i uvećanog za vrijeme putovanja od vrha i do vrha j. Obratno, ako je $x_{ij} = 0$, nejednakost $s_i + \theta_{ij} - M_{ij} \leq s_j$ je očito uvijek zadovoljena za dovoljno velike M_{ij}.

Složenost ove minimizacije je polinomijalna, jer imamo $2n$ uvjeta iz (1.19) i (1.20), jedan uvjet iz (1.21) koji možemo zanemariti jer ga prilikom implementacije možemo uvažiti neovisno o ostalima, zatim $n(n-2)$ uvjeta iz (1.22) te $2n$ uvjeta iz (1.23). U zbroju imamo $n^2 + 2n$ uvjeta, dakle polinomijalnu složenost, tj. $O(n^2)$.

1.4 Crno-bijeli problem trgovačkog putnika

Za kraj ovog poglavlja, promotrimo tzv. crno-bijeli problem trgovačkog putnika *(engl. Black-and-White TSP, BWTSP)*. Često se u primjeni javlja situacija da imamo ograničeno kretanje, odnosno postoje neke restrikcije u količini i mogućnosti gibanja, npr. vozilo ima određen domet, nakon čega mora pristati i ponovno napuniti svoj spremnik goriva da bi mogao putovati dalje. U ovakvom tipu problema razlikujemo dvije vrste vrhova: *crne*, gdje se vozilo ili putnik obnavljaju i *bijele*, gdje vozilo ili putnik pružaju usluge. Stoga ovakav tip problema nazivamo crno-bijeli problem trgovačkog putnika. Naravno, u praksi možemo proći samo određen broj lokacija između dva obnavljanja, pa je stoga

prirodan zahtjev da imamo određen maksimalan broj bijelih vrhova između dva crna, kao i maksimalnu dozvoljenu udaljenost dva crna vrha.

Formalno, za graf $G = (N, E)$, gdje se N sastoji od skupa crnih vrhova C i bijelih vrhova B, za najmanje dva crna vrha, tražimo najkraći Hamiltonov ciklus uz sljedeće uvjete:

1. Uvjet kardinalnosti: Broj bijelih vrhova između dva uzastopna crna je najviše Q
2. Uvjet duljine: Udaljenost svaka dva uzastopna crna vrha je najviše L

Za $Q = L = \infty$, BWTSP se svodi na klasični TSP. Kao i kod problema s vremenskim ograničenjem, temeljimo model na Miller-Tucker-Zemlinovoj formulaciji s modificiranim velikim-M uvjetom. Koristimo notaciju kao i ranije, uz uvođenje nekih novih oznaka:

- Neka je N skup svih vrhova koje moramo posjetiti, $N = B \cup C$
- Neka je B skup svih bijelih vrhova, a C skup svih crnih vrhova
- Neka je E skup svih dostupnih bridova kojima možemo prolaziti
- Neka je n kardinalnost skupa N, tj. $n = |N|$
- Neka je Q maksimalni broj uzastopnih bijelih vrhova između dva crna, a L maksimalna udaljenost dva uzastopna crna vrha
- Neka je c_{ij} udaljenost od vrha i do vrha j, $c^*_{ij} = min\{c_{ij} : i \in B, j \in C\}$ najmanja udaljenost između svih bijelih i svih crnih vrhova, $c^{**}_{ij} = min\{c_{ij} : i \in C, j \in B\}$ najmanja udaljenost između svih crnih i svih bijelih vrhova, $c^{***}_{ij} = min\{c_{ij} : i, j \in B\}$ najmanja udaljenost između bijelih vrhova
- Neka je M neki veliki pozitivni broj, npr. $M = L - c^*_{ij} - c^{**}_{ij} + c^{***}_{ij}$ (minimalna udaljenost između 2 crna vrha t.d. je između njih barem 1 bijeli vrh je $c^*_{ij} + c^{**}_{ij}$, a taj broj je manji ili jednak L jer bi u suprotnom crni vrhovi bili susjedni, pa je ovakav M doista pozitivan)
- Neka je
$$x_{ij} = \begin{cases} 1 & \text{ako trgovački putnik putuje od } i \text{ do } j, i \neq j \\ 0 & \text{inače} \end{cases}$$
- Neka je $u_i \geq 0$ varijabla odluke iz Miller-Tucker-Zemlinove formulacije za eliminaciju podciklusa
- Neka je $t_i \geq 0$ brojač posjećenih bijelih vrhova nakon vrha i od zadnjeg posjećenog crnog vrha

- Neka je $l_i \geq 0$ brojač udaljenosti od vrha i do zadnjeg posjećenog crnog vrha

Iz ovoga imamo sljedeći minimizacijski problem:

$$\sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \to \min \tag{1.25}$$

$$\forall i \in N, \quad \sum_{\substack{j \in N \\ j \neq i}} x_{ij} = 1 \tag{1.26}$$

$$\forall j \in N, \quad \sum_{\substack{i \in N \\ i \neq j}} x_{ij} = 1 \tag{1.27}$$

$$\forall (i, j) \in E, i, j \neq 1, \quad u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n - 2 \tag{1.28}$$

$$\forall i \in N, i \neq 1, \quad 1 \leq u_i \leq n - 1 \tag{1.29}$$

$$\forall i, j \in B, (i, j) \in E, \quad t_i - t_j + Qx_{ij} + (Q-2)x_{ji} \leq Q - 1 \tag{1.30}$$

$$\forall i \in B, \quad 1 \leq t_i \leq Q \tag{1.31}$$

$$\forall i, j \in B, (i, j) \in E, \quad l_i - l_j + Mx_{ij} + (M - c_{ij} - c_{ji})x_{ji} \leq M - c_{ij} \tag{1.32}$$

$$\forall i \in B, \quad \sum_{\substack{j \in C \\ (j,i) \in E}} c_{ij} x_{ji} \leq l_i \tag{1.33}$$

$$\forall i \in B, \quad l_i + \sum_{\substack{j \in C \\ (i,j) \in E}} c_{ij} x_{ij} \leq L \tag{1.34}$$

$$x_{ij} \in \{0, 1\}, u_i, t_i, l_i \geq 0 \tag{1.35}$$

Uvjeti (1.25)-(1.27) su standardni za TSP, uvjeti (1.28) i (1.29) su ekvivalent uvjeta za eliminaciju podciklusa iz MTZ formulacije. U (1.30) uspostavljamo vezu između t_i i t_j, odnosno ovisno o tome je li $x_{ij} = 1$ ili $x_{ji} = 1$ u vrh j se dolazi nakon vrha i, odnosno u vrh i se dolazi nakon vrha j. Uvjet (1.31) ograničava broj posjećenih bijelih vrhova između dva bijela na maksimalno Q. (1.32) i (1.33) su analogoni (1.30) i (1.31) za varijable l_i i l_j. (1.34) je uvjet duljine. Složenost je polinomijalna s najviše $3n^2 + 6n$ uvjeta.

Poglavlje 2

Problem usmjeravanja vozila

2.1 Općenita formulacija problema usmjeravanja vozila

Problem usmjeravanja vozila možemo zamišljati kao proširenje problema trgovačkog putnika, odnosno obratno, problem trgovačkog putnika možemo smatrati kao poseban slučaj problema usmjeravanja vozila, kada imamo samo jedno vozilo. Problem usmjeravanja vozila je dakle, općeniti slučaj trgovačkog putnika kada na raspolaganju imamo neki pozitivan broj vozila koja moraju obići neke lokacije, naravno, s ciljem minimizacije troška. Sva vozila imaju ograničen kapacitet, odnosno, mogu prevoziti samo određenu količinu tereta, stoga se ovakav problem naziva i kapacitirani problem usmjeravanja vozila *(engl. Capacitated Vehicle Routing Problem, CVRP).*

Formalno, imamo skup vozila $K = \{1, 2, \ldots, |K|\}$ koji vrše dostave na skup lokacija $N = \{1, 2, \ldots, n\}$. Neka sva vozila imaju isti kapacitet Q, te ona kreću iz centralnog skladišta (kojeg označavamo s 0) u koji se moraju i vratiti. U svakom vrhu $i \in N$ imamo zahtjev q_i, tj. koliko robe treba dostaviti na lokaciju i. Između svaka dva vrha i i j imamo trošak c_{ij}. Želimo pronaći $|K|$ Hamiltonovih ciklusa tako da zadovoljimo zahtjeve u svih n vrhova (na način da svaki posjetimo točno jednom) i da je pritom trošak minimalan. Uz ovakve oznake, definiramo još dodatno i:

- Neka je $\delta^+(S) = \{(i, j) \in E : i \in S, j \notin S\}$, $S \subseteq N$, tj. skup svih bridova koji izlaze iz skupa S
- Neka je $\delta^-(i) = \{j : (j, i) \in E\}$, te $\delta^+(i) = \{j : (i, j) \in E\}$, tj. skup svih vrhova čiji bridovi ulaze odnosno izlaze iz vrha i
- Neka je $r(S)$ najmanji broj ruta vozila koji mogu opskrbiti skup $S \subseteq N$

Napomena 2.1.1. *Prije postavljanja minimizacijske zadaće, uočimo da a priori nemamo način za određivanje vrijednosti $r(S)$ jer se u suštini radi o problemu pakiranja u pros-*

toru. Odnosno, ako imamo konačan skup I, tako da za svaki od njegovih elemenata $i \in I$ *imamo da je* $s(i) \in \mathbb{R}^+$ *neko mjerilo svakog od tih elemenata, te* $L \in \mathbb{N}, B \in \mathbb{R}^+$, B *veličina spremnika, postoji li particija skupa I na disjunktne skupove* $I_1, I_2, \ldots, I_L$ *tako da je* $\sum_{i \in I_k} s(i) \leq B, \forall k \in \{1, 2, \ldots, L\}$ *[5]. Konkretno u našem slučaju konačan skup N predstavlja I, gdje svaki vrh i ima pridruženu mjeru* q_i, Q *je veličina spremnika gdje tražimo najmanju takvu particiju, odnosno, najmanju vrijednost L. No, to je NP-težak optimizacijski problem, pa stoga koristimo doljnju ogradu* $\left\lceil \frac{q(S)}{Q} \right\rceil$, *gdje je* $q(S)$ *ukupno zahtjevanje skupa* S, *tj.* $q(S) = \sum_{i \in S} q_i$.

Napomena 2.1.2. *Broj vozila* $|K|$ *ovdje nije predmet optimizacije, iako to općenito ne mora biti tako (vidi odjeljak 2.2). Također, uočimo da uvjet da posjetimo svaki vrh točno jednom osigurava konstrukciju* $|K|$ *Hamiltonovih ciklusa, no to nužno ne mora biti tako. Pogledajmo sljedeći primjer za ilustraciju: neka je* $Q = 15$ *kapacitet svih vozila,* $|K| = 3$, *dakle imamo 3 vozila na raspolaganju, te* $q_1, q_2, q_3 = 10$ *potraživanja vrhova* $1, 2, 3$ *i krećemo i završavamo u vrhu* 0. *Ako imamo uvjet da se svaki vrh posjećuje točno jednom, imamo slučaj kao na lijevoj slici. No, ako dopustimo da vrhove možemo posjećivati više puta, tada potraživanja svih vrhova možemo zadovoljiti sa 2 vozila kao na desnoj slici. Tada se konstruiraju 2 Hamiltonova ciklusa, iako je* $|K| = 3$, *dok se uz restrikciju o jedinstvenosti posjeta doista konstruira* $|K|$ *Hamiltonovih ciklusa.*

Jedinstvenost posjeta vrhovima Nejedinstvenost posjeta vrhovima

Slika 2.1: Ilustracija uvjeta o jedinstvenosti posjeta vrhova

Sada imamo minimizacijski problem:

$$\sum_{i\in N}\sum_{j\in N} c_{ij}x_{ij} \to \min \tag{2.1}$$

$$\forall i \in N, \quad \sum_{j\in\delta^+(i)} x_{ij} = 1 \tag{2.2}$$

$$\forall j \in N, \quad \sum_{i\in\delta^-(j)} x_{ij} = 1 \tag{2.3}$$

$$\forall i \in N, \quad \sum_{j\in\delta^+(0)} x_{0j} = |K| \tag{2.4}$$

$$\forall S \subseteq N, S \neq \emptyset, \quad \sum_{(i,j)\in\delta^+(S)} x_{ij} \geq r(S) \tag{2.5}$$

$$\forall (i,j) \in E, \quad x_{ij} \in \{0,1\} \tag{2.6}$$

Uvjeti (2.1)-(2.3) su standardni, uvjet (2.4) osigurava konstrukciju točno $|K|$ ruta, dok je uvjet (2.5) eliminacija podciklusa. Ova formulacija se naziva kompaktna, no, kao i kod DFJ formulacije problema trgovačkog putnika, eliminacija podciklusa na ovaj način daje eksponencijalnu složenost.

Promotrimo stoga MTZ formulaciju problema usmjeravanja vozila. Kao i prije, imamo veliki-M uvjet kojeg ćemo modificirati tako da zadovoljimo uvjet eliminacije podciklusa i uvjet kapaciteta, odnosno da svaki vrh i zaprimi zahtjevanu količinu. Stoga uvodimo varijable $u_i \geq 0$ koje predstavljaju količinu dostavljenu vrhu i u zbroju (odnosno, varijabla u_i govori koliko je vozilo do vrha i, uključivo s vrhom i, dostavilo robe). Minimizacijski problem u MTZ formulaciji stoga izgleda ovako:

$$\sum_{i\in N}\sum_{j\in N} c_{ij}x_{ij} \to \min \tag{2.7}$$

$$\forall i \in N, \sum_{j\in\delta^+(i)} x_{ij} = 1 \tag{2.8}$$

$$\forall j \in N, \sum_{i\in\delta^-(j)} x_{ij} = 1 \tag{2.9}$$

$$\forall i \in N, \sum_{j\in\delta^+(0)} x_{0j} = |K| \tag{2.10}$$

$$\forall (i,j) \in E, u_i - u_j + Qx_{ij} \leq Q - q_j \tag{2.11}$$

$$\forall i \in N, q_i \leq u_i \leq Q \tag{2.12}$$

$$x_{ij} \in \{0,1\}, u_i \geq 0 \tag{2.13}$$

Uvjet (2.11) je veliki-M uvjet koji služi za eliminaciju podciklusa, odnosno, ako je $x_{ij} = 1$, tada je $u_i + q_j \leq u_j$ (dakle u zbroju, količina dostavljena u vrh j je najmanje količina dostavljena u vrh i plus zahtjev vrha j, tj. q_j), te ako je $x_{ij} = 0$, onda je $u_i - (u_j - q_j) \leq Q$, a to je uvijek zadovoljeno zbog (2.12). Kao i kod problema trgovačkog putnika, prednost MTZ formulacije nad DFJ je polinomijalna složenost MTZ formulacije.

Uočimo da u ove dvije formulacije imamo pretpostavku o identičnom kapacitetu svih vozila, no to nije realna pretpostavka, stoga bismo željeli formulaciju u kojoj možemo imati vozila različitih kapaciteta. Također je logično zahtijevati da kamioni manjeg kapaciteta imaju manju potrošnju goriva, pa trošak između dva vrha nije nužno isti za sva vozila. Ove zahtjeve ćemo riješiti pomoću formulacije mrežnog toka na način da ćemo tokove identificirati s vozilima te konstruirati onoliko tokova koliko imamo vozila. Za ovo će nam trebati malo drukčija notacija nego ranije, pa definiramo oznake:

- Neka je N skup svih lokacija na koje moramo izvršiti dostave, $N = \{1, \ldots, n\}$
- Neka je 0 točka početka, tj. točka odlaska iz centralnog skladišta, a $n + 1$ točka povratka u centralno skladište
- Neka je $V = N \cup \{0, n + 1\}$ skup svih vrhova, E skup svih bridova, te K skup svih vozila
- Neka je q_i zahtjev koliko robe treba dostaviti u vrh i, te Q_k kapacitet vozila k
- Neka je c_{ijk} trošak putovanja od vrha i do vrha j vozila k
- Neka je

$$x_{ijk} = \begin{cases} 1 & \text{ako vozilo } k \text{ putuje od } i \text{ do } j, \text{ za } k \in K, (i, j) \in E \\ 0 & \text{inače} \end{cases}$$

- Neka je $u_{ik} \geq 0$, brojač koliku je količinu dostavilo vozilo $k \in K$ do (uključivo) vrha $i \in N$

U ovako danim oznakama imamo minimizacijski problem problema usmjeravanja vozila u formulaciji mrežnog toka:

$$\sum_{k \in K} \sum_{(i,j) \in E} c_{ijk} x_{ijk} \to \min \tag{2.14}$$

$$\forall i \in N, \quad \sum_{k \in K} \sum_{\substack{j \in V \\ j \neq i}} x_{ijk} = 1 \tag{2.15}$$

$$\forall k \in K, \quad \sum_{\substack{j \in V \\ j \neq 0, n+1}} x_{0jk} = 1 \tag{2.16}$$

$$\forall k \in K, \forall h \in N, \quad \sum_{\substack{i \in V \\ i \neq h}} x_{ihk} - \sum_{\substack{j \in V \\ j \neq h}} x_{hjk} = 0 \tag{2.17}$$

$$\forall k \in K, \quad \sum_{\substack{i \in V \\ i \neq n+1}} x_{i,n+1,k} = 1 \tag{2.18}$$

$$\forall (i, j) \in V \times V, i \neq j, \forall k \in K, \quad u_{ik} - u_{jk} + x_{ijk} Q_k \leq Q_k - q_j \tag{2.19}$$

$$\forall i \in N, \forall k \in K, \quad q_i \leq u_{ik} \leq Q_k \tag{2.20}$$

$$x_{ijk} \in \{0, 1\}, u_{ik} \geq 0 \tag{2.21}$$

Minimizacijska funkcija (2.14) minimizira ukupni trošak za sva vozila. Uvjet (2.15) osigurava da je svaki vrh posjećen (uključujući i $n + 1$) iz nekog drugog vrha različitog od 0 i $n + 1$ (dakle, između odlaska i povratka u centralno spremište smo posjetili sve vrhove), dok uvjet (2.16) osigurava da svako vozilo iz centralnog skladišta odlazi u neki vrh (dakle stvara se $|K|$ ruta). Uvjet (2.17) je uvjet očuvanja toka za sve vrhove (osim skladišta) i sva vozila na način da vozilo koje ulazi u vrh izlazi iz istog vrha. (2.18) je uvjet povratka u skladište za svako vozilo. (2.19) je veliki-M uvjet za eliminaciju podciklusa za sva vozila (on je generalizacija (2.11) za vozila različitog kapaciteta). Prednost formulacije mrežnog toka nad MTZ formulacijom je mogućnost korištenja vozila različitih kapaciteta i različitih troškova putovanja, a to, naravno, bolje modelira stvarnu situaciju, pa će nam stoga formulacija mrežnog toka biti neka vrsta „odskočne daske” za daljnje, kompleksnije i speciliziranije formulacije.

2.2 Familija problema usmjeravanja vozila

U ovom odjeljku iznosimo neke od najvažnijih varijanti problema usmjeravanja vozila razdijeljene u nekoliko kategorija. Svaka od kategorija se sastoji od problema sličnog tipa, tj. klasifikacija se bazira na karakteristikama problema usmjeravanja vozila koje mijenjamo. Ovaj odjeljak se temelji na [2] i [4, str. 8 - 23].

Mrežne karakteristike

U klasičnim problemima usmjeravanja vozila smo imali točke, odnosno vrhove grafova koje smo morali obići *(problem usmjeravanja vrhova, engl. node routing problem, NRP)*, dok u ovim varijantama problema moramo obići sve bridove.

- *Problem usmjeravanja bridova*, npr. imamo graf koji predstavlja grad, te su bridovi ulice, a vrhovi raskrižja kojima možemo doći u druge ulice. Tada smetlarski kamioni ili ralice moraju proći sve ulice, pa imamo problem usmjeravanja bridova. Dostavu pošte također možemo uvrstiti pod ovu varijantu problema, jer iako poštar mora posjetiti svaku kuću zasebno, kuće/stanovi/zgrade su gusto zbijene u ulicama, pa je dostava pošte također problem obilaženja bridova

- *Općeniti problem usmjeravanja*, gdje imamo obilaženje i bridova i vrhova, npr. ako imamo autobus koji kupi putnike unutar grada i van grada, tada sela van grada možemo zamišljati kao vrhove, dok su autobusne stanice na ulicama unutar grada bridovi

Uočimo da u ovim varijantama problema bridovi predstavljaju zasebne ulice dok su u klasičnom VRP-u bridovi samo putevi između dvije točke, pa se mogu sastojati od velikog broja ulica i cesta (to svojstvo nazivamo *granularnost* i ono je najbitnija razlika između ovih varijanti problema).

Transportacijski uvjeti

U CVRP-u smo imali klasični zahtjev odlaska iz centralnog skladišta i dostave robe na određene lokacije. U ovim varijantama problema imamo drukčije uvjete transportacije.

- *Utovar i istovar*, gdje umjesto dostave kod klijenata imamo utovar i istovar robe kod klijenata. Ovisno o prirodi utovara i istovara imamo različite formulacije problema (npr. obavlja li se utovar i istovar istovremeno na svakoj lokaciji ili postoje neke lokacije koje su strogo određene za utovar i one koje su određene za istovar; vršimo li utovar i istovar na početku/kraju ili tokom rute). Primjer problema gdje imamo istovremen utovar i istovar na svakoj lokaciji tokom rute je dostava gajbi s pićem u restorane i kafiće i istovremeni odvoz praznih gajbi (praznih boca pića)

- *Jednostavni posjeti i zakazivanje vozila*, gdje nemamo kapacitete vozila jer nije potreban utovar ili istovar (*jednostavni posjeti*, npr. tehničar posjećuje kuće radi instaliravanja ili postavljanja odgovarajućih uređaja), ili je potreban točan raspored za obilaženje vrhova (*zakazivanje vozila*, npr. određivanje rasporeda vožnje autobusa i tramvaja u javnom prijevozu)

- *Alternativne i indirektne usluge*, npr. dostava paketa se može izvršiti na adresu stanovanja ili radnu adresu (tokom radnog vremena), pa dostavljač ima više opcija za mjesto dostave (u određenim uvjetima); alternativno, možemo imati lokaciju za dostavu paketa koja je „dovoljno blizu”, recimo kod susjeda ili u najbližu poštu (ako imamo slučaj zabačenih naselja)

- *Ponovljene dostave*, gdje klijenti zahtijevaju nekoliko dostava u nekom vremenskom periodu, npr. klijent zahtijeva tri dostave na tjednoj bazi s najviše dva dana razmaka između dvije dostave. Alternativno, možemo imati i slučaj kada ne želimo da nam ponestane robe, npr. trgovački lanac nadzire svaku od svojih poslovnica te ne želi da u bilo kojoj ponestane voća i povrća, pa nam se stoga svaki dan broj vrhova mijenja, odnosno rješavamo VRP na dnevnoj bazi jer nikada ne znamo koje će poslovnice ostati bez robe

- *Usmjeravanje s profitom*, gdje pretpostavljamo da jednostavno nemamo dovoljno vozila da bismo poslužili sve klijente, pa moramo odabrati samo neke na način da imamo najveći profit, koji računamo na način da troškove usmjeravanja i dostave oduzmemo od zarade od klijenata (za svakog klijenta je zarada i trošak drugačiji)

- *Dinamičko i stohastičko usmjeravanje*, gdje pod dinamičkim usmjeravanjem smatramo da su parametri i informacije znane tokom same rute, dok su kod stohastičkog usmjeravanja parametri nepoznati, ali je ta „nepoznatnost" dana kao vjerojatnostna distribucija, npr. ne znamo kakva je protočnost neke prometnice (gužve, prometne nesreće, nepropisno parkiranje), pa ne znamo koliki ćemo trošak imati putujući tom prometnicom

Lokalni uvjeti

Jedna od ključnih stvari pri rješavanju problema usmjeravanja vozila je izvedivost ruti. Kod ovih varijanti problema zadajemo lokalne uvjete na rute koje nam ograničavaju moguće rute na izvedive i neizvedive.

- *Kapacitet*, možemo imati klasičan uvjet na kapacitet kao kod CVRP-a, ali možemo ga i zadati u 2D ili 3D (kao pravokutnike/kvadre koje moramo na izvediv način posložiti u pravokutni/kvadarni spremnik). Također možemo imati i LIFO *(engl. Last-In-First-Out)* uvjet, odnosno redoslijed dostavljanja je obrnut redoslijedu ukrcavanja, dakle, pošiljku koju smo zadnju ukrcali moramo prvu dostaviti, i obratno

- *Duljina rute*, možemo zahtijevati da svako vozilo k ne smije proći određenu kilometražu L pa dobivamo uvjet $\sum_{(i,j)\in E} t_{ij}x_{ijk} \leq L$, za t_{ij} udaljenost između vrhova i i j. Analogno možemo imati i s vremenom, ali se onda mijenja značaj varijabli L i t_{ij}

- *Ponovno korištenje vozila*, ako imamo ograničen broj vozila na raspolaganju, možda ne možemo posjetiti sve vrhove na način da sva vozila koristimo samo jednom, stoga je logično zahtijevati da se neka vozila ponovno koriste (npr. vozila malog kapaciteta ili kratkog dometa), pa ako imamo p ruta s trajanjima $T_1, T_2, \ldots, T_p$, jedno vozilo ih može sve obaviti ako vrijedi uvjet $\sum_{i=1}^{p} T_i \leq T$, gdje je T vrijeme u kojem sva vozila moraju obaviti svoje rute

- *Vremenski okviri*, gdje želimo da su sve dostave obavljene u nekom vremenu, ili svaka točka dostave ima svoje radno vrijeme, ili svaki vozač ima određen broj sati koji smije provesti na cesti, itd. Ovakav primjer smo već imali u prvom poglavlju (vidi odjeljak 1.3), te ćemo se i kasnije baviti njime pa ovdje ne iznosimo detalje

Karakteristike vozila

Do sada smo većinom promatrali probleme u kojima imamo identična vozila, ali kako smo već vidjeli, vozila mogu imati različite kapacitete, ali i brzinu, troškove i mogućnosti utovara/istovara, stoga ovdje promatramo različita vozila.

- *Višeskladišni VRP*, kako i ime govori, pretpostavljamo da imamo homogena (identična) vozila koja počinju i završavaju rute u različitim skladištima. U teoriji, svako vozilo može imati svoje skladište iz kojeg kreće i u kom završava, stoga svakom vozilu k pridružujemo o_k početnu točku i d_k završnu točku

- *Miješana vozila*, opet, iz imena zaključujemo da se radi o slučaju kada su vozila heterogena (različita). Pretpostavljamo da je flota vozila K particionirana u $|P|$ podskupova homogenih vozila (vrsta vozila), tj. $K = K_1 \cup K_2 \cup \cdots \cup K_{|P|}$. Za svako vozilo tipa p, tj. $\forall k \in K_p$, imamo zadan kapacitet Q_k, trošak c_{ijk}, vrijeme putovanja t_{ijk} i ponekad (ovisno o problemu kojem modeliramo) trošak mirovanja FC_k i podskup dostižnih vrhova V_k. Troškove mirovanja koristimo u slučaju kada ne koristimo sva vozila. Tada FC_k modelira trošak održavanja i pripreme vozila, te dostupnost vozača (u slučaju da je vozilo ipak potrebno)

Globalni uvjeti

Često izvedivost rješenja ovisi isključivo o lokalnim uvjetima, tj. izvedivosti ruta. Ovdje promatramo izvedivost rješenja, ovisno o tome kakve su izvedive rute.

- *Ravnotežni uvjet*, koji daje toleranciju na razliku vremenski najdulje i najkraće rute, pa se tada rad može ravnomjerno rasporediti između vozača.

- *Restriktivni uvjet*, kojim određujemo gornju granicu za neku stavku, npr. najveći broj vozila koji može doći u neko skladište. Također možemo ograničiti neke karakteristike ruta, npr. broj dugačkih ruta (ruta duljih od L, na kojima treba obići više od S vrhova ili za koje je vrijeme povratka veće od T) ili rute koje prolaze kroz neko područje (možda je cestarina kroz neku državu skupa, pa želimo ograničiti broj prolazaka kroz nju)

- *Uvjet usklađivanja*, gdje imamo neovisne rute i vozila koje želimo koordinirati, npr. želimo uskladiti dva tehničara koji obilaze lokacije tako da prvi dolazi na lokaciju,

ostavlja materijal i odlazi, a drugi dolazi neposredno nakon njega i obavlja posao sa materijalom. Također možemo imati i varijantu u kojoj ne moramo obilaziti vrhove već samo bridove, npr. na višetračnim autocestama, potrebno je više ralica koje čiste svaka svoju traku, ali svaka je za neko vrijeme t ispred druge, te čisti snijeg sa svoje trake na onu do nje, nakon čega dolazi druga koja čisti svoju traku itd.

Ciljevi

Do sada smo imali isključivo probleme u kojima smo minimizirali trošak, no možemo imati i drugačije ciljeve, od jedinstvenog cilja do hijerarhijskog.

- *Optimizacija jedinstvenog cilja*, umjesto minimizacije troška ili maksimizacije zarade, možemo željeti maksimizirati zadovoljstvo kupaca, čime zadovoljstvo povećavamo ranijim posluživanjem. Također, ako vrhove zamislimo kao mjesta na kojima je potrebna humanitarna pomoć, tada želimo u najkraćem mogućem vremenu dostaviti humanitarnu pomoć u sve vrhove. Nadalje, možemo zamišljati vozila kao javni prijevoz, a dostave na točke kao broj ljudi koji izlazi na određenoj stanici, pa želimo minimizirati vrijeme koje osobe provedu u javnom prijevozu. Sve ove varijacije problema smatramo kao minimizaciju vremenskih ograničenja. No, možemo željeti optimizirati i ravnotežni uvjet (min-max cilj), dakle želimo minimizirati najdulju rutu. U praksi, ovakva optimizacija je rijetko kad smislena jer su dobivene rute često vrlo neefektivne

- *Hijerarhijski ciljevi*, općenito govoreći, minimizacija duljina ruta, trajanja ruta i vremena izvršavanja cijele operacije su međusobno konfliktni ciljevi, pa ako još tome dodamo i minimizaciju flote vozila, koja je opet konfliktna s njima, nećemo doći daleko. Stoga je praksa da se najprije minimizira broj vozila, jer broj vozila dalje određuje troškove ruta, te da se zatim odabere sekundarni minimizacijski cilj, u praksi je to najčešće minimizacija duljina ruta

Ostalo

Vidjeli smo da možemo konstruirati vrlo zanimljive probleme kada klasični VRP uparimo s nekim logističkim zahtjevom iz stvarnog svijeta. Stoga navodimo još neke:

- *Dosljednost ruta*, ovaj uvjet imamo kada imamo čestu ili redovitu dostavu robe. Ovdje zahtijevamo da isti vozač poslužuje iste klijente u otprilike isto vrijeme svaki put kada taj klijent treba dostavu. Često je ovaj uvjet uparen sa sljedećim

- *Kompaktnost ruta*, gdje konstruiramo kompaktne teritorije od skupa svih vrhova koji nadalje služe za dnevno određivanje ruta. Alternativno, možemo od skupa svih vr-

hova napraviti particiju na grupe (*clustere*), koji se ponašaju na način da ako vozilo uđe u jedan vrh u grupi (*clusteru*), ne može izaći iz njega prije nego posluži sve vrhove u njemu

Od navedenih varijanti problema usmjeravanja vozila mi ćemo se detaljno baviti samo nekima, tj. VPR s vremenskim ograničenjima, VRP s utovarom-istovarom i periodičnim VRP. Za VRP s utovarom-istovarom i periodičnim VRP-om zadržavamo zahtjeve vremenskih ograničenja jer utovar-istovar i periodične operacije također treba zadovoljiti u nekom vremenskom intervalu.

2.3 Problem usmjeravanja vozila s vremenskim ograničenjima

VRP s vremenskim ograničenjima je, naravno, direktno poopćenje TSP-a s vremenskim ograničenjima iz prvog poglavlja, gdje imamo zahtjeve za najranije i najkasnije posluživanje kupca. Formalno, promatramo flotu $K = \{1, 2, \ldots, |K|\}$ heterogenih vozila koji poslužuju n kupaca, $N = \{1, 2, \ldots, n\}$. Svaki kupac $i \in N$ zahtijeva q_i proizvoda koji mu se moraju dostaviti u vremenskom intervalu $[a_i, b_i]$. Svako vozilo $k \in K$ ima kapacitet Q_k, trošak c_{ijk} za prijelaz od vrha i do vrha j, te potrebno vrijeme τ_{ijk}. Cilj nam je pronaći $|K|$ Hamiltonovih ciklusa, na način da obiđemo sve kupce točno jednom, i u zahtjevanom vremenskom okviru, i pritom minimiziramo trošak. Neka je M neki dovoljno veliki broj, te $s_{ik} \geq 0$ vrijeme kada je vozilo $k \in K$ posjetilo vrh i. Vrijednost varijable s_{ik} je dana velikim-M uvjetom (2.28), pa formalno, ako vozilo k ne posjećuje vrh i, iz velikog-M uvjeta i uvjeta $s_{ik} \geq 0$, određujemo vrijednost varijable.

Uz identične oznake kao i kod općenitog problema usmjeravanja vozila, uz dodatak gornjih, imamo sljedeću minimizacijsku zadaću:

$$\sum_{k \in K} \sum_{(i,j) \in E} c_{ijk} x_{ijk} \to \min \tag{2.22}$$

$$\forall i \in N, \quad \sum_{k \in K} \sum_{\substack{j \in V \\ j \neq i}} x_{ijk} = 1 \tag{2.23}$$

$$\forall k \in K, \quad \sum_{\substack{j \in V \\ j \neq 0, n+1}} x_{0jk} = 1 \tag{2.24}$$

$$\forall k \in K, \forall h \in N, \quad \sum_{\substack{i \in V \\ i \neq h}} x_{ihk} - \sum_{\substack{j \in V \\ j \neq h}} x_{hjk} = 0 \tag{2.25}$$

$$\forall k \in K, \quad \sum_{\substack{i \in V \\ i \neq n+1}} x_{i,n+1,k} = 1 \tag{2.26}$$

$$\forall k \in K, \quad \sum_{i \in N} (q_i \sum_{j \in V} x_{ijk}) \leq Q_k \tag{2.27}$$

$$\forall (i, j) \in V \times V, i \neq j, \forall k \in K, \quad s_{ik} + \tau_{ijk} - M(1 - x_{ijk}) \leq s_{jk} \tag{2.28}$$

$$\forall i \in N, \forall k \in K, \quad a_i \leq s_{ik} \leq b_i \tag{2.29}$$

$$\forall k \in K, \quad s_{0k} = 0 \tag{2.30}$$

$$x_{ijk} \in \{0, 1\}, s_{ik} \geq 0 \tag{2.31}$$

Uvjeti (2.22)-(2.26) su identični kao i kod opće formulacije VRP-a. (2.27) osigurava da suma svega utovarenog u k-to vozilo ne prelazi kapacitet vozila. (2.28) je veliki-M uvjet koji smo imali i kod TSP-a s vremenskim ograničenjima. Dakle, za $x_{ijk} = 1$ imamo da je $s_{ik} + \tau_{ijk} \leq s_{jk}$, dakle, vrijeme dolaska u vrh j je manje ili jednako vremenu dolaska do vrha i uvećanog za vrijeme putovanja od i do j; ako je $x_{ijk} = 0$, tvrdnja vrijedi jer smo za M uzeli neki dovoljno velik broj (baš tako da bi ovo vrijedilo). (2.29) osigurava da u svaki vrh stignemo u zadanom vremenskom intervalu (ako vozilo k ne posjećuje vrh i, vrijeme s_{ik} u ovom uvjetu je nebitno, pa se zanemaruje), dok (2.30) govori da sva vozila odlaze iz centralnog skladišta u trenutku 0.

Uočimo da je ovakva formulacija direktno proširenje MTZ formulacije TSP-a s vremenskim ograničenjima, pa veliki-M uvjet rješava problem stvaranja podciklusa, te nam dodatno regulira i vremena dolaska u vrhove za sva vozila.

2.4 Problem usmjeravanja vozila s utovarom-istovarom

U praksi se često traži da se roba pokupi na nekoj lokaciji prije dostave na drugu lokaciju, naravno u nekom vremenskom intervalu. Stoga VRP s utovarom-istovarom možemo zamišljati kao VRP s vremenskim ograničenjima uz još neke restrikcije. Neka imamo n lokacija za dostavu, dakle tada imamo skup vrhova $N = \{0, 1, \ldots, 2n + 1\}$. Vrhove 0 i $2n + 1$ identificiramo sa odlaskom i povratkom u centralno skladište, dok od ostatka skupa N radimo particiju na dva skupa: $P = \{1, 2, \ldots, n\}$ skup vrhova u kojima moramo utovariti robu, te $D = \{n+1, n+2, \ldots, 2n\}$ skup vrhova u koje moramo dostaviti robu. Svaki zahtjev $i = 1, 2, \ldots, n$ treba utovariti s težinom $q_i > 0$ u vrhu $i \in P$ te ga isto vozilo koje ga je utovarilo treba dostaviti u vrh $i + n \in D$ s težinom $q_{i+n} = -q_i$. Utovar, odnosno istovar, traje d_i vremena. Na raspolaganju je heterogena flota vozila s kapacitetom Q_k, vremenom putovanja τ_{ijk}, i troškom putovanja c_{ijk}. U vrh $i \in D$ mora biti izvršena dostava u intervalu $[a_i, b_i]$. Iako imamo particiju skupa vrhova kao i kod crno-bijelog TSP-a, zbog različitosti zahtjeva, minimizacijski problem nije analogon BWTSP-a. Oznake su kao i kod VRP-a sa

vremenskim ograničenjem, uz dodatak $Q_{ik} \geq 0$ koji predstavlja količinu robe koju je vozilo k utovarilo do vrha i, uključujući i njega.

$$\sum_{k \in K} \sum_{(i,j) \in E} c_{ijk} x_{ijk} \to \min \tag{2.32}$$

$$\forall i \in P, \quad \sum_{k \in K} \sum_{\substack{j \in N \\ j \neq i}} x_{ijk} = 1 \tag{2.33}$$

$$\forall k \in K, \forall i \in P, \quad \sum_{\substack{j \in N \\ j \neq i}} x_{ijk} - \sum_{\substack{j \in N \\ j \neq n+1}} x_{i+n,jk} = 0 \tag{2.34}$$

$$\forall k \in K, \quad \sum_{\substack{j \in N \\ j \neq 0, 2n+1}} x_{0jk} = 1 \tag{2.35}$$

$$\forall k \in K, \forall i \in P \cup D, \quad \sum_{\substack{j \in N \\ j \neq i}} x_{jik} - \sum_{\substack{j \in N \\ j \neq i}} x_{ijk} = 0 \tag{2.36}$$

$$\forall k \in K, \quad \sum_{\substack{i \in N \\ i \neq 0, 2n+1}} x_{i,2n+1,k} = 1 \tag{2.37}$$

$$\forall (i,j) \in N \times N, i \neq j, \forall k \in K, \quad s_{ik} + \tau_{ijk} + d_i - M(1 - x_{ijk}) \leq s_{jk} \tag{2.38}$$

$$\forall (i,j) \in N \times N, i \neq j, \forall k \in K, \quad Q_{ik} + q_j - M(1 - x_{ijk}) \leq Q_{jk} \tag{2.39}$$

$$\forall i \in P, \forall k \in K, \quad s_{ik} + d_i + \tau_{i,i+n} \leq s_{i+n,k} \tag{2.40}$$

$$\forall i \in N, \forall k \in K, \quad a_i \leq s_{ik} \leq b_i \tag{2.41}$$

$$\forall k \in K, \quad s_{0k} = 0 \tag{2.42}$$

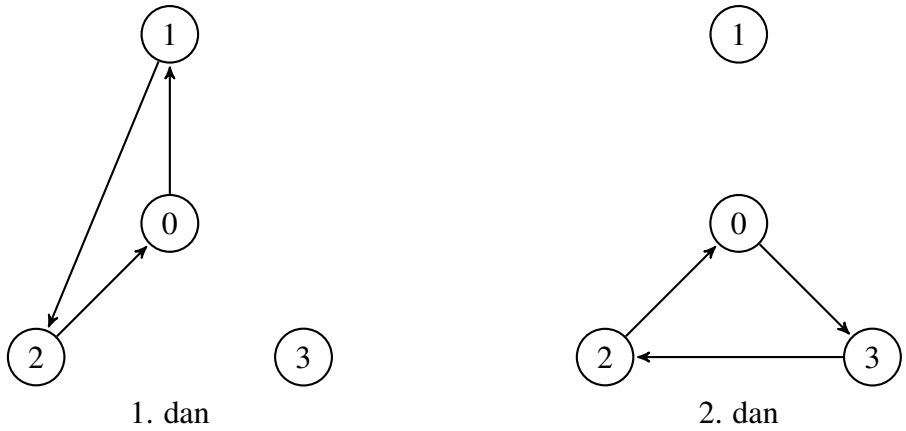
$$\forall i \in N, \forall k \in K, \quad \max\{0, q_i\} \leq Q_{ik} \leq \min\{Q_k, Q_k + q_i\} \tag{2.43}$$

$$x_{ijk} \in \{0, 1\}, s_{ik} \geq 0, Q_{ik} \geq 0 \tag{2.44}$$

(2.32) je minimizacija. (2.33) osigurava da se iz svih vrhova utovara odlazi u točno jedan vrh. (2.34) osigurava da isto vozilo obavlja utovar i istovar za danog kupca, dakle svi kupci su posluženi. Uvjet (2.35) nam garantira da sva vozila kreću iz centralnog skladišta. (2.37) je uvjet očuvanja toka. (2.37) osigurava da se svako vozilo mora vratiti u centralno skladište. (2.38) je veliki-M uvjet za regulaciju vremena iz VRP-a s vremenskim ograničenjima, uz dodani član d_i za vrijeme utovara i istovara. (2.39) je veliki-M uvjet za regulaciju kapaciteta vozila, sličan kao (2.19). Uvjet (2.40) osigurava da najprije obavimo utovar pa zatim istovar. (2.41) osigurava dostavljanje u zadanim intervalima. (2.42) osigurava da sva vozila kreću iz centralnog skladišta u trenutku 0. (2.43) daje doljnju i gornju granicu kumulativne količine robe u i-tom vrhu.

2.5 Periodični problem usmjeravanja vozila

Periodični VRP je varijanta problema usmjeravanja vozila s ponovljenim dostavama (vidi odjeljak 2.2). Kao primjer jednog periodičnog VRP možemo uzeti odvoz otpada u gradu u kojem imamo određen broj smetlarskih kamiona. Možemo pretpostaviti da neke lokacije, npr. ulice (ovdje bi također mogli prijeći na problem usmjeravanja bridova, ali se zadržavamo na vrhovima) zahtijevaju češći odvoz otpada od drugih. Dakle, na tjednoj bazi moramo posjetiti svaku ulicu određen broj puta, ne nužno s istim vozilima i ne nužno u istom redoslijedu (rute se mogu mijenjati iz dana u dan). Konkretno, uzmimo primjer poštanske dostave kroz dva dana na tri lokacije s jednim kamionom. Neka vrh 2 očekuje dvije dostave, a vrhovi 1 i 3 po jednu dostavu. Jedna mogućnost za rješenje ovakvog periodičnog VRP je dana sa slikom (2.2).



Slika 2.2: Primjer periodičnog VRP-a kroz 2 dana

Formalno, za periodični VTP imamo skup vrhova $N = \{1, 2, \ldots, n\}$, s heterogenom flotom vozila K, te $1, 2, \ldots, T$ vremenskih perioda za koje planiramo dostavu. Kao i prije, 0 i $n + 1$ predstavljaju odlazak i povratak u centralno skladište, pa je kao i prije $V = \{0, 1, \ldots, n + 1\}$. Vrh i mora biti poslužen najmanje θ_i puta, ovisno o rasporedu vozila R. Za neki raspored $r \in R$ koristimo oznaku $a_{rt} = 1$ ako je vremenski period $t \in T$ u rasporedu $r \in R$, inače $a_{rt} = 0$. Dakle, ako gledamo dane i rasporede vozila (koje vrhove obilaze) $a_{12} = 1$ bi značilo da se raspored 1 izvršava u 2. danu. Vrh i zahtijeva količinu q_i za koju je potrebno s_i vremena, te taj zahtjev mora biti obrađen u intervalu $[e_i, l_i]$. Svako vozilo $k \in K$ ima kapacitet Q_k, te može obraditi najviše D_k zahtjeva po ruti. Trošak i vrijeme su kao i prije c_{ijk} i τ_{ijk}. Cilj nam je pronaći skup rasporeda za određen broj vremenskih perioda takvih da su svi zahtjevi vrhova zadovoljeni, a da pritom minimiziramo trošak.

Potrebno je zadovoljiti dva bitna uvjeta: najprije moramo osigurati da naš skup rasporeda zadovolji minimalan broj posjeta za svaki vrh, te tek onda odrediti koje vrhove posjećujemo u kojim periodima tako da zadovoljimo rasporede. Stoga su nam potrebne malo drukčije binarne varijable nego ranije:

- $x^t_{ijk} = 1$, ako vozilo k putuje od i do j u vremenskom periodu t, a 0 inače
- $y_{ir} = 1$, ako je raspored r pridružen vrhu i, inače 0
- $z_{kt} = 1$, ako vozilo k vozi u vremenskom periodu t, inače 0
- $w^t_{ik} \geq 0$, početak usluge vozila k za vrh i u periodu t (nije binarna varijabla)

Minimizacijski problem je dan sa:

$$\sum_{t\in T}\sum_{k\in K}\sum_{(i,j)\in E} c_{ijk}x^t_{ijk} \to \min \tag{2.45}$$

$$\forall i \in N, \quad \sum_{r\in R} y_{ir} = 1 \tag{2.46}$$

$$\forall i \in N, \quad \sum_{r\in R}\sum_{t\in T} y_{ir}a_{rt} \geq \theta_i \tag{2.47}$$

$$\forall (i,j) \in E, \forall k \in K, \forall t \in T, \quad x^t_{ijk} \leq z_{kt} \tag{2.48}$$

$$\forall t \in T, \quad \sum_{k\in K} z_{kt} \leq |K| \tag{2.49}$$

$$\forall k \in K, \forall t \in T, \quad \sum_{j\in N} x^t_{0jk} = z_{kt} \tag{2.50}$$

$$\forall k \in K, \forall t \in T, \quad \sum_{i\in N} x^t_{i,n+1,k} = z_{kt} \tag{2.51}$$

$$\forall i \in N, \forall t \in T, \quad \sum_{k\in K}\sum_{\substack{j\in V\\ j\neq i}} x^t_{ijk} = \sum_{r\in R} y_{ir}a_{rt} \tag{2.52}$$

$$\forall i \in N, \forall k \in K, \forall t \in T, \quad \sum_{\substack{j\in V\\ j\neq i}} x^t_{ijk} = \sum_{\substack{j\in V\\ j\neq i}} x^t_{jik} \tag{2.53}$$

$$\forall k \in K, \forall t \in T, \quad \sum_{i\in N} q_i \sum_{j\in V} x^t_{ijk} \leq Q_k \tag{2.54}$$

$$\forall (i,j) \in E, \forall k \in K, \forall t \in T, \quad w^t_{ik} + s_i + \tau_{ijk} - M(1 - x^t_{ijk}) \leq w^t_{jk} \tag{2.55}$$

$$\forall i \in N, \forall k \in K, \forall t \in T, \quad e_i \sum_{\substack{j\in V\\ j\neq i}} x^t_{ijk} \leq w^t_{ik} \leq l_i \sum_{\substack{j\in V\\ j\neq i}} x^t_{ijk} \tag{2.56}$$

$$\forall k \in K, \forall t \in T, \quad w^t_{n+1,k} \leq D_k z_{kt} \tag{2.57}$$

$$\forall k \in K, \forall t \in T, \quad w^t_{0k} = 0 \tag{2.58}$$

$$x^t_{ijk}, y_{ir}, z_{kt}, w^t_{ik} \geq 0 \tag{2.59}$$

(2.45) je minimizacija, (2.46) pridružuje točno jedan raspored svakom vrhu. Uvjet (2.47) osigurava da je svaki vrh posjećen najmanje onoliko puta koliko želimo. (2.48) garantira da se vrh ne može posjetiti u nekom periodu ako to vozilo nije aktivno u tom periodu. (2.49) govori da je maksimalan broj aktivnih vozila u danu (tj. vremenskom periodu) najviše broj dostupnih vozila. (2.50) i (2.51) osiguravaju da aktivno vozilo u periodu t kreće iz i vraća se u centralno skladište. (2.52) kaže da svaki vrh koji je pokriven u periodu mora biti poslužen od strane točno jednog vozila. (2.53) je uvjet očuvanja toka. Uvjet (2.54) govori da svako vozilo ne smije utovariti više od svog kapaciteta. (2.55) je veliki-M uvjet za regulaciju vremena kakav smo već imali. U ovom slučaj on za $x^t_{ijk} = 1$ glasi $w^t_{ik} + s_i + \tau_{ijk} \leq w^t_{jk}$, dakle vrijeme kad vozilo k uslužuje vrh j u periodu t je najviše vrijeme kada uslužuje vrh i uvećan za zbroj vremena koje mu treba da posluži vrh i i vremena putovanja do j. (2.56) osigurava da vozilo koje vrši dostave u vremenskom periodu poslužuje vrh i u željenom intervalu. (2.57) daje gornju granicu na aktivnost vozila, tj. zadnje vrijeme kada se vozilo mora vratiti u centralno skladište. (2.58) govori da sva vozila kreću iz centralnog skladišta u vremenu 0.

Poglavlje 3

Implementacija optimizacijske formulacije

U ovom odjeljku ćemo prikazati implementaciju i rezultate neke od optimizacijskih formulacija VRP-a. Točnije, bavit ćemo se homogenim i heterogenim VRP-om, te VRP-om s vremenskim ograničenjima. Implementaciju vršimo u programskom jeziku Python (verzija 3.12.4). Koristit ćemo se već ugrađenim programskim knjižnicama *pulp, scipy, matplotlib* za optimizaciju, numeriku i vizualizaciju.

Ideja je uzeti gradove u sjeverozapadnoj Hrvatskoj u kojima se nalaze poslovnice nekog trgovačkog lanca, te iz centralnog skladišta u Zagrebu dovesti određenu količinu robe u svaku poslovnicu prije povratka u centralno skladište. Konkretno, naš skup podataka će se sastojati od 30 gradova, za koje učitavamo njihove geografske koordinate, te cestovne udaljenosti između svaka dva grada (na način da za udaljenost grada od samog sebe postavljamo neki veliki broj, a ne 0, da se prilikom optimizacije ne bismo zaglavili u nekom gradu). Nakon toga, računamo funkciju troška ovisno o udaljenosti. U heterogenom slučaju, funkcija troška će također ovisiti i o kapacitetu kamiona. Kod VRP-a s vremenskim ograničenjima također računamo i vremena putovanja svake vrste kamiona između svaka dva grada. Nakon učitavanja svih podataka postavljamo optimizacijski model te ga rješavamo pomoću funkcija iz programske knjižnice *pulp*. Kada smo riješili problem, ispisujemo vrijednost funkcije koju smo minimizirali, vizualni prikaz rješenja, te tumač vizualnog prikaza i vrijeme izvršavanja.

Svaka sljedeća implementacija se nadograđuje na prethodnu, dakle, najprije implementiramo homogeni VRP (slučaj kada imamo flotu identičnih vozila), koju potom generaliziramo na heterogeni VRP (imamo različite tipove vozila, u našem slučaju će to biti 3) na koji zatim dodajemo vremenska ograničenja da bismo dobili VRP s vremenskim ograničenjima. Dakle, kod VRP-a s vremenskim ograničenjima imamo i vremenska ograničenja i različite tipove vozila.

Za svaku od implementacija prilažemo programski kod, ispis i vizualni prikaz rješenja. Važno je napomenuti da se boja crta u svakoj od ruta bira nasumično, tako da se može dogoditi da je nekoliko različitih ruta označeno istom bojom. Također, praksom se pokazalo da se optimum postiže i prije nego program stvarno završi (npr. za homogeni VRP se program vrtio preko 2 sata, te još nije završio, no terminiranjem programa ranije, nakon 10, 5, pa čak i 3 minute, dobivao se isti rezultat za koji su naredbe $status = model.solve()$ i $print(LpStatus[status])$ ispisale vrijednost $Optimal$). Stoga svaku od implementacija terminiramo ranije naredbom CTRL + C, pa stoga i ispis za proteklo vrijeme možda nije reprezentativan. Programske kodove pozivamo na način da otvorimo .py datoteku u sučelju IDLE Shell 3.12.4 te naredbom F5 pokrećemo kod.

Sadržaj .txt datoteka koje učitavamo jest sljedeći:

```
ZAGREB
ČAKOVEC
PRELOG
VARAŽDIN
LUDBREG
KRIŽEVCI
KOPRIVNICA
ZELINA
VRBOVEC
ZABOK
KRAPINA
DUGO SELO
BJELOVAR
ČAZMA
DARUVAR
IVANIĆ
ZAPREŠIĆ
VELKA GORICA
SVETA NEDELJA
SAMOBOR
JASKA
KARLOVAC
OGULIN
KUTINA
NOVSKA
PETRINJA
SISAK
VARAŽDINSKE TOPLICE
PREGRADA
PITOMAČA
VIROVITICA
```

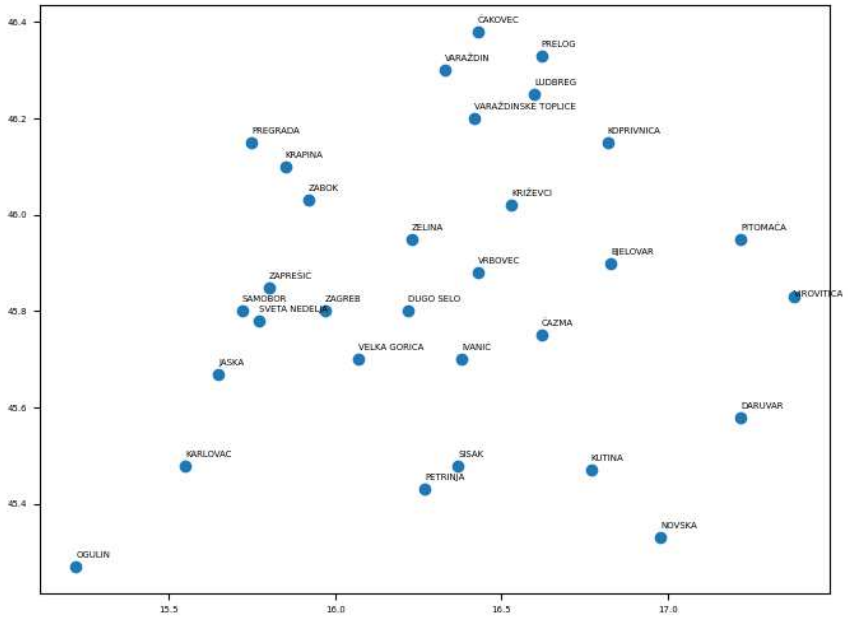
Slika 3.1: Datoteka Imena.txt

```
45.8 15.97
46.38 16.43
46.33 16.62
46.3 16.33
46.25 16.6
46.02 16.53
46.15 16.82
45.95 16.23
45.88 16.43
46.03 15.92
46.1 15.85
45.8 16.22
45.9 16.83
45.75 16.62
45.58 17.22
45.7 16.38
45.85 15.8
45.7 16.07
45.78 15.77
45.8 15.72
45.67 15.65
45.48 15.55
45.27 15.22
45.47 16.77
45.33 16.98
45.43 16.27
45.48 16.37
46.2 16.42
46.15 15.75
45.95 17.22
45.83 17.38
```

Slika 3.2: Datoteka Matrica_koordinata.txt

0	107	106	90.6	101	72.9	104	40.6	48.1	43.7
107	0	16.24	14.96	30.73	68.62	54.2	67.29	85.37	90.3
106	16.24	0	22.75	15.76	49.2	38.96	66.16	84.21	91.34
90.6	14.96	22.75	0	26.54	52.22	46.2	50.89	68.97	59.81
101	30.73	15.76	26.54	0	34.45	20.88	55.52	60.6	79.96
72.9	68.62	49.2	52.22	34.45	0	31.83	32.84	22.79	66.39
104	54.2	38.96	46.2	20.88	31.83	0	78.1	57.6	108
40.6	67.29	66.16	50.89	55.52	32.84	78.1	0	19.57	40.48
48.1	85.37	84.21	68.97	60.6	22.79	57.6	19.57	0	64.1
43.7	90.3	91.34	59.81	79.96	66.39	108	40.48	64.1	0
59.6	64.8	72.65	47.26	69.8	77.04	95.1	51.13	103.46	18.47
23	95.13	94	79.4	83.36	43.76	76.1	23.5	18.09	68.49
88.1	100.88	73.5	87.1	58.49	32.01	39.04	60.89	40.68	104.6
73.3	110	109	93.9	105	37.7	61.3	46.8	26	83.4
131	142	129	161	129	86	92.5	115	94.6	172

Slika 3.3: Dio datoteke Matrica_udaljenosti.txt



Slika 3.4: Prikaz svih gradova

Homogeni VRP

Homogeni VRP je klasični primjer problema usmjeravanja vozila, u slučaju kada je $k = 1$, odnosno kada imamo samo jedan tip vozila. Dakle, svi kamioni imaju jednak kapacitet, te smo u ovom primjeru posebno stavili da je zahtjev svakog grada također jednak. Implementaciju baziramo na formulaciji mrežnog toka iz početka prethodnog poglavlja.

Implementacija homogenog VRP-a u Pythonu:

```
import matplotlib.pyplot as plt
import numpy as np
from pulp import *
import random, time

start = time.time()

n = 30 #Broj točaka odnosno gradova za obilazak
kol = 8  #Broj dostupnih kamiona za obilazak


#UČITAVANJE PODATAKA IZ TXT DATOTEKA
f = open("Matrica_koordinata.txt", "r")
cm = []
for i in range(n+1):
    line = f.readline()
    ls = line.split()
    cm.append(ls[:])
f.close()

for i in range(n+1):
    for j in range(2):
        cm[i][j] = float(cm[i][j])

f = open("Matrica_udaljenosti.txt", "r")
dm = []
for i in range(n+2):
    line = f.readline()
    ls = line.split()
    dm.append(ls[:])
f.close()

for i in range(n+2):
    for j in range(n+2):
        dm[i][j] = float(dm[i][j])
        if dm[i][j] == 0:
            dm[i][j] = 10000

f = open("Imena.txt", "r", encoding = "utf-8")
```

```
nm = []
for i in range(n+1):
    line = f.readline()
    nm.append(line)
f.close()

cm1 = []
cm2 = []

for i in range(n+1):
    cm1.append(cm[i][0])
    cm2.append(cm[i][1])

#KONSTRUKCIJA "KARTE" (TOČKE LOKACIJE GRADOVA + NJIHOVA IMENA)
plt.rcParams['font.size'] = 5
fig = plt.figure(figsize = (8,6))
ax = fig.add_subplot(111)
plt.plot(cm2, cm1, 'o')
for i, txt in enumerate(nm):
    ax.text(cm2[i], cm1[i], txt)

#KONSTRUKCIJA MATRICE TROŠKA
tdm = []
tmp = []
for i in range(n+2):
    for j in range(n+2):
        #pretpostavljamo potrošnju od 30l/100km, po cijeni od 1.45€/l dizela za sve kamione radi homogenosti
        tdm.append(dm[i][j]*0.3*1.45)

#KLASIČNI CVRP
#SLUČAJ HOMOGENE FLOTE VOZILA

model = LpProblem("Homogeni_CVRP", LpMinimize)

V = range(n+2)          #SVI VRHOVI
N = range(1, n+1, 1)    #SVI OSIM PRVOG I ZADNJEG (CENTRALNO SKLADIŠTE)
K = range(kol)
Q = 35                  #KAPCITET VOZILA, INT ZBOG HOMOGENOSTI, INAČE VEKTOR
q = 5                   #ZAHTJEV KUPACA, INAČE VEKTOR, ALI ZA POČETAK CONST
var_x = LpVariable.dicts("x", [(i,j,k) for i in V for j in V for k in K], cat="Binary")
var_u = LpVariable.dicts("u", [(i,k) for i in N for k in K], lowBound = 0, cat="Continuous")

```

```
#FUKCIJA KOJU MINIMIZIRAMO
#JER IMAMO HOMOGENU FLOTU, TROŠKOVI SVIH VOZILA SU ISTI PA JE tdm[i][j][k]=tdm[i][j]
model += lpSum([tdm[i+(n+2)*j]*var_x[(i,j,k)] for i in V for j in V for k in K])

#UVJETI
for i in N:
    tmp = list(V)
    tmp.remove(i)
    model += lpSum([var_x[(i,j,k)] for j in tmp for k in K]) == 1
for k in K:
    model += lpSum([var_x[(0,j,k)] for j in N]) == 1
for k in K:
    for h in N:
        tmp = list(V)
        tmp.remove(h)
        model += lpSum([var_x[(i,h,k)] for i in tmp]) - lpSum(var_x[(h,j,k)] for j in tmp) == 0
for k in K:
    model += lpSum([var_x[(i, n+1,k)] for i in N]) == 1
for i in N:
    for j in N:
        if i != j:
            for k in K:
                model += var_u[(i,k)] - var_u[(j,k)] + Q*var_x[(i,j,k)] <= Q - q
for i in N:
    for k in K:
        model += var_u[(i,k)] >= q
        model += var_u[(i,k)] <= Q

#DODANI UVJET KOJI DAJE DA SVAKI KAMION MORA OBIĆI BAR DVIJE LOKACIJE
#ZA ELIMINIRANJE DEGENERIRANIH SLUČAJEVA
for k in K:
    model += lpSum(var_x[(i,j,k)] for i in N for j in N) >= 1


#RJEŠAVANJE
print(model)
status = model.solve()
print(LpStatus[status])
print("Minimalni trošak:", round(value(model.objective),4),"€")

end = time.time()
print("Proteklo vrijeme: ", round((end - start)/60,4), "min")
```

```
125 #GRAFIČKI PRIKAZ RJEŠENJA
126 color = ["#"+''.join([random.choice('0123456789ABCDEF') for j in range
        (6)]) for i in K]
127 for k in K:
128     c = color[k]
129     for i in V:
130         for j in V:
131             if value(var_x[(i,j,k)]) == 1 and i != j:
132                 if i == n+1:
133                     i = 0
134                 elif j == n+1:
135                     j = 0
136                 elif i == n+1 and j == n+1:
137                     i = 0
138                     j = 0
139                 plt.arrow(cm2[i], cm1[i], -cm2[i]+cm2[j], -cm1[i]+cm1[j
        ], color=c)
140
141 flag = 0
142 for k in K:
143     for i in V:
144         for j in V:
145             if value(var_x[(i,j,k)]) == 1:
146                 print(k+1, ". kamion ide za: ", nm[j])
147                 flag = 1
148             if flag == 1: break
149         if flag == 1: break
150     flag = 0
151
152 plt.show()
```

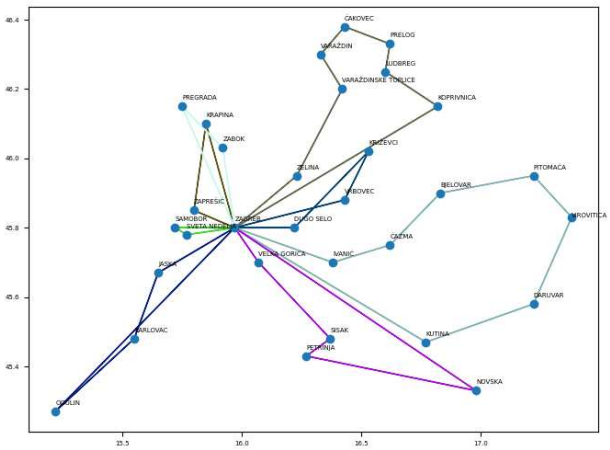
Od linija 72 do 114 koda definiramo i postavljamo uvjete na model, koji su zapravo uvjeti (2.14)-(2.21). Odnosno, najprije definiramo model kao problem minimizacije naredbom u 72. retku, zatim definiramo skupove vrhova, kapacitete i zahtjeve, te binarne i neprekidne varijable. Nadalje, naredbama *model+=...* dodajemo uvjete u definirani model. Svaka instanca naredbe *model+=...* definira redom jedan od uvjeta iz formulacije mrežnog toka, odnosno uvjete (2.14)-(2.21). Iznimka je posljednja pojava naredbe *model+=...* koju smo dodali kako bismo eliminirali degenerirane slučajeve Hamiltonovih ciklusa, tj. kamion posjećuje jedan grad i odmah se vraća u centralno skladište. Naredba *model.solve()* rješava problem minimizacije metodom linearnog programiranja, gdje je uobičajeni (*defaultni*) rješavač koji smo mi koristili *COIN-OR Branch and Cut* (CBC) rješavač. Na kraju, prilikom grafičkog prikaza rješenja (linije koda 125-140), valja (ponovno) napomenuti da se boja svakog od Hamiltonovih ciklusa bira nasumično, dakle, moguće je da dvije različite trase imaju istu boju. To ne znači da se isti kamion vraća u Zagreb, te zatim nastavlja vožnju novom trasom, već se radi o nasumičnosti odabira boja (tome služe linije koda 141-150,

da se točno ispiše koji kamion ide kojom trasom). Također, valja napomenuti da su trase crtane ravnim linijama zbog jednostavnosti dok su udaljenosti u stvarnosti cestovne.

Ispis i vizualizacija rješenja homogenog VRP-a:

```
Squeezed text (31425 lines).
Optimal
Minimalni trošak: 672.7231 €
Proteklo vrijeme:  6.2958 min
1 . kamion ide za:  SAMOBOR

2 . kamion ide za:  KUTINA

3 . kamion ide za:  KRAPINA

4 . kamion ide za:  KOPRIVNICA

5 . kamion ide za:  OGULIN

6 . kamion ide za:  PREGRADA

7 . kamion ide za:  VELKA GORICA

8 . kamion ide za:  VRBOVEC
```

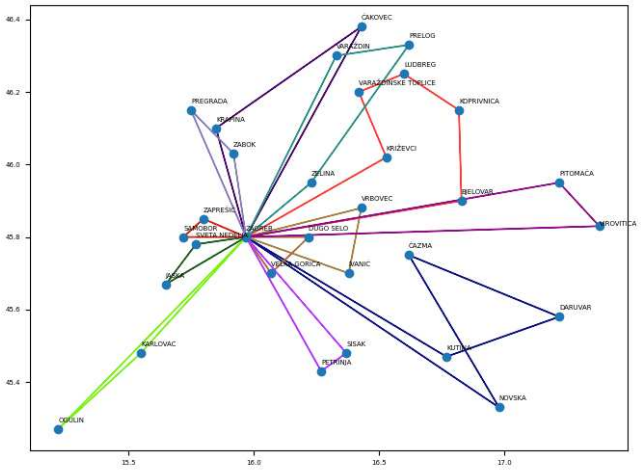
Slika 3.5: Ispis za homogeni VRP s 8 kamiona



Slika 3.6: Vizualizacija rješenja homogenog VRP-a s 8 kamiona

```
Squeezed text (47271 lines).
Optimal
Minimalni trošak: 982.4171 €
Proteklo vrijeme:  4.2709 min
1 . kamion ide za:  KUTINA

2 . kamion ide za:  KRIŽEVCI

3 . kamion ide za:  ČAKOVEC

4 . kamion ide za:  JASKA

5 . kamion ide za:  VARAŽDIN

6 . kamion ide za:  IVANIĆ

7 . kamion ide za:  PREGRADA

8 . kamion ide za:  DUGO SELO

9 . kamion ide za:  VIROVITICA
10 . kamion ide za:  ZAPREŠIĆ

11 . kamion ide za:  PETRINJA

12 . kamion ide za:  OGULIN
```

Slika 3.7: Ispis za homogeni VRP s 12 kamiona



Slika 3.8: Vizualizcija rješenja homogenog VRP-a s 12 kamiona

Heterogeni VRP

Heterogeni VRP je općeniti oblik CVRP-a u formulaciji mrežnog toka, za općeniti broj tipa vozila $k \in \mathbb{N}$. U homogenom slučaju smo imali samo jedan tip vozila, odnosno $k = 1$. Za heterogeni, svaki kamion pripada jednom od k tipova kamiona od kojih svaki tip ima svoju potrošnju i kapacitet. Dakle, potrošnja i kapacitet su identični unutar klase istih tipova vozila. Ovdje smo i poopćili potraživanja gradova, na način da nemaju svi gradovi istu potražnju kao kod homogenog CVP-a.

Jer imamo identični prvi dio (do linije 61) koda kada učitavamo datoteke, te posljednji dio (od linije 125) kada crtamo Hamiltonove cikluse i biramo boje, taj dio izostavljamo. Bitno je samo napomenuti da smo uveli varijablu l kojom smo označili broj tipova vozila. Uz ove napomene, imamo implementaciju heterogenog VRP u Pythonu (za $kol = 9$ i $l = 3$):

```
1  #KONSTRUKCIJA MATRICE TROŠKA
2  tdm = []
3  tmp = []
4  for k in range(kol):
5      for i in range(n+2):
6          for j in range(n+2):
7              #potrošnja ovisi o kojem se tipu kamiona radi
8              potr = 0.3 - (k % l) * (0.3/(2*l))
9              tdm.append(dm[i][j]*potr*1.45)
10
11 #HETEROGENI CVRP
12
13 model = LpProblem("Heterogeni_CVRP", LpMinimize)
14
15 V = range(n+2)           #SVI VRHOVI
16 N = range(1, n+1, 1)     #SVI OSIM PRVOG I ZADNJEG (CENTRALNO SKLADIŠTE)
17 K = range(kol)
18 Q = [40, 30, 20, 40, 30, 20, 40, 30, 20]     #KAPACITETI VOZILA
19 q = [5, 3, 7, 4, 5, 5, 4, 5, 5, 5, 6, 6, 4, 5, 5, 6, 6, 6, 6, 6, 7, 7,
       6, 4, 3, 7, 3, 3, 4, 7]  #ZAHTJEVI KUPACA
20 var_x = LpVariable.dicts("x", [(i,j,k) for i in V for j in V for k in K
       ], cat="Binary")
21 var_u = LpVariable.dicts("u", [(i,k) for i in N for k in K], lowBound =
       0, cat="Continuous")
22
23 #FUKCIJA KOJU MINIMIZIRAMO
24 model += lpSum([tdm[i+(n+2)*j+(n+2)**2*k]*var_x[(i,j,k)] for i in V for
       j in V for k in K])
25
26 #UVJETI
27 for i in N:
28     tmp = list(V)
29     tmp.remove(i)
```

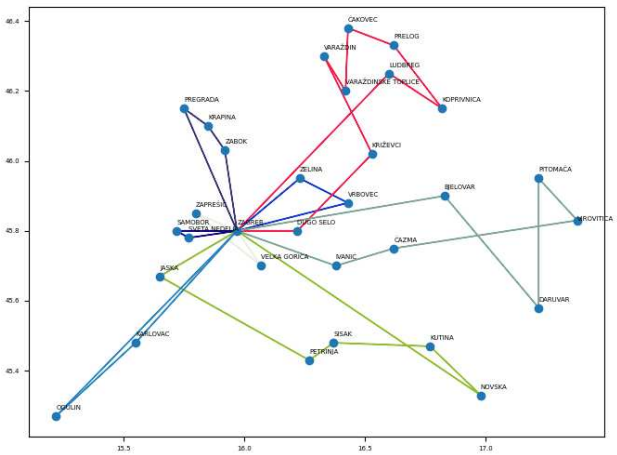
```
30     model += lpSum([var_x[(i,j,k)] for j in tmp for k in K]) == 1
31 for k in K:
32     model += lpSum([var_x[(0,j,k)] for j in N]) == 1
33 for k in K:
34     for h in N:
35         tmp = list(V)
36         tmp.remove(h)
37         model += lpSum([var_x[(i,h,k)] for i in tmp]) - lpSum(var_x[(h,j
    ,k)] for j in tmp) == 0
38 for k in K:
39     model += lpSum([var_x[(i, n+1,k)] for i in N]) == 1
40 for i in N:
41     for j in N:
42         if i != j:
43             for k in K:
44                 model += var_u[(i,k)] - var_u[(j,k)] + Q[k]*var_x[(i,j,k
    )] <= Q[k] - q[j-1]
45 for i in N:
46     for k in K:
47         model += var_u[(i,k)] >= q[i-1]
48         model += var_u[(i,k)] <= Q[k]
49
50 #DODANI UVJET KOJI DAJE DA SVAKI KAMION MORA OBIĆI BAR DVIJE LOKACIJE
51 #ZA ELIMINIRANJE DEGENERIRANIH SLUČAJEVA
52 for k in K:
53     model += lpSum(var_x[(i,j,k)] for i in N for j in N) >= 1
54
55
56 #RJEŠAVANJE
57 print(model)
58 status = model.solve()
59 print(LpStatus[status])
60 print("Minimalni trošak:", round(value(model.objective),4),"€")
61
62 end = time.time()
63 print("Proteklo vrijeme: ", round((end - start)/60,4), "min")
```

Uočimo da smo u odnosu na formulaciju mrežnog toka za homogeni slučaj, gdje smo imali identične zahtjeve i kapacitete, ovdje u liniji 44 stavili *Q[k]* i *q[j-1]* jer sada kapaciteti i zahtjevi više nisu uniformni, tj. skalari, nego su vektori kojima moramo pristupati ovisno o kojem se gradu i kamionu radi. U linijama 47 i 48 se događa ista stvar: kako više nemamo skalare kao kapacitete i zahtjeve, moramo osigurati da je varijabla u_{ik} ograđena pripadnim Q_k i q_i koji su sada vektori, dakle pristupamo pripadnim elementima vektora. Kao i kod homogenog VRP-a, vrijede napomene o nasumičnosti boja Hamiltonovih ciklusa, te o vizualizaciji preko ravnih crta (udaljenosti nisu Euklidske već cestovne, Euklidske udaljenosti su nacrtane radi bolje vizualizacije rješenja).

Ispis i vizualizacija rješenja heterogenog VRP-a:

```
Squeezed text (31388 lines).
Optimal
Minimalni trošak: 603.6705 €
Proteklo vrijeme:  3.2976 min
1 . kamion ide za:  VELKA GORICA

2 . kamion ide za:  ZABOK

3 . kamion ide za:  VRBOVEC

4 . kamion ide za:  DUGO SELO

5 . kamion ide za:  JASKA

6 . kamion ide za:  KARLOVAC

7 . kamion ide za:  SAMOBOR

8 . kamion ide za:  BJELOVAR
```

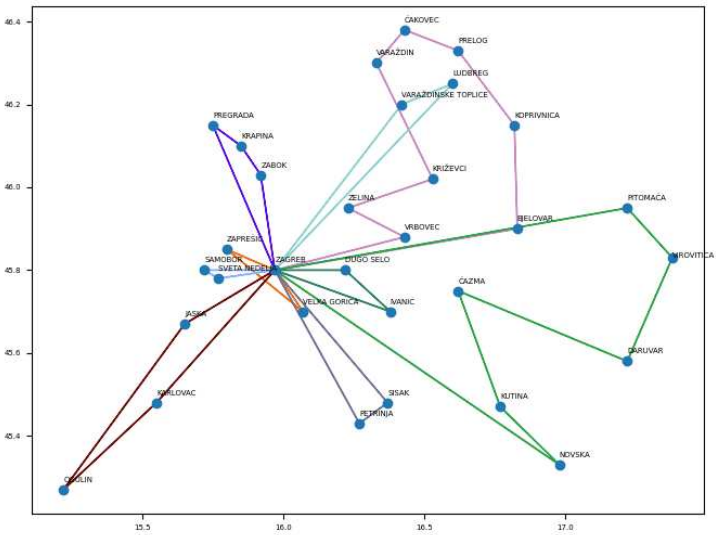
Slika 3.9: Ispis za heterogeni VRP za kol=8 i l=3



Slika 3.10: Vizualizacija rješenja heterogenog VRP-a za kol=8 i l=3

```
Squeezed text (35391 lines).
Optimal
Minimalni trošak: 637.3163 €
Proteklo vrijeme:  4.5081 min
1 . kamion ide za:  BJELOVAR

2 . kamion ide za:  PITOMAČA

3 . kamion ide za:  ZABOK

4 . kamion ide za:  ZAPREŠIĆ

5 . kamion ide za:  IVANIĆ

6 . kamion ide za:  KARLOVAC

7 . kamion ide za:  SISAK

8 . kamion ide za:  VARAŽDINSKE TOPLICE

9 . kamion ide za:  SVETA NEDELJA
```

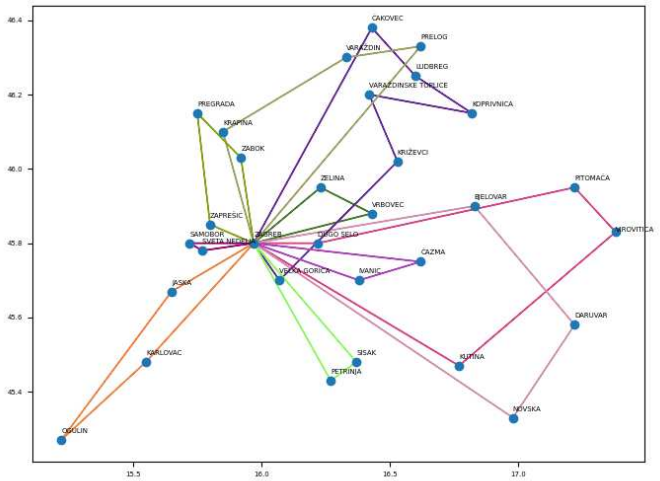
Slika 3.11: Ispis za heterogeni VRP za kol=9 i l=4



Slika 3.12: Vizualizacija rješenja heterogenog VRP-a za kol=9 i l=4

```
Squeezed text (39295 lines).
Optimal
Minimalni trošak: 685.4939 €
Proteklo vrijeme:  5.0865 min
1 . kamion ide za:  VRBOVEC

2 . kamion ide za:  KUTINA

3 . kamion ide za:  SAMOBOR

4 . kamion ide za:  KARLOVAC

5 . kamion ide za:  BJELOVAR

6 . kamion ide za:  ČAKOVEC

7 . kamion ide za:  IVANIĆ

8 . kamion ide za:  PRELOG

9 . kamion ide za:  ZAPREŠIĆ

10 . kamion ide za:  SISAK
```

Slika 3.13: Ispis za heterogeni VRP za kol=10 i l=5



Slika 3.14: Vizualizacija rješenja heterogenog VRP-a za kol=10 i l=5

VRP s vremenskim ograničenjima

VRP s vremenskim ograničenjima je nadogradnja na heterogeni VRP na način da za svaki tip vozila još računamo, za svaka dva grada, vrijeme potrebno da vozilo dođe iz jednog u drugi, te tada možemo uvažiti vremenska ograničenja. Odnosno, svaki od gradova zahtjeva da mu se roba dostavi u određenom intervalu, dakle imamo najranije i najkasnije vrijeme dostave. Naša implementacija se bazira na formulaciji mrežnog toka za VRP s vremenskim ograničenjima. U stvarnoj implementaciji bi možda dodatno zahtijevali da vozači ne čekaju (npr. ako kreću u 7, i do svake lokacije im je potrebno sat vremena, a vremenska ograničenja ne dopuštaju primitak robe prije 9) ili da imaju najveću dozvoljenu količinu vremena koju smiju provesti u vozilu, dok ovdje nema takvih ograničenja.

Kao i ranije, izbacujemo dijelove koda koji su identični. Implementacija VRP-a s vremenskim ograničenjima u Pythonu tada glasi:

```
#KONSTRUKCIJA MATRICE TROŠKA
tdm = []
tmp = []
vdm = []
for k in range(kol):
    for i in range(n+2):
        for j in range(n+2):
            #potrošnja ovisi o kojem se tipu kamiona radi
            potr = 0.3 - (k % l) * (0.3/(2*l))
            #prosječna brzina ovisi o tipu kamiona
            brz = 90 + (k % l) * 10
            tdm.append(dm[i][j]*potr*1.45)
            vdm.append(dm[i][j]/brz)

#HETEROGENI CVRP

model = LpProblem("TWVRP", LpMinimize)

V = range(n+2)          #SVI VRHOVI
N0 = range(n+1)         #SVI OSIM ZADNJEG
N = range(1, n+1, 1)    #SVI OSIM PRVOG I ZADNJEG (CENTRALNO SKLADIŠTE)
K = range(kol)
Q = [40, 30, 20, 40, 30, 20, 40, 30, 20]    #KAPACITETI VOZILA
q = [5, 3, 7, 4, 5, 5, 4, 5, 5, 5, 6, 6, 4, 5, 5, 6, 6, 6, 6, 6, 7, 7,
    6, 4, 3, 7, 3, 3, 4, 7]  #ZAHTJEVI KUPACA
A = [1, 2, 1, 2, 1, 2, 2, 2, 3, 2, 2, 1, 3, 2, 2, 2, 1, 3, 2, 3, 1, 2,
    2, 3, 3, 2, 3, 3, 3, 2]  #NAJRANIJE VRIJEME DOSTAVE
B = [8, 7, 6, 7, 8, 7, 7, 7, 6, 7, 7, 8, 6, 7, 7, 7, 8, 6, 7, 6, 8, 7,
    7, 6, 6, 7, 6, 6, 6, 7]  #NAJKASNIJE VRIJEME DOSTAVE
M = 50 #DOVOLJNO VELIKI BROJ ZA VELIKI-M UVJET
var_x = LpVariable.dicts("x", [(i,j,k) for i in V for j in V for k in K
    ], cat="Binary")
```

```
var_s = LpVariable.dicts("s", [(i,k) for i in N0 for k in K], lowBound = 0, cat="Continuous")

#FUKCIJA KOJU MINIMIZIRAMO
model += lpSum([tdm[i+(n+2)*j+(n+2)**2*k]*var_x[(i,j,k)] for i in V for j in V for k in K])

#UVJETI
for i in N:
    tmp = list(V)
    tmp.remove(i)
    model += lpSum([var_x[(i,j,k)] for j in tmp for k in K]) == 1
for k in K:
    model += lpSum([var_x[(0,j,k)] for j in N]) == 1
for k in K:
    for h in N:
        tmp = list(V)
        tmp.remove(h)
        model += lpSum([var_x[(i,h,k)] for i in tmp]) - lpSum(var_x[(h,j,k)] for j in tmp) == 0
for k in K:
    model += lpSum([var_x[(i, n+1,k)] for i in N]) == 1
for k in K:
    model += lpSum([q[i-1]*var_x[(i,j,k)] for i in N for j in V]) <= Q[k]
for i in N:
    for j in N:
        if i != j:
            for k in K:
                model += var_s[(i,k)] + vdm[i+(n+2)*j+(n+2)**2*k] - M*(1-var_x[(i,j,k)]) <= var_s[(j,k)]
for i in N:
    for k in K:
        model += var_s[(i,k)] >= A[i-1]
        model += var_s[(i,k)] <= B[i-1]
for k in K:
    model += var_s[(0,k)] == 0

#DODANI UVJET KOJI DAJE DA SVAKI KAMION MORA OBIĆI BAR DVIJE LOKACIJE
#ZA ELIMINIRANJE DEGENERIRANIH SLUČAJEVA
for k in K:
    model += lpSum(var_x[(i,j,k)] for i in N for j in N) >= 1


#RJEŠAVANJE
print(model)
status = model.solve()
```

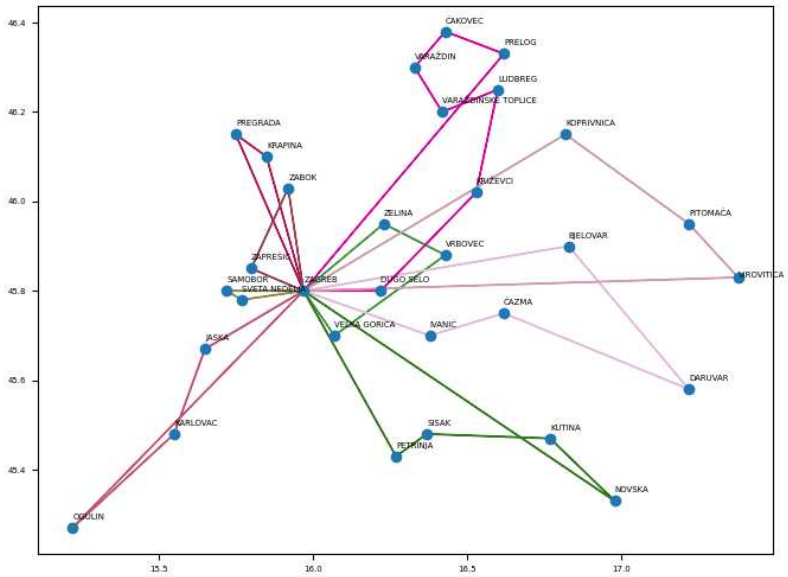
```
print(LpStatus[status])
print("Minimalni trošak:", round(value(model.objective),4),"€")

end = time.time()
print("Proteklo vrijeme: ", round((end - start)/60,4), "min")
```

Ovaj puta uvjeti koje modeliramo su uvjeti (2.22)-(2.31) iz formulacije problema usmjeravanja vozila s vremenskim ograničenjima. Svaka pojava naredbe *model*+=... modelira jedan od uvjeta (2.22)-(2.31) redom, osim uvjeta (2.29) koji smo rastavili na dva, te uvjeta (2.31) koji je zadan preko definicije varijabli. Sve napomene iz prethodnih implementacija i dalje vrijede. Dakle, rješavač je i dalje isti, boja Hamiltonovih ciklusa je nasumična, te ravne crte u vizualizaciji ne predstavljaju stvarne putove između gradova.

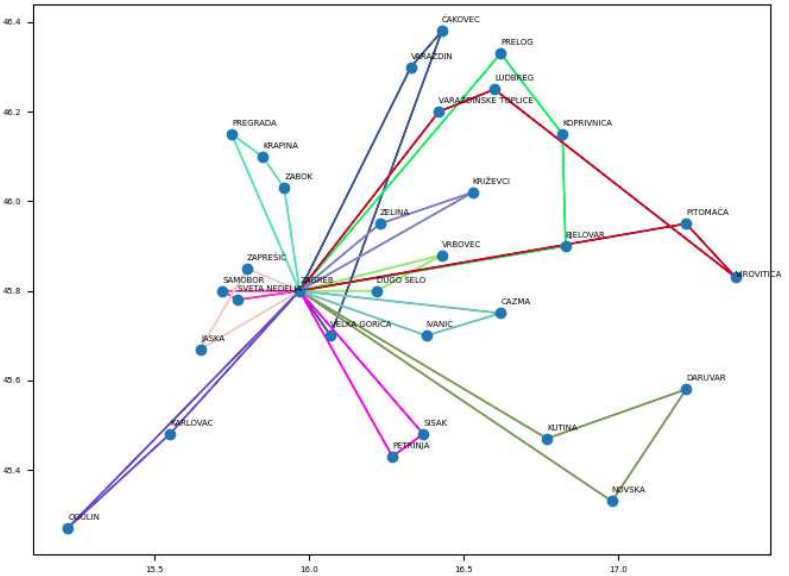
Ispis i vizualizacija rješenja VRP-a s vremenskim ograničenjima:

```
Squeezed text (37505 lines).
Optimal
Minimalni trošak: 608.8463 €
Proteklo vrijeme:  4.0384 min
1 . kamion ide za:  VELKA GORICA

2 . kamion ide za:  BJELOVAR

3 . kamion ide za:  NOVSKA

4 . kamion ide za:  PRELOG

5 . kamion ide za:  KRAPINA

6 . kamion ide za:  VIROVITICA
7 . kamion ide za:  SAMOBOR

8 . kamion ide za:  ZABOK

9 . kamion ide za:  JASKA
```

Slika 3.15: Ispis za VRP s vremenskim ograničenjima za kol=9 i l=3



Slika 3.16: Vizualizacija rješenja VRP-a s vremenskim ograničenjima za kol=9 i l=3



Slika 3.17: Vizualizacija rješenja VRP-a s vremenskim ograničenjima za kol=12 i l=3

```
Squeezed text (50142 lines).
Optimal
Minimalni trošak: 757.0225 €
Proteklo vrijeme:  6.467 min
1 . kamion ide za:  SAMOBOR

2 . kamion ide za:  VRBOVEC

3 . kamion ide za:  OGULIN

4 . kamion ide za:  ZAPREŠIĆ

5 . kamion ide za:  VELKA GORICA

6 . kamion ide za:  PREGRADA

7 . kamion ide za:  BJELOVAR

8 . kamion ide za:  NOVSKA

9 . kamion ide za:  PITOMAČA

10 . kamion ide za:  IVANIĆ

11 . kamion ide za:  PETRINJA

12 . kamion ide za:  KRIŽEVCI
```

Slika 3.18: Ispis za VRP s vremenskim ograničenjima za kol=12 i l=3

Bibliografija

[1] Ivica Nakić, *Diskretna matematika*, https://web.math.pmf.unizg.hr/nastava/komb/predavanja/predavanja.pdf, 2011, [Online; pristupljeno 7. ožujka 2024.].

[2] Adam Letchford i Richard Eglese, *The General Routing Problem*, str. 206–207, siječanj 2001.

[3] Haitao Li, *Optimization Modeling for Supply Chain Applications*, World Scientific, 2023.

[4] Paolo Toth i Daniele Vigo, *Vehicle routing: problems, methods, and applications*, SIAM, 2014.

[5] Wikipedia contributors, *Bin packing problem — Wikipedia, The Free Encyclopedia*, https://en.wikipedia.org/w/index.php?title=Bin_packing_problem&oldid=1214191623, 2024, [Online; pristupljeno 17. ožujka 2024.].

[6] ———, *Vehicle routing problem — Wikipedia, The Free Encyclopedia*, https://en.wikipedia.org/w/index.php?title=Vehicle_routing_problem&oldid=1211240282, 2024, [Online; pristupljeno 6. ožujka 2024.].

Sažetak

U ovom radu smo prikazali problem usmjeravanja vozila, kao poopćenje problema trgovačkog putnika. Zapis problema trgovačkog putnika kao problem traženja Hamiltonovog ciklusa najmanje težine smo sveli na problem optimizacije, jer je pronalaženje minimalnog Hamiltonovog ciklusa NP-težak problem.

Optimizacijski problem trgovačkog putnika nam je bila motivacija zapisa problema usmjeravanja vozila u terminima minimizacijske zadaće. Dali smo pregled i opis problema iz općenite klase problema usmjeravanja vozila, fokusirajući se na probleme s vremenskim ograničenjima, probleme utovara-istovara, te periodičnih problema.

Implementaciju optimizacijske formulacije problema usmjeravanja vozila smo izvršili u programskom jeziku Python, gdje smo uspjeli implementirali homogeni, heterogeni i problem s vremenskim ograničenjima na skupu podataka od 30 gradova.

Summary

In this paper we have given an overview of the vehicle routing problem (VRP), a generalization of the travelling salesman problem (TSP). We have shown that TSP as a problem of a minimal Hamiltonian cycle can be written as an optimization problem, since the finding of a minimal Hamiltionian cycle is a NP-hard problem.

TSP as an optimization problem also motivated us to write VRP as an optimization problem. We have given an overview of the generalized class of vehicle routing problems, of which we have focused on VRP with time windows, VRP with pickup-delivery, and periodic VRP.

The implementation of the VRP as an optimization problem was achieved via Python programming language, where we have managed to implement homogeneous VRP, heterogeneous VRP and VRP with time windows, with a dataset of 30 cities.

Životopis

Rođen sam 24.10.1999. u Bjelovaru. Nakon završetka Osnovne škole Čazma, upisao sam gimnaziju u Srednjoj školi Čazma. 2019. godine upisujem preddiplomski studij matematike na Matematičkom odsjeku PMF-a u Zagrebu, gdje 2022. upisujem diplomski studij Primijenjena matematika.