SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

**PROJEKT - Heurističke metode optimizacija**

# Usmjeravanje školskih autobusa s odabirom autobusnih stanica

*Zvonimir Jurelinac, Andrija Miličević*
Voditelj: *Izv. Prof. Dr. Sc. Lea Skorin-Kapov*

Zagreb, siječanj 2018.

# SADRŽAJ

# 1. Opis problema

Problem usmjeravanja školskih autobusa je varijacija problema usmjeravanja vozila, koji se od osnovnog problema razlikuje po tome što nije striktno zadano koja je sve mjesta potrebno posjetiti, već je dana lista potencijalnih stanica, i na algoritmu je da sam odabere koje će od njih i kojim redom posjetiti, te da pritom zadovolji sva zadana ograničenja.

Konkretno, u problemu su definirane lokacije potencijalnih stanica školskih autobusa, lokacije svih učenika koji školskim autobusima putuju do škole, te lokacija same škole u koju svi studenti moraju u što kraćem vremenu prispjeti. Zadatak se sastoji u tome da se odabere skup stanica koje će školski autobusi posjetiti, potom odredi koji će učenici propješačiti i biti pokupljeni na kojim stanicama, te na kraju, koji će autobusi i kojim redom posjetiti sve potrebne stanice. Svi učenici imaju (jednak) ograničen radijus kretanja, tako da im se može dodijeliti samo neka od stanica u njihovu dometu. Također, svi autobusi imaju ograničen broj mjesta, tako da ne mogu odjednom prevesti više učenika od onoga koliki im je kapacitet. Isto tako, jednu stanicu može posjetiti samo jedan autobus, i svi učenici koji čekaju na njoj moraju se tada ukrcati u taj autobus.

# 2. Algoritam

Tokom rada na projektu isprobano je mnoštvo različitih pristupa i algoritama. Pristup prikazan u nastavku dao je najbolje rezultate.

Algoritam djeluje neovisno na nekoliko razina. Prva razina je pridruživanje svakom studentu autobusnu stanicu pazeći pritom da se ne premaši granica maksimalnog kapaciteta jednog autobusa. Na drugoj razini iz skupa stanica koje se posjećuju odabire se redosljed njihovog posjećivanja. Na trećoj razini izvodi se lokalna pretraga koja pokušava poboljšati trenutni redosljed. Konačno na četvrtoj razini iz prethodnog redosljeda na optimalan način se raspoređuju autobusi koristeći dinamičko programiranje.

Algoritam izbacuje stanice koje su napunile maksimalni kapacitet te ih na kraju samo dodaje u rješenje. Ukoliko postoji 11 ili manje stanica koje nisu napunile maksimalni kapacitet, umjesto lokalne pretrage se vrši pretraga svih mogućih permutacija obilaska stanica.

Struktura algoritma dana je u nastavku. Svaki korak je detaljnije opisan u sljedećim podpoglavljima.

1. Pridruživanje autobusnih stanica studentima
2. Odabir redosljeda posjećivanja autobusnih stanica
3. Lokalna ili optimalna pretraga
4. Dinamičko programiranje

## 2.1. Prva razina - pridruživanje autobusnih stanica studentima

U praksi nam se pokazalo kako je dobro započeti rješenje pohlepnim pristupom te potom nekom usmjerenom pretragom poboljšati to rješenje. Tako se i u prvom koraku ovdje predloženog algoritma pokušava pridjeliti autobusne stanice studentima.

Dva kriterija su bila glavna za pridruživanje. Prvi kriterij je da se ukupno odabere što manje autobusnih stanica. Drugi kriterij je da su te stanice što bliže školi. Naime očito je isplativije da studenti hodaju što bliže školi nego da autobus ide duljim putem po njih. A i intuicija govori da ukoliko se posjećuje manje autobusnih stanica, vjerojatnije je da će ukupan put biti manji.

Algoritam gradi početno rješenje tako da kreće od stanica najbližih školi te pokušava naći stanicu koju mogu posjetiti najveći broj studenata. Ukoliko postoji više studenata koji mogu posjetiti određenu stanicu nego što je maksimalni kapacitet, algoritam na slučajan način odabire maksimalni podskup studenata koji će dodjeliti toj stanici. Ovo omogućuje da će se rješenja međusobno razlikovati.

Konačni algoritam je bio vrlo brz što nam je omogućilo da možemo mnogo puta pokretati algoritam s različitim početnim rješenjima. Zato je u pohlepno građenje početnog rješenje uveden šum koji neće uvijek uzeti najbolje rješenje prema ovoj pohlenoj heuristici. Postoji vjerojatnost da će se neke najbliže stanice ponekad preskočiti.

Ostali pokušaji generiranja početnog rješenja na drukčiji način nisu usporedivi s prethodno navedenim pristupom.

## 2.2. Druga razina - Odabir redosljeda posjećivanja autobusnih stanica

Iz prethodnog koraka dobili smo skup stanica koje se trebaju posjetiti. U ovom koraku se odabire redosljed posjećivanja stanica.

Algoritam prvo provjerava da li postoje stanice koje su napunile maksimalni kapacitet. Takve stanice će sigurno posjetiti samo jedan autobus te ih možemo odmah izbaciti iz daljnjeg algoritma te ih samo naknadno dodati u rješenje. Time smo smanjili vrijeme izvođenja te to vrijeme možemo dodatno utrošiti na pretragu prostora ili višestruko pokretanje.

Stanice koje nisu napunile maksimalni kapacitet se sortiraju prema kutu u odnosu na školu. Stanice sa sličnijem kutom bit će vjerojatno bliže u prostoru te je logičnije da isti autobus upravo njih posjećuje.

Sljedeći korak algoritma se izvodi nad ovim rješenjem, ali i nad rješenjem dobivenim rotacijom ovog rješenja. Ovakvim pristupom smo odredili gdje započinje prvi autobus, a najbolje rješenje ne mora nužno krenuti od kuta s najmanjom vrijednošću pri sortiranju stanica.

## 2.3. Treća razina - Lokalna ili optimalna pretraga

Ovaj korak se razlikuje u ovisnosti u broju stanica koje nisu maksimalno popunjene. Ukoliko je broj takvih stanica manje ili jednak 11, izvode se sve permutacije redosljeda stanica te se takvo rješenje šalje u sljedeći korak.

Ako je broj takvih stanica prevelik, izvodi se lokalna pretraga koja pokušava jednostavnim mutacijama pronaći bolja rješenja. Mutacije su zamjene dviju stanica u neposrednoj blizini ( ne nužno susjedne stanice ).

## 2.4. Četvrta razina - Dinamičko programiranje

Ovaj korak algoritma dobiva konačni redosljed stanica koje treba posjetiti. On odlučuje koliko će autobusa biti korišteno. Svaki autobus može posjećivati samo stanice koje su susjedne u prethodno dobivenom konačnom redosljedu. Drugim riječima, autobus može posjetiti samo uzastopni podniz stanica. Pritpom je bitno da autobus ne premaši kapacitet. Za svaki uzastopni podniz jednostavno je odrediti njegovu cijenu.

Ovo se može rješiti na optimalan način koristeći dinamičko programiranje. Rekurzivna relacija dana je u nastavku:

$dp(i) = max(dp(i), dp(i-k) + C(i-k+1, i))$

za svaki k >= 1 za koji autobus ne premašuje kapacitet. $C(i, j)$ je cijena obilaska uzastopnih stanica od indexa $i$ do $j$ jednom autobusom.

# 3. Rezultati

| Instanca | Ukupan put |
|---|---|
| 1 | 140.44 |
| 2 | 101.06 |
| 3 | 2587.08 |
| 4 | 1555.28 |
| 5 | 1864.46 |
| 6 | 1223.03 |
| 7 | 1129.24 |
| 8 | 632.51 |
| 9 | 324.29 |
| 10 | 135.80 |

# 4. Zaključak

Prethodno opisani algoritam vrlo brzo dolazi do pontecijalno kvalitetnih rješenja. U sljedećim koracima je potrebno usporediti ta rješenja sa state of the art algoritmima.

Dodatno poboljšanje moguće je ubrzanjem vremena izvođenja što se može postići parelelizacijom samog algoritma. Moguće je i na jednostavan način pokrenuti više instanca istog algoritma. Ono što se još čini prikladno je napraviti nekakvu metaheuristiku na najvišoj tj. prvoj razini algoritma umjesto pohlepnog odabira stanica ili barem metaheuristika nad tim pohlepnim početnim rješenjima.