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 učenici 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

Pri rješavanju ovog problema iskušano je nekoliko različitih pristupa i algoritama, i izrađena su dva sasvim odvojena rješenja, a sveukupno najboljim se pokazalo rješenje opisano u nastavku.

Konačni algoritam, po uzoru i na strukturu problema, je slojevit, i sastoji se od dvije razine. Na prvoj se za svakog učenika odabire odgovarajuća autobusna stanica, ali tako da dobiveni raspored zadovolji sva ograničenja problema. Na drugoj se pak razini za odabrani raspored učenika po stanicama određuje optimalan redoslijed autobusnih ruta koje će posjetiti sve odabrane stanice, i to se vrši u tri koraka. Prvo se odabire neki inicijalni redoslijed posjećivanja stanica, potom se on lokalnim pretraživanjem pokušava što više poboljšati, a konačno se optimalan raspored autobusa po odabranim stanicama određuje pomoću dinamičkog programiranja.

Algoritam pritom (u odabiru redoslijeda posjećivanja stanica) ne razmatra stanice koje su ispunjene do ruba kapaciteta (tj. maksimalnog kapaciteta autobusa), već njih samo na kraju dodaje u rješenje. Ako je u rješenju nakon prve razine prisutno manje od 12 stanica koje nisu popunjene do kraja, umjesto lokalne pretrage provodi se iscrpno pretraživanje svih mogućih permutacija obilazaka stanica.

U nastavku su pobliže opisani struktura i rad samog algoritma, svaki korak u svom potpoglavlju.

2.1. Pridruživanje autobusnih stanica učenicima

U stvarnosti se pokazalo kako je često dobro rješenje optimizacijskih problema započeti pohlepnim pristupom, i potom to pohlepno rješenje poboljšati nekom metodom usmjerene pretrage. Stoga je i u ovom algoritmu prvi korak, pridjeljivanje stanica učenicima, učinjen pohlepnim pristupom.

Dva su kriterija kojima je to pridruživanje bilo vođeno. Prvi je kriterij bio

da se u rješenju odabere što manje autobusnih stanica, a drugi da su odabrane stanice čim bliže školi. Razlozi za to su vrlo jednostavn. Ako nam je cilj minimizirati ukupnu prijeđenu udaljenost svih autobusa, očito je isplativije da što veći dio puta od svoga doma do škole učenici prijeđu pješice, tj. da ih autobus pokupi na stanici koja je najbliža školi. Također, što je manje stanica koje autobusi moraju posjetiti, za očekivati je da će i njihov ukupno prijeđeni put time biti manji.

Algoritam opisane kriterije u izgradnji rasporeda učenika i stanica koristi na sljedeći način: počevši od stanica najbližih školi, pokušava se pronaći ona stanica koju može posjetiti najveći broj učenika, i kada se takva stanica pronađe, obavi se pridruživanja učenika stanici. Ako neku stanicu može posjetiti više učenika nego li je njezin kapacitet, na slučajan se način izabire kojim će učenicima ona dodijeliti, što omogućuje da se u više pokretanja algoritma rješenja međusobno razlikuju.

Kako je konačni algoritam vrlo brz, bilo ga je moguće pokretati mnogo puta za različite rasporede učenika po stanicama, pa je stoga u opisan pohlepni pristup raspoređivanju uveden i šum, koji uz određenu vjerojatnost ponekad preskače neku od bližih stanica i odabire dalju, čime se povećava broj ostvarivih početnih rasporeda, a tako i proširuje područje pretrage.

Drugi pokušaji izgradnje početnog rješenja, a iskušano ih je nekoliko, nisu se pokazali mjerljivima s prethodno opisanim pristupom, te su stoga napušteni.

2.2. Odabir inicijalnog redosljeda posjećivanja stanica

Za prethodno dobiveni raspored učenika po stanicama u ovom se koraku određuje preliminaran redoslijed kojim će se posjećivati sve aktivne stanice (one na kojima čeka barem jedan učenik).

Odmah u početku, algoritam provjerava postoje li stanice koje su napunjene do maksimalnog kapaciteta. Sve takve stanice sigurno će biti posjećene odvojeno od svih ostalih (zasebnom autobusnom linijom), te se stoga ne razmatraju u nastavku algoritma, već se samo na kraju pridodaju rješenju. Time se smanjuje trajanje jednog izvođenja algoritma, te se to dobiveno vrijeme može korisnije utrošiti na pretragu šireg prostora ili višestruko pokretanje algoritma.

One pak stanice koje nisu ispunjene do maksimalnog kapaciteta sortiraju se prema kutu u odnosu na školu kao ishodište – jer je vjerojatnije da će se stanice sa bliskim središnjim kutem naći na ruti istog autobusa – i tako sortiran poredak stanica predstavlja inicijalno rješenje obilaska. No kako najbolje rješenje ne mora nužno krenuti baš od stanice s najmanjom vrijednošću središnjeg kuta (tj. prvom u tom poretku), ne uzima se samo jedno inicijalno rješenje, već njih N, tj. sve moguće kružne rotacije tog poretka.

2.3. Lokalna ili optimalna pretraga

Prethodno ostvareni inicijalni poredak u ovom se koraku optimizira lokalnim ili iscrpnim pretraživanjem, ovisno o broju stanica. Ako ih je manje od 11, iscrpno se iskušavaju sve permutacije i odabire ona koja ima najbolju vrijednost, dok se za više od 11 stanica provodi lokalno pretraživanje koje kao operator susjedstva iskušava sve zamjene parova bliskih stanica u poretku. Kao funkcija prikladnosti koristi se ukupna udaljenost potrebna za pohlepno posjećivanje stanica, koje se vrši na način da se, počevši od prve, stanice posjećuju jedna za drugom sve dok se tako ne premaši kapacitet autobusa, kada se autobus vrati do škole, isprazni, i nastavi dalje sa prvom idućom stanicom.

2.4. Dinamičko programiranje

I na kraju, za najbolje rješenje ostvareno prethodnim korakom optimalan redoslijed posjećivanja određuje se pomoću dinamičkog programiranja. Konkretno, u ovom se koraku određuje u kojim se sve trenutcima autobus treba vraćati do škole kako bi ostavio ukrcane učenike i potom prazan nastavio dalje sa svojim obilaskom. Drugim riječima, za u prethodnom koraku odabrani redoslijed posjećivanja stanica, ovaj korak odlučuje koji je optimalni rastav tog redoslijeda na uzastopne podnizove koje će autobus posjetiti u jednoj turi, a da pritom ne premaši svoj dopušteni kapacitet.

Ta se odluka može donijeti optimalno pomoću dinamičkog programiranja, koristeći sljedeću rekurzivnu relaciju:

$$DP[i] = \max(DP[i-k] + C[i-k+1, i])$$

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

Algorithm 1 Pridruživanje autobusnih stanica učenicima

function (*students*, *stops*)
  $studentStopAssignment \leftarrow \{\varnothing\}$
  while $\exists$ *student without assigned stop* **do**
    $bestStop \leftarrow \arg\max_{stop}(no.\ of\ possible\ students\ assigned\ to\ stop)$
    if $|bestStop| > 1$ **then** // more than one best stop
      $bestStop \leftarrow ClosestToSchool(bestStop)$
    end if
    $studentStopAssignment \leftarrow$
      $AssignPotentialStudents(bestStop, studentStopAssignment)$
  end while
  $stopsToVisit \leftarrow \{stop : AssignedStudents(stop) \neq \varnothing\}$
  $bestSolution \leftarrow InitialOrderAndOptimize(stopsToVisit)$
  return *bestSolution*
end function

Algorithm 2 Odabir inicijalnog redoslijeda posjećivanja

function INITIALORDERANDOPTIMIZE(*stopsToVisit*)
  $stopsToVisit' \leftarrow stopsToVisit \setminus \{stopsFilledToCapacity\}$
  $initialOrder \leftarrow SortByAngle(stopsToVisit')$
  $optimalOrder' \leftarrow SearchOptimize(initialOrder)$
  $optimalOrder \leftarrow optimalOrder' \cup \{stopsFilledToCapacity\}$
  return *optimalOrder*
end function

Algorithm 3 Lokalna ili optimalna pretraga

function SEARCHOPTIMIZE(*initialOrder*)
  if $|initialOrder| < 12$ **then**
    $bestOrder \leftarrow TryAllPermutations(initialOrder)$
  else
    $bestOrder \leftarrow LocalSearch(initialOrder, neighborhood = swapNearby)$
  end if
  $bestSolution \leftarrow DynamicProgrammingOptimize(bestSolution)$
  return *bestSolution*
end function

3. Rezultati

U tablici 3.1 prikazani su svi najbolji ostvareni rezultati po pojedinim instancama problema.

Tablica 3.1: Najbolji ostvareni rezultati za pojedine instance problema

Instanca	Ukupan put
1	140.4434
2	101.0653
3	2587.0818
4	1544.3800
5	1863.9424
6	1217.2701
7	1129.2498
8	632.5131
9	324.2942
10	135.2466

Kao što je vidljivo, instance 1, 2, 7, 8 i 9 razlikuju se od ostalih po značajno manjim vrijednostima ostvarenih rješenja, a razlog tome je drugačija konfiguracija instanci. Naime, u tim su instancama gotovo svi učenici u dosegu svih stanica, pa je većinu učenika moguće smjestiti na stanice koje se nalaze tik do škole, a te školi bliske stanice najčešće bivaju i potpuno ispunjene, tako da algoritamu preostaje odrediti redoslijed posjećivanja tek malog broja nesasvim-popunjenih stanica (njih manje od 12), tako da to čini koristeći iscrpno pretraživanje.

Nasuprot tome, primjeri $3-7$ bitno su veći po pitanju broja stanica koje treba optimalno poredati, pa se na njima primjenjuje lokalno pretraživanje, a i

sam se algoritam pokreće više puta (jer zbog šuma daje različite početne rasporede, a zbog brzine se stigne izvesti mnogo puta unutar zadanog vremena), što bitno doprinosi kvaliteti ostvarenih rezultata.

Uz ovaj ovdje opisani algoritam, postojao je još jedan drugi, neovisni, s bitno drugačijim pristupom koji je uključivao i genetske algoritme, ali usporedbom rezultata oba rješenja pokazalo se da ovaj algoritam postiže značajno bolja rješenja od drugoga, te je drugi na kraju napušten.

4. Zaključak

Na temelju ostvarenih rezultata i uočenih performansi, zaključeno je da ovaj algoritam vrlo brzo pronalazi potencijalno vrlo kvalitetna rješenja. Idući bi korak u njegovoj evaluaciji bila usporedba sa *state-of-the-art* algoritmima za ovaj problem, i tek bi se tada mogao donijeti konačan sud o njegovoj kvaliteti.

Daljnje poboljšanje samog algoritma moguće je ubrzavanjem njegova vremena izvođenja, što se može postići paralelizacijom, bilo unutar samog algoritma, bilo njegovim pokretanjem u više neovisnih instanci. Također, prva razina algoritma (određivanje rasporeda učenika po stanicama) zasada se još uvijek temelji na (randomiziranom) pohlepnom pristupu, pa bi ga vjerojatno bilo moguće zamijeniti nekom metaheuristikom, bilo u potpunosti, bilo da ga ona u sebe uključi i vodi prema stvaranju još boljih rješenja.