STRUKTURE PODATAKA I ALGORITMI

Projekt br. 2: Algoritam 2

Prirodno-matematički fakultet

Ema Mandal
emamandal@hotmail.com

SADRŽAJ

OPIS KODA

IDEJA:

Prilikom implementacije 3-opt algoritma na postojeći kod dodano je 5 novih funkcija: vratiTuruNajmanjaGrana(), Zamjena_twoopt(int v1, int v2, vector<int> tura), vratiTwoOptTuru(), two_opt(), three_opt(). Funkcija vratiTuruNajmanjaGrana je ista kao i funkcija turaNajmanjaGrana, samo što umjesto dužine puta vraća red obilaska gradova. Ova funkcija je potrebna da bismo je pomoću two_opt() funkcije mogli optimizirati, to jeste naći kraći put. Ideja je da pomoću dvije for petlje prođemo kroz sve moguće kombinacije od dva grada i mijenjamo dvije grane u trenutnom grafu. Mijenjanje se vrši tako što zamijenimo red gradova koji se nalaze između ona dva grada nad kojima je pozvana zamjena. Zatim računamo dužinu nove ture i ako je kraća, postavljamo je da bude naša nova tura. Proces se izvršava sve dok je moguće optimizirati trenutnu turu.

Što se tiče implementacije three_opt algoritma koji mijenja tri grane, ona je veoma slična implementaciji two_opt algoritma. Ali u ovom slučaju imamo tri petlje te mijenjamo grane između tri čvora. Mijenjanje vršimo pomoću iste funkcije koju smo koristili i za mijenjanje dvije grane, i to prvo mijenjamo dvije grane između i-tog i k-tog grada, a zatim između j-tog i k-tog grada, gdje je i<j<k.

FUNKCIJE:

`vector<int> vratiTuruNajmanjaGrana()` – kao što je već objašnjeno, funkcija je ista kao i turaNajmanjaGrana() samo što umjesto dužine ture vraća red obilaska gradova

`vector<int> Zamjena_twoopt( int v1, int v2, vector<int> tura )` – funkcija prvo kopira sve gradove od 0 do v1 (koji je proslijeđen) u novi vektor nazvan ruta. Zatim kopira gradove između v1 i v2 u obrnutom poretku, te nastavlja kopiranje gradova od v2 + 1 do kraja. Zaista ovakav način zamjene mijenja dvije grane u grafu. Na kraju se vraća vector<int> ruta nad kojim je izvršena zamjena.

`int two_opt()` – funkcija koja mijenjajući dvije grane optimizuje trenutnu rutu i vraća dužinu nove rute. Najprije se poziva funkcija koja vraća turu najmanja grana i zatim tu turu optimizujemo tako što pomoću dvije for petlje prolazimo između svaka dva čvora i mijenjamo dvije grane u trenutnom grafu. Prilikom svake zamjene poredi se nova dužina rute sa trenutnom dužinom, te ukoliko je nova dužina manja, trenutna tura postaje nova tura. Ovo se izvršava sve dok se mogu zamijeniti dvije grane tako da put bude kraći. Da bismo znali kada će funkcija stati, na početku inicijalizujemo brojač na 0. On se povećava samo ako se trenutna ruta poboljšala. Na kraju ako je taj brojač jednak 0 to znači da trenutnu rutu nismo mogli poboljšati pa se prekida while petlja i vraća se trenutna dužina rute koja je i ujedno i 2opt optimalna.

`vector<int> vratiTwoOptTuru()` – funkcija radi isto kao i prethodno objašnjena two_opt() samo što umjesto najbolje udaljenosti vraća vektor najboljeg poretka gradova.

`int three_opt()` – funkcija radi slično kao i two_opt. Međutim ona optimizuje turu koja je dobivena primjenjujući two_opt algoritam. Ona prolazi kroz trenutnu turu i pomoću tri for petlje mijenja tri grane u

trenutnom grafu. Mijenjanje se vrši pomoću prethodno objašnjene funkcije Zamjena_twoopt tako što mijenjamo poredak gradova izmeću i-tog i k-tog grada, a zatim između j-tog i k-tog grada, pri čemu je i<j<k. Isto kao i kod two_opt algoritma ukoliko je nova tura bolja onda ona postaje trenutna tura. Algoritam staje kada je nemoguće poboljšati trenutnu turu. Zaustavljanje algoritma obezbjeđeno je pomoću brojača koji se povećava onda kada je tura optimizovana. Ukoliko je taj brojač jednak 0 to znači da se tura nije mogla poboljšati i prekida se izvršavanje while petlje.

ANALIZA OPTIMALNOSTI

Nakon testiranja algoritama, dobiju se sljedeći rezultati:

IME DRŽAVE	TURA NAJBLIŽI SUSJED	TURA NAJMANJA GRANA	TRENUTNO OPTIMALNO RJEŠENJE	2OPT RJEŠENJE	RAZLIKA OD TRENUTNOG OPTIMALNOG	3OPT RJEŠENJE	RAZLIKA OD TRENUTNOG OPTIMALNOG
DJIBOUTI	9745	7019	6656	6660	0,06%	6660	0,06%
WESTERN SAHARA	36388	39691	27603	28804	4,35%	28804	4,35%
QATAR	11640	11499	9352	10331	10,47%	10143	8,46%
LUXEMBURG	14370	13464	11340	11973	5,58%	*	

*Kopmajler ne može izračunati u roku od 3 sata

Procentualna razlika izračunata je kao omjer razlike optimalnog i našeg rješenja podijeljen sa optimalnim rješenjem i na kraju cijeli taj broj je pomnožen sa 100.

Vidimo da za neke države 2opt i 3opt algoritmi daju rješenje koje je bliže optimalnom ili čak optimalno, dok je za neke razlika rješenja dosta veća. To naravno zavisi i od algortima koji optimizujemo. U ovom slučaju optimizuje se algoritam najmanja grana, koji je sam po sebi u većini slučajeva optimalniji od algoritma najbliži susjed. Naravno, optimalnost ovih algoritama zavisi od države i rasporeda gradova. 3opt algoritam se primjenjuje nad rutom koja je već 2opt optimalna te nju poboljšava. Ukoliko bi dalje nad 3opt optimalnom rutom primjenili 4opt, dobili bismo rutu koja je još približnija trenutnoj optimalnoj. Tj. povećavajući k (broj grana koje se mijenjaju) dobijamo optimalnije rute. Međutim svaka nova zamjena povećava vrijeme izvršavanja tako da je testiranje k-opt algoritama dugotrajan proces te za neke države i za dovoljno veliko k se ne može izvršiti u relanom vremenu.

VRIJEME IZVRŠAVANJA

Tura najbliži susjed – u implementaciji algoritma se koriste 2 for petlje. Prva se izvršava n-1 puta, a druga n puta pa je vrijeme izvršavanja ovog algoritma n*(n-1) što je zapravo kvadratno vrijeme

Tura najmanja grana – for petlja se izvršava n puta, ali unutar nje se provjerava da li ima ciklusa, a ta se funkcija izvršava n puta, tako da se algoritam izvršava u kvadratnom vremenu

2opt algoritam – ovaj algoritam se izvršava u vremenu O($c * n^3$) jer imamo petlju u petlji, prva se izvršava n, dok druga n-i puta, i još se unutar druge petlje poziva funkcija zamjenaTwoOpt koja se izvršava u linearnom vremenu. Sve skupa daje kubno vrijeme izvršavanja, ali treba napomenuti da je sve smješteno unutar jedne while petlje koja se izvršava sve dok je moguće optimizirati trenutnu turu. Pa je zbog toga vrijeme izvršavanja O($c * n^3$), gdje *c* označava broj izvršavanja while petlje.

3opt algoritam – cijeli algoritam je sličan 2opt algoritmu s dodatkom jedne for petlje. Pomoću tri for petlje prolazimo kroz sve čvorove i dva puta mijenjamo dvije grane. Sve se to izvršava unutar while petlje koja je „aktivna“ sve dok se trenutna ruta može poboljšati. Zbog toga imamo vrijeme izvršavanja O($c * n^4$) gdje je *c* broj izvršavanja while petlje.

USMJERENI GRAFOVI

Implementirani algoritmi ne bi radili kod usmjerenih grafova, zbog načina mijenjanja grana. Funkcija ZamjenaTwoOpt je implementirana tako da poreda vrhove između dva proslijeđena vrha u suprotnom smijeru. To bi kod usmjerenog grafa značilo izbacivanje dvije grane i dodavanje dvije nove, ali i mijenjanje smijera još nekim granama čime one postaju druge grane. Dakle mijenjaju se više od 2 (2opt) odnosno 3 (3opt) grane.

Jedna ideja za rješenje ovog problema je da se funkcija Zamjena_twoopt doradi na sljedeći način:

Na kraju trenutnog algoritma za zamjenu dodatno sve grane (i, j) koje se nalaze između vrhova v1 i v2 (koji su proslijeđeni funkciji), ne uljučujući njih, zamijenimo sa granama (j, i).

Ali u ovakvoj implementaciji se može javiti novi problem a to je da jnema puta od jednog vrha do drugog. Dakle moguće je primijeniti zamjenu na određene usmjerene grafove, ali mislim da postoje oni za koje je zamjenu nemoguće implementirati.