Rešavanje dvostepenog problema instalacije neograničenih kapaciteta

Stefan Mišković
Matematički fakultet
Univerzitet u Beogradu

Opis problema

- Neka je *N* skup terminala, *M* skup mogućih lokacija za koncetratore prvog nivoa i *K* skup mogućih lokacija za koncentratore drugog nivoa.
- Za svaki od terminala data je cena pridruživanja datom koncetratoru prvog nivoa, za svaki koncetrator prvog nivoa data je cena njegove instalacije i pridruživanja koncentatoru drugog nivoa, a za svaki koncetrator drugog nivoa data je cena njegove instalacije.
- Potrebno je minimizovati cenu pridruživanja svih terminala nekim koncetratorima prvog, odnosno drugog nivoa.
- Slikovitiji primer: Prenošenje robe iz kineskih gradova do mesta u SAD preko evropskih prestonica.

Sličan problem

- Razmotrimo jedan sličan problem. Neka je dato *N* terminala koje treba povezati sa nekim od *M* mogućih lokacija koncentratora.
- Ako je data cena pridruživanja svakod terminala svakom koncetratoru i cena instalacije svakog koncetratora, minimizovati ukupnu vrednost pridruživanja terminala nekim koncetratorima i instalacije tih koncetratora.
- Ovaj problem je veoma poznat i široko proučavan, pa su za njega razvjene brojne heuristike koje mogu dati zadovoljavajuće rešenje i za nešto veće ulazne instance.

Matematička formulacija

- Uvedimo sledeće oznake:
  - C_{ij} – cena pridruživanja terminala i koncetratoru prvog reda na lokaciji j
  - B_{jk} – suma cene postavljanja koncentratora prvog reda na lokaciji j i cene njegovog pridruživanja koncentratoru drugog reda na lokaci
  - F_k – cena postavljanja koncetratora drugog reda na lokaciju k
  - $x_{ij} = 1$ ako je terminal i dodeljen koncetratoru prvog reda j, $x_{ij} =$ 0 inače
  - $y_{jk} = 1$ ako je koncentrator prvog reda j dodeljen koncetratoru drugog reda k, $y_{jk} = 0$ inače
  - $z_k = 0$ ako je koncentator drugog reda k instaliran, $z_k = 0$ inače

- Potrebno je minimizovati funkciju koja je obika

$$\sum_{i\in N}\sum_{j\in M} C_{ij}x_{ij} + \sum_{j\in M}\sum_{k\in K} B_{jk}y_{jk} + \sum_{k\in K} F_k z_k$$

imajući pritom u vidu sledeće uslove:

$$(\forall i \in N)\sum_{j\in M} x_{ij} = 1$$

$$(\forall j \in M)\sum_{k\in K} y_{jk} \leq 1$$

$$(\forall i \in N)(\forall j \in M)x_{ij} \leq \sum_{j\in M} y_{jk}$$

$$(\forall j \in M)(\forall k \in K)y_{jk} \leq z_k$$

$$(\forall i \in N)(\forall j \in M)x_{ij} \in \{0,1\}$$

$$(\forall j \in M)(\forall k \in K)y_{jk} \in \{0,1\}$$

$$(\forall k \in K)z_k \in \{0,1\}$$

Rešenje preko brute-force algoritma

- Prvi, naivan, pristup ovom problemu bi se mogao zasnivati na algoritmu koji koristi grubu silu.
- Za N terminala koncetratore prvog reda je moguće odabrati na 2^M načina. Pritom ćemo sve kombinacije isprobati i izračunati njihovu cenu. Zatim ćemo te odabrane koncetratore pridružiti koncetratorima drugog reda koji se mogu izabrati na 2^K načina. Jasno je da je složenost ovog algoritma eksponencijalna u zavisnosti od M, N i K.

- Dvostepeni problem instalacije neograničenih kapaciteta je NP-težak, pa egzaktni algoritmi, iako daju uvek tačno rešenje su za veće ulazne instance izuzetno spori.
- Zbog toga se koriste razne heuristike, a ovde je prikazano rešenje koji koristi genetski algoritam.

Genetski algoritmi

- Za idejnog tvorca genetskih algoritama se uzima Džon Holand. Nastali su ranih sedamdesetih godina 20. veka
- Oni su u potpunosti bazirani na Darvinovoj teoriji evolucije. Ideja je da se u potpunosti imitira prirodni proces prilagođavanja.
- Jedinke unutar jedne populacije se stalno nadmeću za resurse, a one koje su u većoj meri prilagođene okruženju znatno češće preživljavaju i imaju veći uticaj na formiranje novih generacija.
- Svaka jedinka nove generacije nastaje kombinovanjem genetskog materijala svojih roditelja. Pritom ponekad može doći do slučajne promene genetskog materijala.

- Osnovna konstrukcija je *populacija jedinki*.
- Svaka jedinka predstavlja moguće rešenje u *pretraživačkom prostoru* za dati probem (prostoru svih rešenja) i predstavljena je *genetskim kodom* nad nekom azbukom simbola.
- Često se koristi *binarno kodiranje*, gde se genetski kôd sastoji od niza bitova.
- Svakoj jedinki se dodeljuje *funkcija prilagođenosti* koja ocenjuje njen kvalitet.
- Bolje prilagođavanje svake jedinke i populacije u celini se postiže uzastopnom primenom *selekcije*, *ukrštanja* i *mutacije*.
- Nekad je prisutan i *elitizam* – direktno prenošenje jedinki u novu populaciju.

Selekcija i ukrštanje

- Selekcija se kod genetskih algoritama zasniva na prirodnoj selekciji – favorizuju se natprosečno prilagođene jedinke pri formiranju nove generacije.
- Operator ukrštanja vrši rekombinaciju gena jedinki i time doprinosi raznovrsnosti genetskog materijala. Rezultat ukrštanja je nedeterministička razmena genet-skog materijala između jedinki, sa mogućnošću da dobro prilagođene jedinke generišu još bolje jedinke.
- U osnovnoj verziji algoritma se na slučajan način bira tačka ukrštanja, tj. pozicija u kodovima roditelja u udnosu na koji će simboli u ta dva koda razmeniti mesta. Operator ukrštanja može biti definisan sa dve ili više takača ukrštanja.

Ukrštanje (jedna tačka ukrštanja)

Ukrštanje (dve tačke ukrštanja)

Mutacija

- Višestrukom primenom selekcije i ukrštanja moguće je gubljenje genetskog materijala, odnosno postaju nedostupni neki regioni pretraživačkog prostora.
- Mutacija vrši slučajnu promenu određenog gena, sa datom malom verovatnoćom p_m, čime je moguće vraćanje izgubljenog genetskog materijala u populaciju.
- To je osnovni mehanizam za sprečavanje preuranjene konvergencije genetskog algoritma u lokalnom ekstremu.

Pseudokod jednostavnog GA

```
ucitavanje_ulaznih_podataka();
inicijalizacija_populacije();
while (!kriterijum_zavrsetka())
{
  for (i = 0; i < br_pop; i++)
    p[i] = vrednosna_funkcija();
  funkcija_prilagodjenosti();
  selekcija();
  ukrstanje();
  mutacija();
}
stampanje_izlaznih_podataka();
```

Turnirska selekcija

- U turnirskoj selekciji, svaki element populacije se bira za prelazak u narednu generaciju ukoliko ima bolju funkciju prilagođenosti od nekoliko ostalih proizvoljno odabranih elemenata populacije.
- Parametar selekcije je veličina turnira, označimo je sa *N*. Za *N* se uvek uzima pozitivan ceo broj.
- Turnirska selekcija je jedan od najpopularnijih načina selekcije. Razvitkom paralelnih genetskih algoritama, njena popularnost je u poslednjem periodu još više izražena.

Pseudokod turnirske selekcije

Ulaz:
populacija A (velicine br_pop)
velicina turnira N

Izlaz:
populacija A' (velicine br_pop)

Algoritam:
for (i=0; i<br_pop; i++)
  A'[i] = element populacije A koji ima
        bolju funkciju prilagođenosti
        medju njenih N proizvoljno
        odabranih elemenata;
return A';

Fino graduirana turnirska selekcija

- Izvršavanjem prethodnog algoritma se često dešava da se sâm proces odvija veoma sporo ili pak dolazi do prerane konvergencije. Zato se uvodi tzv. fino graduirana turnirska selekcija. Klasična turnirska selekcija je njen specijalan slučaj.
- Umesto celobrojnog parametra *N* (veličina turnira) kori-sti se realno vrednosti parametar *F* – željena prosečna veličina turnira. Od ovog parametra zavisi proces selekcije, pa bi prosečna veličina turnira u populaciji trebalo da bude što bliža njegovoj vrednosti.
- Pri implementaciji razlike veličinâ turnirâ su manje ili jednake 1 i izabrane su tako da je njihova prosečna veličina što bliža parametru *F*.

Pseudokod fino graduirane turnirske selekcije

- **Ulaz:** populacija A (velicine br_pop), zeljena prosecna velicina turnira F
- **Izlaz:** populacija A' (velicine br_pop)
- **Algoritam:**

```
F_manje = trunc(F);
F_vece = trunc(F) + 1;
sr_pop = trunc(n * (1 – frac(F)));
for (i = 0; i < sr_pop; i++)
   A'[i] = element populacije A koji ima bolju funkciju prilagođenosti medju
           njenih F_manje proizvoljno  odabranih elemenata;
for (i=sr_pop; i<br_pop; i++)
   A'[i] = element populacije A koji ima bolju funkciju prilagođenosti  medju
           njenih F_vece proizvoljno  odabranih elemenata;
return A';
```

Primena opisanog GA na naš problem

- Za kodiranje promenljivih iskorišćen je binarni kôd, koji se za ovakve vrste problema donekle prirodno nameće.
- Inicijalizacija populacije se generiše na dva načina – pohlepnim algoritmom ili na sasvim slučajan način. Verovatnoća odabiranja jednog načina je 0.5.
- Uzeto je da je broj elemenata populacije 300, a broj generacija 750(M+N+K).
- Korišćen je elitizam, pri čemu je 3/5 jedinki preneto direktno u narednu generaciju.

- Preostale 2/5 nove generacije su dobjene operatorom ukrštanja. Za taj operator, roditelji za dato dete se biraju nasumično. Kao optimalni broj tačaka permutacije uzeta je vrednost *trunc(sqrt(N))*, gde je N broj terminala.
- Za operator selekcije je korišćena turnirska selekcija. Testirane su dve verzije algoritma – obična turnirska selekcija sa parametrom $N = 6$ i fino graduirana turnirska selekcija sa parametrom $F = 7.3$.
- Kod ukrštanja od roditelja nasleđuje za date terminale u potpunosti i koncetratori oba reda. Zbog toga se mutacijom menja jedino način pridruživanja koncetratora prvog nivoa koncetratorima drugog. Za mutaciju je uzeta verovatnoća $p_m = 0.2$.

Poređenje rezultata za CPLEX, GA1 i GA2

Ulaz	N	M	K	Rezultat	CPLEX	GA 1	GA 2
1.txt	3	3	3	122.000	0.016	0.001	0.001
2.txt	3	4	3	5.000	0.020	0.001	0.001
3.txt	3	4	5	5.000	0.018	0.001	0.001
4.txt	7	8	9	17.000	0.778	0.004	0.004
5.txt	8	8	8	144.000	0.566	0.004	0.004
6.txt	10	10	12	1123.000	1.101	0.007	0.007
7.txt	15	15	15	32.000	1.304	0.015	0.015
8.txt	19	19	17	376.000	3.309	0.020	0.021
9.txt	20	20	20	55.000	7.018	0.018	0.017
10.txt	25	25	25	1246.000	15.066	0.029	0.033
11.txt	50	50	50	15349.000	> 60.000	0.525	0.521
12.txt	100	100	100	27005.000	>60.000	2.139	2.155

Poređenje rezultata genetskih algoritama koji koriste običnu (GA1) i fino graduiranu turnirsku selekciju (GA2)

Ulaz	N	M	K	CPLEX	r(GA1)	t(GA1)	r(GA2)	t(GA2)
101.txt	150	150	150	-	36176	2.234	36132	2.212
102.txt	200	150	100	-	77356	3.755	77293	3.993
103.txt	200	200	200	-	84004	7.196	83918	8.033
104.txt	500	500	500	-	213755	21.433	212938	20.198
105.txt	500	500	500	-	198904	20.313	196233	20.107
106.txt	1000	1000	1000	-	513992	63.913	509359	66.318
107.txt	1000	1000	1000	-	493213	74.437	489744	69.536

Analiza rezultata

- U prvoj tabeli prikazano je poređenje rezultata dobjenih preko CPLEX-a i genetskih algoritama koji koriste običnu i fino graduiranu turnirsku selekciju. Vidimo da se za manje ulazne istance svi rezultati poklapaju, dok se veće ulazne instance CPLEX i nije u mogućnosti da u realnom vremenu reši problem.
- U drugoj tabeli su upoređeni GA1 i GA2. CPLEX ovoga puta nije razmatran, zbog njegove nemogućnosti da generiše rezultat. Na ovim primerima vidimo da je vreme izvršavanja približno isto (jer je i slična implementacija), dok GA2 prikazuje bolje preformanse od GA1 pri generisanju rezultata.