Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

PROJEKT

iz predmeta

Heurističke metode optimizacija

PROBLEM RASPOREĐIVANJA TESTOVA

Željko Baranek

Zagreb, siječanj 2018.

Problem

Problem raspoređivanja testova je problem iz skupine NP-teških problema raspoređivanja (engl. *scheduling problem*). Obzirom na ogroman prostor pretraživanja, nemogućnost određivanja egzaktnog algoritma koji bi dao optimalno rješenje i složenosti dodatnih ograničenja koje otežavaju pronalazak zadovoljivih rješenja, pogodi su za primjenu heuristika koje bi usmjerile pretragu.

U zadanom problemu postoji:

- N testova (t_i, d)
  - naziv testa t_i , i element od 1
  - njegovo cjelobrojno trajanje d, d >= 1
  - Lista strojeva na kojemu se test može izvoditi
  - Lista resursa koje test zahtijeva tijekom svog izvođenja
- M strojeva s oznakom m_i, i element od 1 do M
- R resursa i njihova brojnost s oznakom (r_i, n) i element od 1 do R, n >= 1

Želimo rasporediti izvođenje testova na strojeve u vremenu, tako da je razlika vremena od početka izvođenja prvog testa do završetka izvođenja zadnjeg testa minimalna, uz zadovoljenje uvjeta da se testovi mogu izvršavati samo na jednom stroju iz definirane liste strojeva na kojima se mogu izvršavati te da tijekom cijelog izvođenja testa na raspolaganju moraju biti resursi koje izvođenje testa zahtijeva.

Primijenjeni algoritam

Algoritam se može nazvati konstruktivnim naivnim, odnosno pohlepnim algoritmom koji prvo rasporedi testove koji se mogu izvoditi samo na jednom stroju na prvo zadovoljivo mjesto, a zatim nasumičnim redoslijedom odabire preostale testove i njih također raspoređuje na prvo slobodno zadovoljivo mjesto na nasumično odabran stroj na kojem se može izvoditi.

Strukture podataka

Nakon uspješnog parsiranja ulazne datoteke, algoritam ima sljedeće strukture podataka fiksne tijekom cijelog svog izvođenja:

- Rječnik testova, gdje je ključ naziv testa, a vrijednost razred Test koji ima atribute:
  - Trajanje testa (cjelobrojna vrijednost)
  - Lista strojeva na kojima se može izvršavati
  - Lista resursa koje zahtijeva
- Skup strojeva
- Rječnik resursa gdje je ključ naziv resursa, a vrijednost broj koliko takvih resursa ukupno ima na raspolaganju

Za svoj rad, algoritam koristi sljedeće dodatne strukture podataka:

- Lista testova koji još nisu raspoređeni
- Rječnik rasporeda po strojevima kroz vrijeme, gdje su ključevi strojevi, a vrijednosti polja naziva testova koji se tada izvode, gdje se indeks tog polja smatra vremenskim trenutkom
- Rječnik dostupnih resursa kroz vrijeme, gdje je ključ naziv resursa, a vrijednost polje cjelobrojnih vrijednosti gdje je indeks tog polja vremenski trenutak

Postoje i dvije globale cjelobrojne varijable:

- Maksimalno trajanje – zbroj trajanja svih testova
- Trajanje rasporeda – rješenje

Rad algoritma

Inicijalizacija struktura podataka

Nakon popunjavanja prethodno opisanih fiksnih struktura podataka koje predstavljaju pojedini problem, izračunava se maksimalno moguće trajanje, odnosno zbroj trajanja svih testova, obzirom da je taj podataka potreban za inicijalizaciju daljnjih struktura podataka. Rječnik rasporeda po strojevima popunjava svim strojevima (ključevi), a kao vrijednosti stavlja polja praznih nizova znakova (null-vrijednosti) maksimalnog mogućeg trajanja. Također popunjava rječnik dostupnih resursa po vremenu po istom načelu. Konačno, algoritam u listu testova koji nisu raspoređeni preslikava sve testove nasumičnim redoslijedom te zatim kreće izvođenje.

Izvođenje glavne petlje

Iz popisa neraspoređenih testova prvo se uzima podskup testova čije izvođenje se može ostvariti samo na jednom stroju. Zatim se provodi raspoređivanje preostalih testova, nasumičnim redoslijedom.

Funkcija raspoređivanja

Funkcija prima test i nasumično odabrani stroj na kojem se želi smjestiti test. Zatim krenuvši od nultog trenutka, funkcija poziva pomoćnu funkciju koja prima test, stroj i vremenski trenutak koja provjerava

- Izvodi li se neki drugi test
- Ima li dovoljno resursa koje test zahtijeva

u intervalu od zadanog početnog trenutka pa do tog trenutka uvećanog za trajanje testa. Ako nije pronađen niti jedan konflikt, pomoćna funkcija vraća vrijednost istine te početna funkcija izmjenjuje vremenske rječnike tako da postavlja nazive testova u smješten interval i oduzima broj dostupnih globalnih resursa u smještenom intervalu. Ako pomoćna funkcija pronađe konflikt da se drugi test već izvršava u tom trenutku, dojavljuje početnoj funkciji indeks na kojem treba nastaviti pretragu (uvećan za duljinu izvođenja konfliktnog testa), a ako nema dostupnih globalnih resursa, preskače se samo taj i ide na idući trenutak (jer nije moguće znati kad će biti oslobođen).

Zapis rezultata

Nakon što se isprazni lista neraspoređenih testova, obilazi se rječnik izvođenja po strojevima u vremenu te iz njega iščitavaju počeci izvođenja pojedinog testa. Podaci se zapisuju u listu koja se zatim zapisuje u izlaznu datoteku.

Pseudokôd

```
ulaz = učitaj(ulazna_datoteka)
testovi = parsiraj_testove(ulaz)
strojevi = parsiraj_strojeve(ulaz)
resursi = parsiraj_resurse(ulaz)

max_trajanje = zbroj_trajanja(testovi)
raspored=napravi_rjecnik_polja(kljucevi: strojevi, null polje max_trajanje)
rasp_resursi=napravi_rjecnik(kljucevi: resursi, 0 polje max_trajanje)
nerasp_testovi = testovi.random_redoslijed()

za test iz nerasp_testovi koji se moze izvoditi samo na jednom stroju
      rasporedi(test)
za test iz nerasp_testovi
      rasporedi(test)

ispiši(formatiraj_za_ispis(raspored), izlazna_datoteka)

rasporedi(test)
      za stroj iz test.strojevi
            t = prvi_zadovoljivi_trenutak(test, stroj)
            ako t nije -1
                  za t2 od t do t + test.trajanje
                        raspored[stroj][t2] = test.naziv
                        za resurs iz test.resursi
                              rasp_resursi[resurs]--
                  zaustavi
            inače
                  nastavi za idući stroj

prvi_zadovoljivi_trenutak(test, stroj)
      za t od 0 do t + test.trajanje
            ako je_zadovoljivo(test, stroj, t, referenca t)
                  vrati t
      vrati grešku

je_zadovoljivo(test, stroj, trenutak, referenca idući_t)
      za t od trenutak do t + test.trajanje
            ako raspored[stroj][t] nije slobodan
                  idući_t = testovi[raspored[stroj][t].trajanje + 1
                  vrati laž
            za resurs iz test.resursi
                  ako je rasp_resursi[resurs][t] manji od 1
                        vrati laž
      vrati istinu
```

Dobiveni rezultati

	1 minuta	5 minuta
T1	42504	42886
T2	38786	38505
T3	41475	41192
T4	35966	35966
T5	42933	42669
T6	47918	47685
T7	41984	41709
T8	46060	45385
T9	43443	43243
T10	34009	33655

U tablici su prikazana ukupna vremena izvođenja svih testova (makespan) nakon jedne i pet minuta izvođenja algoritma.

Algoritam za instance problema t1-t7 s po 500 testova za svaku iteraciju treba otprilike 3-6 sekundi, što je vjerojatno i najveća njegova mana. Njegovo dulje izvođenje ne jamči i pronalazak boljeg rješenja, već je to stvar slučajnosti obzirom na to kojim će redoslijedom ići raspoređivati testove i koji će stroj odabrati za smještaj. Ovakav algoritam gotovo pa nikad ne će pronaći optimalno rješenje, ali će uvijek pronaći neko zadovoljivo rješenje. Ne postoje parametri koje bi se moglo podešavati, već bi se morao sam algoritam izmijeniti, ali postoje mjesta na kojima bi minimalna izmjena mogla povoljno utjecati na njegovo ponašanje u određenim problemima. Primjerice, nasumičan redoslijed odabira testova za raspoređivanje te strojeva na kojima će se izvoditi mogao bi se izmijeniti da se nekim testovima i strojevima daje određena prednost, bilo deterministički, bilo i dalje uz određenu slučajnost.

Zaključak

Ovaj algoritam daje izvrsno početno rješenje koje bi se zatim primjenom neke druge heuristike, primjerice, lokalnom pretragom susjedstva, moglo dodatno unaprijediti.