Minimalna Lokalna Aloakacija Registara

Aleksandar Urošević

26. januar 2024.

Sažetak

Ovaj rad istražuje problem minimalne lokalne aloakcije registra, fokusirajući se na optimizaciju alokacije registra unutar blokova koda bez skokova. Cilj je pronaći alokaciju koja minimizuje troškove učitavanja i čuvanja podataka, koristeći ograničeni skup dostupnih registara. Ovaj problem je značajan za optimizaciju kompilatora i interpretera. Prethodni radovi na ovu temu pokazuju da je ovaj problem NP-težak [1]. Postoje razna rešenja ovog problema, poput pretvaranja problema u rešavanje puzla [3]. U ovom radu se pradstavlja rešenje zasnovano na genetskom programiranju.

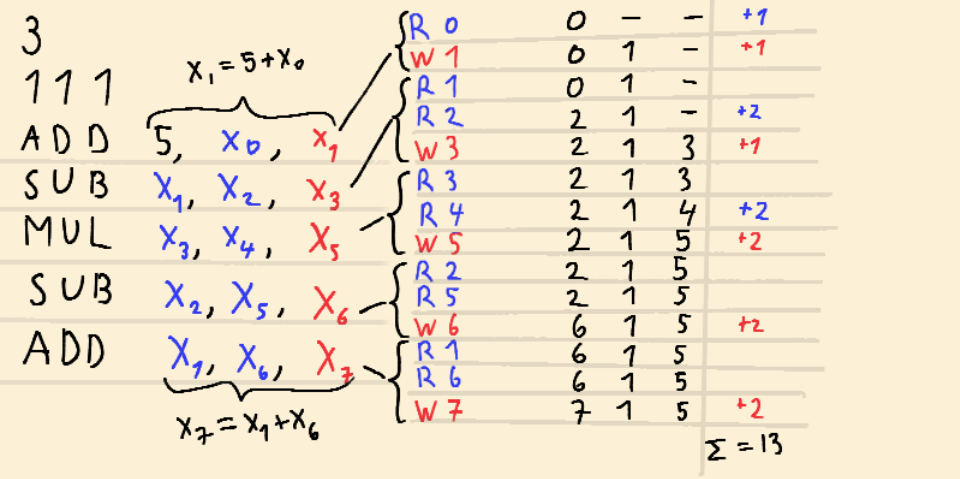
1 Opis problema

Problem o kojem se diskutuje je problem nalaženja najjeftinijeg načina isvršavanja bloka koda (bez skokova). Pretpostavljamo da imamo na raspolaganju N registara, i S_i $(1 \leq i \leq N)$ troškove pri učitavanju ili čuvanja registra i. Za datu sekvencu instrukcija programa treba pronaći optimalnu strategiju učitavanja i čuvanja registra tokom izvršavanja programa.

Sledi opis problema jednim primerom (1). Ovaj program je identican kao primer iz literature [2].

Instrukcija programa $ADD\ x, y, z$ (kao i ostale instrukcije) označava da je neophodno imati promenljive x, y učitanim u registre i neophodno je zapisati novi podatak u z. Ova instrukcija moze da bude prevod linije koda $z = x + y$. Cena učitavanja vrednosti u registar je S_i. Ukoliko su svi registri puni, neophodno je sačuvati vrednost registra i zameniti je novom potrebnom vrednošću, cena ove procedure je $S_i + S_i = 2S_i$, jedno čitanje i jedno pisanje.
U desnoj koloni zapisani su koraci koje bi procesor računara izvrsio, predstavljen je optimalno rešenje dobijeno genetskim algoritmom. Ovaj rezultat je potvrdjeno optimalan putem iscrpne pretrage.

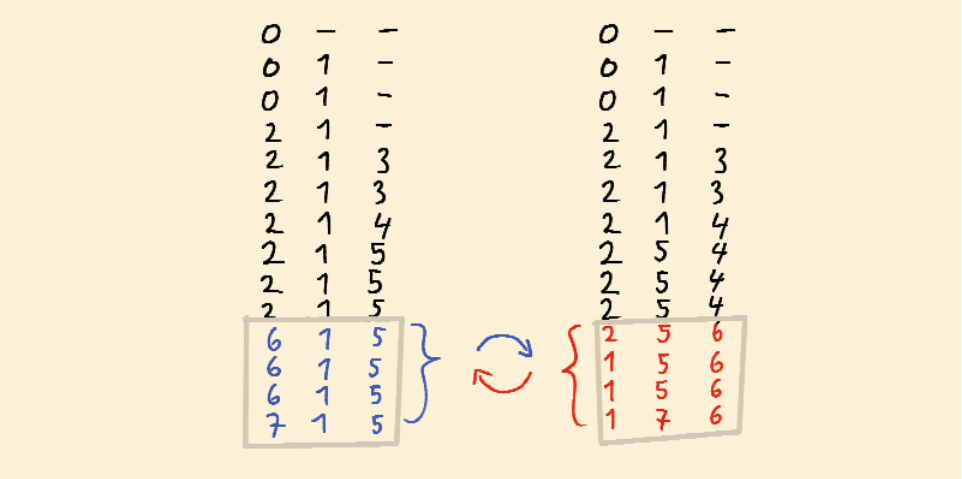
Ilustracija



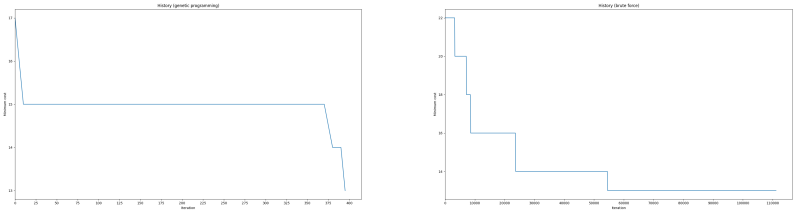
Slika 1: Primer 1

Postoje mnogo ranijih radova na ovom problemu. Od interesanthin radova je „Register Allocation by Puzzle Solving“ gde se diskutuje praktično rešavanje generalnog problema alokacija registra (ne samo lokalnog), i to prevođenjem probema na problem rešavanje puzla[3]. Ovo daje ideju da redove instrukcija mozemo da posmatramo kao blokove koje slažemo. Možemo osmisliti rešenje korišćenjem genetskog programiranja.
Funkcija cilja je suma cena korišćenja registara. Možemo je izračunati prolazeći redom kroz niz alokacija registara rešenja, od prvog reda pa do poslednjeg, posmatrajući kada se i kako registri menjaju.
Ukrštanje implementiramo biranjem nasumičnog broja i td. $0 \leq i < N$, zamenjujemo sve nizove alokacija Aj sa Bj za sve $i \leq j < N$ (2).
Mutacija je implementirana nasumičnim biranjem broja i td. $0 \leq i < N$ koji predstavlja polazni red programa od kojeg nasumično gradimo ostatak niza alokacije.

Ilustracija



Slika 2: Primer 2



(a) genetičko programiranje (b) iscrpna pretraga

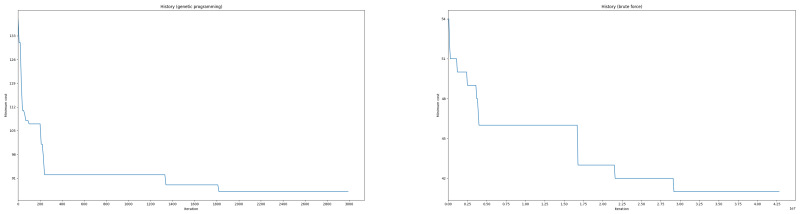
Slika 3: Primena genetičkog programiranja i iscrpne pretrage na primer 1

2 Rezultati

Primenom genetskog programiranja na prvi primer koji je dat i u literaturi [1] dobijamo optimalna rešenja za mali broj iteracija (uz sreću), iscrpa pretraga je takodje prikazana na figuri 3.

Na figuri 4 prikazani su rezultati za veći primer koji se ne nalazi u literaturi.

Figura 6 pokazuje praktični primer. Podaci za ovaj primer su dobijeni korišćenjem programa DUMPBIN i ručnim prevođenjem x64 instrukcija u sintaksu poznatom implementiranom programu.

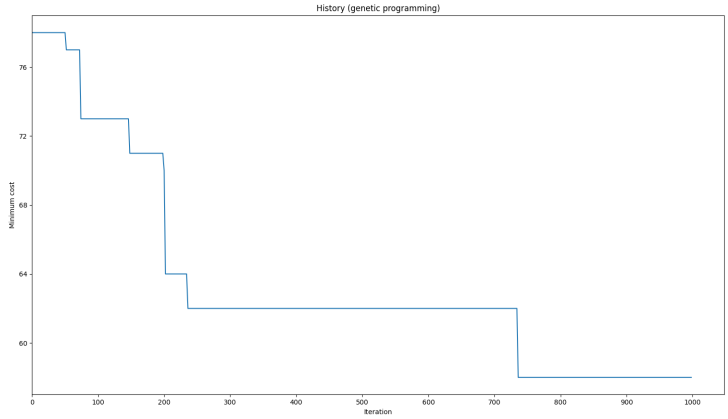


(a) genetičko programiranje (b) iscrpna pretraga

Slika 4: Primena genetičkog programiranja i iscrpne pretrage

```
00007FF62146A42F  mov     eax,4
00007FF62146A434  imul    rax,rax,1
00007FF62146A438  movss   xmm0,dword ptr [rsp+94h]
00007FF62146A441  movss   dword ptr [rsp+rax+160h],xmm0
                  memcpy(&(((struct r_vertex_prop *)model->data)[i*3+(2-
00007FF62146A44A  imul    eax,dword ptr [i],3
00007FF62146A452  mov     ecx,2
00007FF62146A457  sub     ecx,dword ptr [j]
00007FF62146A45E  add     eax,ecx
00007FF62146A460  mov     eax,eax
00007FF62146A462  imul    rax,rax,20h
00007FF62146A466  mov     rcx,qword ptr [model]
00007FF62146A46E  add     rax,qword ptr [rcx+8]
00007FF62146A472  mov     r8d,20h
00007FF62146A478  lea     rdx,[rsp+148h]
00007FF62146A480  mov     rcx,rax
```

Slika 5: DUMPBIN rezultat



Slika 6: Primena genetičkog programiranja na praktični primer

3 Zaključak

Korišćenje genetskog programiranja za nalaženje optimalnog načina alociranja registara možda ne daje rešenja brže od već razvijenih heurističkih algoritama [1] [3]. Razlog je nesigurnost u vremenu koji je potreban da se dostigne dobar rezultat. Ovo se može poboljšati drugim metodama selekcije, mutacije i ukrštanja, ali nesigurnost i dalje ostaje. Razvijena metoda ovde diskutovana daje dobra rešenja, ali sporo u poređenju sa drugim metodama.

Pristup korišćen u ovom radu nije prikladan za x64 arhitekturu procesora zbog postojanja registara koji su proširenja drugih (na primer, registri EAX i RAX).

Literatura

[1] M. Farach and V. Liberatore. On local register allocation. 1998.

[2] Vincenzo Liberatore, Martin Farach-Colton, and Ulrich Kremer. Evaluation of algorithms for local register allocation. 1998.

[3] Fernando Magno Quint˜ao Pereira. and Jens Palsberg. Register allocation by puzzle solving. 2008.