Tema 3 Algoritmi Genetici

Belu Cătălin-Cosmin `Grupa A6`

Hălăucă Andrei `Grupa A1`

January 17, 2019

1 Problemă

Se cere rezolvarea Problemei Comisului Voiajor(TSP) cu o metodă euristică și cu un algoritm genetic. Problema are următoarea cerință:"Dată fiind o listă de orașe și distanțele dintre fiecare două orașe, care este cel mai scurt traseu posibil care vizitează fiecare oraș o singură dată și se întoarce la orașul de origine?", fiind o problemă NP-hard în optimizarea combinatorială. Ca și metodă euristică vom utiliza Hill Climbing Best Improvement.

2 Algoritmul utilizat

2.1 Pseudocod

2.1.1 Algoritm Genetic

t=0
generează P(t)
evaluează P(t)
while $t < NrGen$ **do**
  t=t+1
  SELECTEAZĂ P(t) din P(t-1)
  RECOMBINĂ P(t)
  EVALUEAZĂ P(t)
end while

2.1.2 Hill Climbing Best improvement

t=0
repeat

```
    local=FALSE
    select a candidate solution(bitstring) v_c at random
    evaluate v_c
    repeat
      generate the length strings Hamming-neighbors of v_c
      select v_n, the one among these which gives the largest value for the
      objective function f
      if f(v_c) < f(v_n) then
        v_c = v_n
      else
        local=TRUE
      end if
    until local
    t=t+1
  until t=Max
```

2.2 Detalii de implementare

Pentru reprezentarea algoritmului genetic am folosit reprezentarea cromozomilor ca o mulțime de numere naturale cu valori aleatoare între 0 și NrOrașe. După cum știm, algoritmii genetici sunt de tip probabilist, deci este necesar un generator de numere aleatoare în mai multe etape a algoritmului(generarea populației, selecția indivizilor, alegerea pentru recombinare).

Inițial se va genera o populație de cromozomi, fiecare cromozom fiind format dintr-o mulțime de numere naturale cu valori aleatoare între 0 și NrOrașe, cardinalul acestei mulțimi fiind egal cu NrOrașe.
Folosim, ca și operatori genetici, Selecția, Mutația și Încrucișarea.

Selecția în algoritmul genetic folosit este una probabilistă. Implementarea acesteia ne asigură faptul că, chiar și cel mai slab individ din punct de vedere al fitness-ului are șansă nenulă la supraviețuire. Practic informația specifică a unui individ cu fitness slab poate fi transmisă în generațiile următoare pentru a se regăsi în structura soluției optime. În cadrul implementării noastre am folosit selecția bazată direct pe valorile fitness-ului.

Încrucișarea este operația cu tăiere într-un singur punct(ales aleator) și interschimbarea segmentelor celor doi cromozomi implicați. Mutația este operația de negare a unui bit, ales în mod aleatoriu, din cromozomul implicat.

În algoritmul genetic clasic, funcția de evaluare trebuie să fie strict pozitivă, iar asupra ei să se realizeze maximizare. În cazul în care avem de-a face cu o problemă de minimizare, aceasta poate fi exprimată ca problemă de maximizare prin inversarea funcției fitness.

După cum știm, în proiectarea unui algoritm genetic trebuie luate în calcul valorile unor parametri(dimensiunea populației, probabilitatea de aplicare a mutației, probabilitatea de aplicare a încrucișării etc.). Aceștia au un rol important în rularea algoritmului influențând atât timpul de rulare al algoritmului, cât și rezultatele produse de acesta. Pentru această temă am folosit:

-dimensiunea populației:100
-numărul de generații:1000
-probabilitatea de mutație:0.035
-probabilitatea de încrucișare:0.27
-număr de rulări:30

3 Rezultate experimentale

3.1 Algoritm Genetic

	Minim	Mediu	Maxim	Minim Real
eil51	512	608	874	426

	Minim	Mediu	Maxim	Minim Real
st70	743	894	1328	675

	Minim	Mediu	Maxim	Minim Real
kroA100	22543	26738	33461	21282

3.2 Hill Climbing Best Improvement

	Minim	Mediu	Maxim	Minim Real
eil51	603	753	957	426

	Minim	Mediu	Maxim	Minim Real
st70	859	1047	1684	675

	Minim	Mediu	Maxim	Minim Real
kroA100	25934	29582	37981	21282

4 Comparație între GA și SAHC

După cum se poate observa și din rezultatele de mai sus, cu cât numărul de orașe este mai crescut, cu atât rezultatele sunt mai mari față de minimul real al problemei.

Algoritmul Hill Climbing First Improvement ar trebui, în cele mai multe cazuri, să producă rezultate mult mai apropiate de minimul real al problemei, față de Algoritmul Genetic. În cazul problemei de față, fiind NP-dificilă, Algoritmul Genetic produce rezultate mai bune față de Hill Climbing. Știm că algoritmul Hill Climbing Best Improvement modifică valoarea curentă în timpul căutării doar dacă aceasta este mai mică(mai bună) decât actuala, spre deosebire de First Improvement unde alegem valoarea cea mai apropiată de cea actuală.

Pentru problema TSP, cu cât vom avea mai multe orașe ca input, cu atât rezultatele SAHC se vor depărta mai mult de soluția reală decât rezultatele algoritmului genetic. Astfel, pentru un input de puține instanțe rezultatele celor doi algoritmi sunt apropiate.

Timpul de execuție al algoritmului genetic implementat este însă destul de mare comparativ cu cel al algoritmului Hill Climbing First Improvement, din cauza operatorilor genetici implementați, aceștia crescând considerabil timpul de execuție al programului.