Tema 3 Algoritmi Genetici- Raport

Luca Andrei

December 23, 2016

1 Enunt

Rezolvati problema colorarii grafurilor utilizand un algoritm genetic si o metoda euristica.

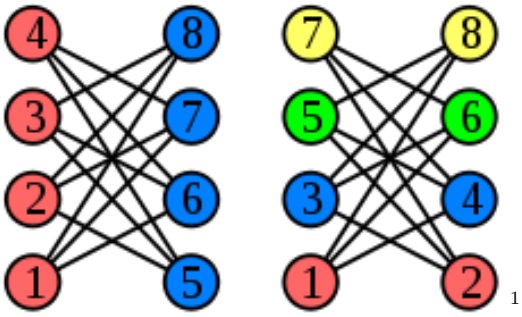
2 Descriere metoda

2.1 Euristica

Idee

Voi folosi un algoritm de tip "greedy" utilizand ordonarea lexicografica a nodurilor. Ideea principala este de a colora in aceasta ordine nodurile alegand prima culoare disponibila (adica prima culoare care nu a fost utilizata in colorarea vecinilor) cu care putem colora acel nod.

Este evident faptul ca algoritmul este dependent de aceasta ordonare pe care o dam nodurilor si figura de mai jos exprima exact acest lucru: in stanga avem o colorare optima(pentru ca am dat o ordonare buna), iar in dreapta avem o colorare neoptima cu numar dublu de culori.



[1]

Pseudocod

```
coloring[1] = 0; // colorez nodul 1 cu culoarea 0
availableColors[0] = false; //culoarea 0 nu mai este disponibila
```

```
availableColors[1] = true; //marchez culoarea 1 ca fiind disponibila
for (i = 2 to numberOfVeticies) {
  coloring[i] = -1; // marchez ca nodurile nu au fost colorate
  availableColors[i] = true; //culorile disponibile
}
for (i = 2 to numberOfVerticies) {
  for (j = 0 to adjacencyList[i].size()) {
    //verific daca nodul vecin adjacenyList[i][j] este colorat
    if (coloring[adjacenyList[i][j]] != -1) {
      // aceasta culoare nu este dispnibila
      availableColors[g.coloring[g.adjacencyList[i][j]]] = false;
    }
  }

  // caut prima culoare disponibila
  for (j = 0 to numberOfVerticies) {
    if (availableColors[j] == true) {
      coloring[i] = j;
    }
  }

  // marchez din nou toate culorile ca fiind nefolosite
  for (j = 0 to numberOfVerticies) {
    availableColors[j] = true;
  }
}
```

2.2 Algoritmul genetic

Pseudocod

```
t = 0;
genereaza(P(t));
evalueaza(P(t));
for (t to numberOfGenerations) {
  selecteaza P(t+1) din P(t)
  aplicaOperatoriiGenetici(P(t))
  evalueaza(P(t));
}
```

Reprezentare

Pentru reprezentare am folosit pentru fiecare cromozom un vector de intregi cu urmatoarea semnificatie: valoarea fiecarei gene reprezinta culoarea cu care va fi colorat nodul din graf cu idicele locusului acelei gene.

Exemplu: coloring = (0, 1, 1, 0) $\Rightarrow$

- Nodul 1: va avea culoarea 0
- Nodul 2: va avea culaorea 1
- Nodul 3: va avea culoarea 1
- Nodul 4: va avea culaorea 0

Fitness

Fitness-ul unui cromozom va fi dat astfel:

$$fitness = \begin{cases} \frac{1}{k*maxC} & k > 0 \\ \frac{1}{maxC} & k = 0 \end{cases}$$

Unde k reprezinta numarul de noduri care au colorari gresite in colorarea curenta, iar $maxC$ numarul culorii maxime din cromozomul respectiv.

Motivul pentru care am ales un astfel de fitness este pentru a favoriza solutiile valide (in care $k = 0$), iar cele invalide sa aiba produsul $k * maxC$ cat mai mic pentru a minimiza atat numarul de nepotriviri, cat si numarul de culori in aceeasi masura. In cazul in care gasesc o solutie valida ($k = 0$), atunci voi incerca sa minimizez numarul de culori folosite, de aici $\frac{1}{maxC}$.

Vor exista discrepante foarte mari intre solutiile valide si cele care nu sunt valide: aprox 0.9. Motivul pentru care fac acest lucru este pentru ca in momentul in care gasesc o solutie valida, vreau sa fie incurajata cu mult fata de cele invalide de a supravietui selectiei(de tip roata norocului).

Selectie

Este menita sa selecteze acei indivizi care ma pot duce spre o solutie optima. In implementarea mea am folosit o selectie de tip "roata norocului" : voi crea un camp de probabilitate astfel incat cromozomii cu fitness mai mare sa aiba o probabilitate mai mare sa fie selectati pentru generatia urmatoare.

Acest lucru se poate realiza creand initial un vector de probabilitati dupa formula $\frac{fitness(i)}{\sum_{j=1}^{nrV} fitness(j)}$ unde i este un cromozom din populatie si apoi generand probabilitatile cumulate pt fiecare cromozom. Astfel, voi selecta cromozomul i daca probabilitatea generata in selectie este intre probabilitatea cumulata pentru cromozomul i si pentru i + 1 $\Rightarrow$ cromozomii cu un fitness mare vor avea o probabilitate mai mare de a fi selectati, deoarece acestia au un interval mai mare de probabilitate(din formula).

Pseudocod:

```
for (i = 0 to popSize)
  eval[i] = fitness(population[i]);
for (i = 0 to popSize)
  S += eval[i];
for (i = 0 to popSize)
```

```
  p[i] += eval[i] / S;
q[0] = 0;
for (i = 0 to popSize)
  q[i + 1] = q[i] + p[i];
for (i = 0 to popSize)
  genereaza prob random r in [0, 1]
  selecteza pentru supravieture individul j
   pentru care q[j] < r <= q[j+1]
```

Mutatia

Stiind ca mutatia in general are un caracter distructiv, am incercat sa o modific pentru a diminua acest caracter astfel:

In loc sa dau o culoare random nodului respectiv fara a verifica deloc ca aceasta schimbare este valida, intai voi parcurge toti vecinii acelui nod si voi marca culorile folosite de acei vecini. Dupa aceasta verificare voi colora nodul respectiv cu o culoare random care nu a fost marcata din intervalul $[0, maxC+1]$ garantand astfel faptul ca acea culoare pe care o folosesc este valida.

Pseudocod

```
for (i = 0 to numberOfVericties) {
  availableColors[i] = true;
}
for (i = 0 to numberOfVerticies) {
  // generez o probabilitate intre [0, 1]
  randomProbability = rand() / RAND_MAX
  if (randomProbability < mutationProbability) {
    // marchez culorile care nu pot fi folosite
    for (j = 0 to adjacencyList[i].size()) {
      availableColors[chromosome[adjacencyList[i][j]]] = false;
    }
    // ii dau genei respective o culoare random care nu a fost marcata
    newColor = rand() % (maxC + 2);
    while (availableColors[newColor] == false) {
      newColor = rand() % (maxC + 2);
    }
    chromosome[i] = newColor;
    // refac vectorul de culori disponibile
    for (j = 0 to numberOfVerticies) {
      availableColors[j] = true;
    }
  }
}
```

Incrucisarea

Incrucisarea este clasica. Generez random un punct de taiere pana la lungimea cromozomului si interschimb incepand cu acel punct valorile cromozomilor.

Exemplu:

- Cromozom 1 : (1, 0, 1, **2, 1, 0**)
- Cromozom 2 : (0, 1, 2, **1, 0, 1**)
- Pozitie generata: 2
- Cromozom 1 rezultat: (1, 0, 1, **1, 0, 1**)
- Cromozom 2 rezultat: (0, 1, 2, **2, 1, 0**)

3 Influenta Parametrilor

Empiric se poate observa ca incrucisarea nu joaca un rol fundamental in rezultatul pe care il ofera algoritmul. In aceasta abordare mutatia impreuna cu numarul de generatii si cu dimensiunea populatiei creste sunt elementele cheie ale unui rezultat bun.

Exemplu pentru a ilustra faptul ca incrucisarea joaca un rol foarte mic in aflarea rezultatului - Restul parametrilor au fost aceeasi

Input	Rezultat GA	Rata Crossover
huck.col	11	0.01
huck.col	11	0.8
miles1000.col	43	0.01
miles1000.col	45	0.8
queen9-9.col	14	0.01
queen9-9.col	13	0.8

In aceste conditii empiric am observat ca mutatia trebuie sa fie in jur de $[0.001, 0.1]$ in functie de numarul de generatii care in general va fi in intervalul $[50, 250]$. Daca avem o mutatie prea mare(sau un numar prea mare de generatii), numarul de culori creste foarte mult (din pricina modului prin care fac mutatia), iar numarul de generatii il putem creste in momentul in care dorim o precizie mai mare, dar pierdem din timp.

Input	Pm	Pc	PopSize	NrGen	Rezultat
queen9_9.col	0.1	0.1	100	25	16
queen9_9.col	0.5	0.1	100	25	20
queen9_9.col	0.1	0.1	100	50	18
queen9_9.col	0.1	0.1	200	25	15

4 Rezultate

Algoritmul genetic l-am rulat de 15 ori cu diferiti parametrii.

Input	Varfuri	Muchii	Greedy	GA	Expected Coloring
miles1000.col	128	6432	44	45	42
homer.col	561	3258	15	20	13
fpsol2.i.1.col	496	11654	65	70	65
anna.col	138	986	12	13	11
myciel5.col	47	236	6	6	6
david.col	87	812	12	11	11
huck.col	74	604	11	11	11
queen9_9.col	81	2112	16	12	10

Se poate observa ca in general algoritmul greedy se comporta mai bine decat algoritmul genetic (si este mai rapid), insa acest lucru tine foarte mult de modul in care este aleasa, asa cum subliniam mai sus, ordinea nodurilor. In acest sens, algoritmul genetic desi este mult mai ineficient, ofera date mai sigure si mai apropiate de realitate fata de algoritmul de tip greedy.

5 Bibliografie

- [1] - https://en.wikipedia.org/wiki/Greedy-coloring
- http://profs.info.uaic.ro/ pmihaela/GA/laborator3.html
- http://ceur-ws.org/Vol-841/submission-10.pdf