Problema comis-voiajorului,
utilizând un Algoritm Genetic şi algoritmul lui Christofides.

Modoranu Cosmin, grupa A6, anul II
Dărăbăneanu Cosmin, grupa A6, anul II

Ianuarie, 2021

1 Abstract

Problema comisului-voiajor este o problemă NP-dificilă şi importantă în domeniul distribuţiei şi a logisticii. Acest document oferă informaţii legate de doi algoritmi utilizaţi în rezolvarea PCV-ului şi mai oferă rezultate care evidenţiază acurateţea mărita a algoritmului genetic faţă de cea a algoritmul lui Cristofides şi diferenţa de viteza de rulare a algoritmului lui Christofides care a rulat mult mai repede faţă de algoritmul genetic.

2 Introducere

Problema comisului-voiajor(PCV) este o problemă bine cunoscută şi importantă în optimizarea combinatorială.Scopul este de a găsi drumul cel mai scurt ce vizitează fiecare oraş dintr-o listă dată, cate o singură dată, drum care se întoarce la oraşul din care a plecat. Se evidenţiază diferenţele de performanţă utilizând algoritm genetic şi o euristică numită Algoritmul lui Christofides.

Rezolvarea PCV este o parte importantă fiind utilizată în multe aplicaţii precum rutarea vehiculelor, cablarea computerelor, în organizare, des întalnită în reţele de comunicaţii .

2.1 Descrierea problemei

Distanţa totală a unei soluţii poate fi exprimată în felul următor:

$$T_d = \sum_{i=1}^{n-1} d(V_i, V_{i+1}) + d(V_n, V_1),$$

unde $Path(\pi)$= $\{V_1, V_2, ..., V_n\}$ este o permutare a oraşelor $\{1, 2, ...n\}$ şi $d(V_i, V_{i+1})$ reprezintă distanţa de la oraşul V_i la oraşul V_j.

Se cere determinarea unui drum de lungime minimă în care fiecare oraş este vizitat o singură dată.

3 Metode

3.1 Algoritm genetic

Un Algoritm genetic : este o metodă ce are ca sursă de inspiraţie evoluţia biologică şi utilizează un vocabular împrumutat din genetică.
Sunt algoritmi ce îmbunătăţesc soluţia pas cu pas de-a lungul mai multor generaţii.
Pentru rezolvarea problemei vom defini următoarele elemente:
-o reprezentare a soluţiilor candidat;
-o procedură de iniţializare a populaţiei iniţiale de soluţii candidat;
-o funcţie de evaluare(funcţia fitness);
-o schemă de selecţie;
-crossover ;
-operatorii genetici ;
Astfel:
Reprezentarea soluţilor este facută printr-un vector de numere intregi ce reprezintă ordinea parcurgerii oraşelor. Este o permutare a elementelor între 0 si $n-1$ pentru n oraşe. . Se extrag coordonatele din unele instanţe din librăria TSPLIB [11] si se calculeaza matricea de cost adiacentă care e folosită de-al lungul algoritmului.
Funcţia fitness : Fitness-ul unei soluţii se calculează ca fiind

$$\mathbf{f(x) = distanta\_tur(x) + 0.0001}$$

Se urmăreste minimizarea acesteia pe parcursul algoritmului genetic.

TournamentSelection Alegem k soluții din populaţia actuală şi îl alegem ca rezultat cel mai bun dintre acestea.

RankSelection: Se sorteaza soluţiile după fitness şi se atribuie fiecăreia o probabilitate de a fi aleasă proporţională cu indecsul pe care se află.

RouletteWheelSelection: Calculăm valoarea fitness pentru toate soluţiile şi le atribuim fiecărei din ele o probabilitate care reflectă cât de bună este în comparaţie cu altele, pentru o soluție mai buă se va obţine o probabilitate mai mare ca aceasta să fie aleasă.

Crossover : Aplicând Crossover peste două soluţii din populaţie obţinem o nouă soluţie pe care o vom numi fiu. Se alege o tăietură aleatorie. Prima parte până la tăietură se menţine în fiu şi apoi se completează cu oraşele din a două soluţie în ordinea în care acestea apar.Fiul îşi completează lista oraşelor din a două soluţie doar dacă aceste soluţii nu se găsesc printre datele pe care fiul le are deja.

Mutatie: Se efectueaza prin **rotire** şi **interschimbare**.
Interschimbarea: Se aleg doua poziţii din soluţia pe care aplicăm operatorul şi se interschimbă valorile de la poziţiile respective.
Rotirea: Similar cu std::reverse. Alegem două poziţii în cadrul soluţie pe care aplicăm operatorul şi rotim elementele dintre cei doi indecşi adică primul element o să fie interschimbat cu ultimul, al doilea element o să fie interschimbat cu penultimul ş.a.m.d.

3.2 Algoritmul lui Christofides

Algoritmul lui Christofides: Algoritm care ne ajuta să găsim soluţii pentru PCV în instanţe care verifică inegalitatea triunghiulară. Este potrivit pentru că lucrăm cu coordonate 2D şi se respectă constrangerea aceasta. Este o schemă de aproximare în timp polinomial care garantează ca soluţiile sunt măcar sub $\frac{3}{2}$ din optim. Pentru a găsi o soluţie pentru PCV şi prin acest algoritm se parcurg urmatorii paşi:
1. Se gaseşte un arbore parţial de cost minim **T**.
2. Se formează mulţimea **O** ce conţine nodurile impare din **T**.
3. Se găseşte un cuplaj perfect **M** în subgraful indus de nodurile din **O**
4. Se combină muchiile lui **M** si **T** pentru a forma un multigraf **H** unde fiecare nod are grad par.
5. Se formează un circuit Eulerian din **H**.
6. Se transformă în circuit Hamiltonian prin reducerea nodurilor care se repetă (se iau scurtături).

4 Descriere experimente

Se rulează Algoritmul Genetic şi Algoritmul lui Christofides pe 6 instanţe cunoscute ale problemei. Algoritmul genetic a fost rulat de $n = 30$ ori şi se înregistrează date precum cea mai bună soluţie, media soluţilor şi cea mai slabă soluţie.
Parametrii AG-ului sunt următorii:
POP_SIZE = 1000 ; $P_{CX} = 0.2$; $P_M = 0.05$; MAX_GENERATIONS = 3000

5 Rezultate experimentale

Algoritm	Număr oraşe	Instanta	Worst result	Average result	Best result	Optim	Avg. time (s)
AGE	76	eil76	619.295	562.275	553.414	538	32s
AGE	198	d198	16821.7	16549.2	16463	15780	90s
AGE	318	lin318	53192.3	49233.7	48213.2	42029	211s
AGE	70	st70	695.4	687.53	686.2	675	24s
AGE	225	tsp225	4453.18	4376.65	4291.3	3919	149s
AGE	280	at280	3021.3	2943.63	2873.2	2568.88	166s
AC	76	eil76	670.45	670.45	670.45	538	1s
AC	198	d198	18183.4	18183.4	18183.4	15780	2s
AC	318	lin318	57608	57608	57608	42029	2s
AC	70	st70	842.3	842.3	842.3	675	1s
AC	225	tsp225	5145.24	5145.24	5145.24	3919	2s
AC	280	at280	3405	3405	3405	2587.88	2s

Figure 1: Un set de date obţinute cu ajutorul unui Algoritm Genetic si a algoritmului lui Christofides.

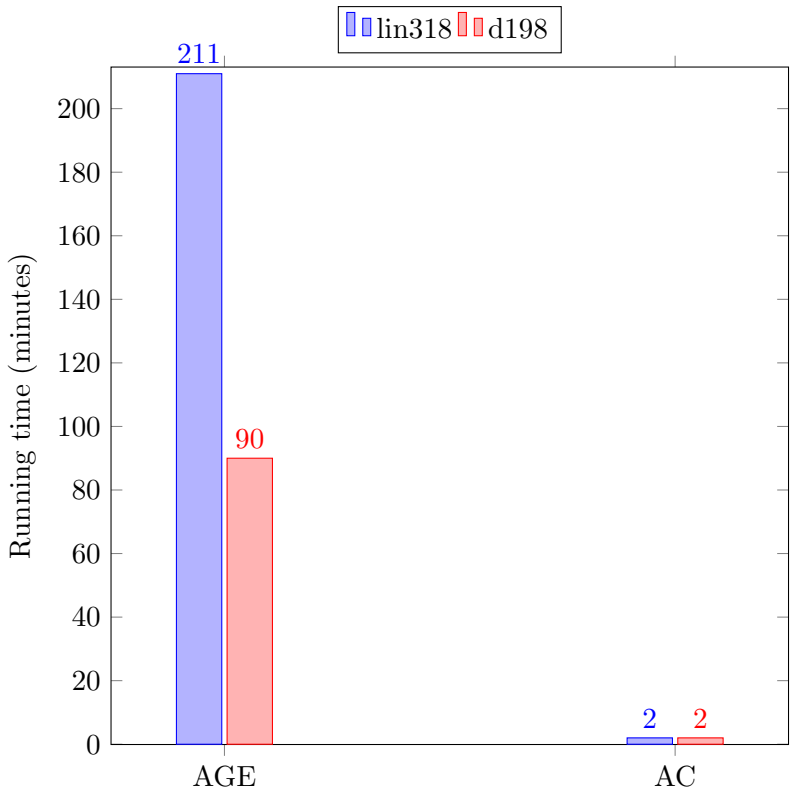

Figure 2: Grafic ce reprezintă timpul mediu al instanţelor d198 şi lin318 utilizând AGE şi AC.

6 Interpretarea rezultatelor

Din punct de vedere al timpului de execuţie algoritmul genetic oferă rezultat mult mai lent faţă de algoritmul lui Christofides. Rezultatele obţinute de algoritmul genetic sunt mult mai precise faţă de rezultatul obţinut de algoritmul lui Christofides dar în schimb AC a rulat foarte repede(1-3s). De exemplu pentru instanţa lin318 pentru AG rezultatul a fost 49233 vs 57600 pentru AC în condiţiile în care algoritmul începe cu prima generaţie la valoarea 400000 şi optimizează pe parcursul rulării un rezultat cu 345000 mai mic.

7 Concluzii

Algoritmul genetic a fost mai bun pentru calitatea raspunsurilor oferite şi este mai potrivit când se cere optimizarea soluţiei cât mai mult, cât mai aproape de optim. Algoritmul lui Christofides funcţionează mai bine pe număr mare de orase din cauza rapidităţii acestuia. Pentru număr mai mic de oraşe ($\leq$ 300) este mai potrivit algoritmul genetic precum diferenţa de timp(in medie 200 secunde) nu este aşa mare. Algoritmul lui Christofides foloseşte mai multă memorie pentru stocarea MST-ului şi a grafurilor obţinute la fiecare pas dar din cauza timpului mic de rulare remarcăm că nu sunt ţinute în memorie prea mult.

Proiectul se afla la : `https://github.com/KenKQDEs/Genetic-Algorithm-TSP/commits/develop?fbclid=IwAR2fnuXLVNRkuoBwt9fd2ScdxSaQp-8zkuPhJ_2_ahX5WF2rH7AnzBNfpEs`

8 Bibliografie

References

[1] Pagina laboratorului: `https://profs.info.uaic.ro/~eugennc/teaching/ga/`

[2] Documentație.Algoritmi Genetici: `http://students.info.uaic.ro/~vladut.ungureanu/Algoritmi-genetici-ID.pdf`

[3] Raport stintific.Autor: Chih-Cheng Hung
Informații utile despre algoritmul genetic utilizat pentru TSP simetric: `https://www.hindawi.com/journals/mpe/2015/212794/?fbclid=IwAR2p3Mp8c2ClB562FWm2O4d5oieIcUGamZUPc_WjV3xte_Nd17Pb9NPSh0k`

[4] Raport stintific.Autor:Xiao Yan Yun
Raport stintific despre TSP. `https://www.scientific.net/AMR.694-697.2901`

[5] Informatii PCV: `https://ro.wikipedia.org/wiki/Problema_comis-voiajorului#Algoritmul_lui_Christofides_pentru_PCV`

[6] Documentatie.Problema comis-voiajorului: `https://www.geeksforgeeks.org/travelling-salesman-problem-set-2-approximate-using-mst/`

[7] Documentatie.Algoritmul lui Christofides: https://youtu.be/M5UggIrAOME

[8] Documentatie.Algoritmul lui Christofides: https://en.wikipedia.org/wiki/Christofides_algorithm

[9] Documentație.Problema comis-voiajorului: https://www.youtube.com/channel/UCzvWh64GQm_OfkJXyYo7whQ

[10] Documentatie Latex: https://www.latex-project.org/help/documentation/

[11] TSPLIB: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/