Rezolvarea Problemei Comisului Voiajor cu ajutorul tehnicii Simulated Annealing si a unui Algoritm Genetic

Stratianu Bianca-Ionela

Ianuarie 2021

Abstract

Acest raport propune in analiza Problema Comisului Voiajor pe care am rezolvat-o cu ajutorul unui Algoritm Genetic si tehnicii Simulated Annealing. Am prezentat cei doi algoritmi, dar si modificarile aduse pentru a imbunatati rezultatele. S-a constatat faptul ca Simulated Annealing are rezultate mult mai bune decat Algoritmul Genetic.

1 Introducere

1.1 Introducere generala

Problema comis-voiajorului (TSP) pune urmatoarea intrebare: Data fiind o lista de orase si distantele intre fiecare doua orase, care este cel mai scurt traseu posibil care viziteaza fiecare oras o singura data si se intoarce la orasul de origine?" Ea este o problema NP-dificila in optimizarea combinatorie.

Problema a fost formulata pentru prima data în 1930 și este una dintre cele mai intens studiate probleme de optimizare. Ea este utilizata ca test pentru multe metode de optimizare. Chiar dacă problema este dificila computațional, se cunosc numeroși algoritmi euristici și exacti, astfel incat unele cazuri cu zeci de mii de orașe pot fi rezolvate complet cu o eroare foarte mica.

TSP este o incercare pentru multe euristici generale concepute pentru optimizare combinatorie, cum ar fi algoritmi genetici, simulated annealing, cautarea tabu, algoritmul optimizat al coloniei de furnici.

Astfel, pentru a gasi ceal mai scurt traseu intre doua oare voi folosi doi algortm: simulated annealing si algoritm genetic.

Acest raport isi propune studierea comportamentului celor doi algoritmi in determinarea minimelor unui numar de 10 trasee, avand un numar diferit de orase.

Vom cauta ce modificari am putea aduce pentru a obtine rezultate mai bune celor doi algoritmi pentru această problema.

1.2 Motivatie

TSP are mai multe aplicatii cum ar fi în planificare, logistica, si la fabricarea de microcipuri. Usor modificata, ea apare ca o subproblema in multe domenii, cum ar fi secventierea ADN-ului. În aceste aplicatii, conceptul de oras reprezinta, de exemplu, clienti, puncte de sudura, sau fragmente de ADN, si conceptul de distantă reprezinta durata de deplasare, costul de realizare, sau o masura a similaritatii intre fragmente de ADN.

Astfel am prezentat modificarile aduse celor doi algoritmi, rezultate obtinute, dar si timpul, pentru a vedea care dintre cei doi poate fi folosit in practica, care aduce un timp cat mai convenabil si o solutie cat mai apropiata de minim.

1.3 Descrierea problemei

Un comis-voiajor pleaca dintr-un oras, trebuie sa viziteze un numar de orase si sa se intoarca in orasul de unde a plecat cu efort minim. Se cere traseul pe care trebuie sa-l urmeze comis-voiajorul, astfel incat sa parcurga un numar minim de kilometri .

TSP poate fi modelata ca graf neorientat ponderat, astfel incat orasele sa fie noduri ale grafului, drumurile sa fie muchii, iar distanțta unei cai sa fie ponderea fiecarei muchii. Este o problema de minimizare cu incepere si terminare într-un anumit nod, dupa ce se viziteaza fiecare nod o singura data.

2 Metode

TSP este o incercare pentru multe euristici generale concepute pentru optimizare combinatorie, cum ar fi algoritmi genetici, simulated annealing, cautarea tabu, algoritmul optimizat al coloniei de furnici. Pentru a rezolva Problema Comisului Voiajor am ales doi algoritmi foarte bine cunuscuti: unul care se bazeaza pe Algoritmi Genetici si unul pe metoda Simulated Annealing.

2.1 Simulated Annealing

Simulated Annealing (SA) este o forma eficienta si generala de optimizare. Este util in gasirea optimelor globale în prezenta unui numar mare de optime locale. Annealing" se refera la o analogie cu termodinamica, în special cu modul in care metalele se racesc si se recocesc, deoarece in acest algoritm foarte imortanta este temperatura de start, dar si modul in care aceasta se modifica, ideal fiind atunci cand se insista mai mult pe valorile mai mici, pentru a ne putea apropia cat mai mult de rezultatul problemei.

Algorimul alege o solutie aleatorie, si un vecin al solutiei alese, tot aleatoriu fiind ales, prin schimbarea valorilor. Algoritmul se opreste la indeplinirea unei conditii, si se termina cand temperatura a scazut suficient.

Acesta este o imbunatatire adusa metodei Hill Climbing, diferenta fiind ca Simulated Annealing permite evadarea din punctele de minim local.

2.2 Algoritm Genetic

Algoritmii genetici iau ca model procesele naturale precum ar fi selectia, incrucisarea, mutatia, migratia, influenta mediului local si a vecinatatilor. Este de remarcat aici ca algoritmii evolutivi lucreaza asupra unei intregi populatii de indivizi (solutii) si nu asupra unei singure solutii. În acest mod este clar ca procesul de cautare are loc in maniera paralela.

Algoritmul genetic propune folosirea urmatoarelor proceduri:

1.Populatia initiala: este o multime de cromozomi care trebuie sa fie suficient de mare pentru ase putea realiza un numar multumitor de combinatii intre cromozomii componenti, dar nu prea mare, deoarece acest lucru va incetini algoritmul

2.Selectia este procedura prin care sunt alesi cromozomii ce vor supravietui in generatia urmatoare, pentru o evolutie cat mai buna, indivizilor mai bine adaptati li se vor da sanse mai mari;

3.Mutatia: se trece prin fiecare gene a cromozomului si se modifica.Apare in principal pentru a evita caderea solutiilor intr-un optim local.

4.Incrucisarea: se alege un punct de taiere aleatoriu si se selecteaza 2 cromozomi.

5.Functia fitnnes: gradul de adaptare la mediu.In cazul problemei comisvoiajorului, fitness-ul este costul drumului codificat derespectivul cromozom. Aici ne intereseaza sa obtinem un cromozom cu un fitness cat maimic.

Solutia returnata de un algoritm genetic este cel mai bun individ din ultima generatie.

3 Experimente

3.1 Simulated Annealing

Semnificative pentru algoritm sunt; temperatura initiala(T=1000), temperatura finala(10^{-8}). Numarul de iteratii pentru inputuri cu dimensiuni<120 este de 300, iar temperatura este inmultita cu 0,99 , iar pentru inputuri cu dimensiuni >=120, numarul de iteratii este de 900, iar temperatura este inmultita cu 0,9999

De asemena, algoritmul a fost rulat de 30 de ori pentru fiecare instanta, pentru a obtine rezultate relevante din punct de vedere statistic.

3.2 Algoritm Genetic

Dimensiunea populatiei este de 500, iar numarul de iteratii de 9000. Astfel, pentru inputuri cu dimensiuni<120 probabilitatea de mutatie este 0,1, iar probabilitatea de crossover este 0.3, iar pentru inputuri cu dimensiuni>=120, probabilitatea de mutatie este 0,005, iar probabilitatea de crossover este 0.2.

Selectia aleasa este roata norocului: parintii sunt selectati in conformitate cu fitness-ul lor. Cu cat sunt mai buni, cu atat sansa lor de a fi alesi este mai mare. Ca o ruleta in care sunt dispusi toti cromozomii din populatie. Fiecarui

cromozom ii corespunde un sector al ruletei, direct proportional, ca marime, cu fitness-ul cromozomului respectiv. In acest fel cromozomiicu fitness mai mare au atasate sectoare mai mari iar cei cu fitness mic au atasate sectoaremai mici. La aruncarea bilei pe ruleta exista mai multe sanse de alegere pentrucromozomii cu fitness mare.

Iar pentru rezultate cat mai aproape de minim, am folosit si elitism(copierea celor mai buni cromozomi in noua populatie, fara a-i schimba) pentru un numar de 3 cromozomi.

De asemena, algoritmul a fost rulat de 30 de ori pentru fiecare instanta, pentru a obtine rezultate relevante din punct de vedere statistic.

4 Rezultate

4.1 Simulated Annealing

Instanta	Dimensiunea	Solutia optima	Minimul	Maximul	Media	Dev. stan.	Timpul
br17	17	39	39	39	39	0	100
ftv33	34	1286	1230	1450	1331	23.01	520
p43	43	5620	5622	5635	5630	2.56	896
ry48p	48	14422	14770	15191	15021	91.87	1032
ftv70	71	1950	2128	2302	2240	43.1	1549
kro124	100	36230	38302	39959	39903	402.32	1743
rbg323	323	1326	1385	1422	1402	8.21	4706
rbg358	358	1163	1193	1263	1240	7.02	5460
rbg403	403	2465	2478	2501	2489	6.76	5808
rbg443	443	2720	2745	2780	2760	8.84	6548

4.2 Algoritm Genetic

Instanta	Dimensiunea	Solutia optima	Minimul	Maximul	Media	Dev. stan.	Timpul
br17	17	39	39	40	39.5	0.30	326
ftv33	34	1286	1370	1450	1393	31.08	599
p43	43	5620	5682	5935	5730	34.96	956
ry48p	48	14422	22070	26031	23409	1645.27	1134
ftv70	71	1950	3721	4292	3995	228.34	1647
kro124	100	36230	74319	87959	80431	6245.93	2343
rbg323	323	1326	3902	4167	4075	97.98	19706
rbg358	358	1163	4136	4690	4376	182.52	25960
rbg403	403	2465	5287	5489	5354	102.34	31108
rbg443	443	2720	5963	6521	6302	190.17	37848

5 Comparatii

Putem observa din tabelele de mai sus ca rezultate mai apropiate de minim sunt obtinute cu algoritmul Simulated Annealing, avand si timpul mai putin

decat in cazul unui algoritm genetic.

Se observa ca algoritmul genetic nu are rezultate foarte bune, avand o rata de esec mai mare, dar si timp de executie foarte mare mai ales pentru instante mai mari.

Se observa ca modificarile aduse in functie de dimensiunea instantei au ajutat la crearea unei solutii mai apropiate de minimul real.

Vedem ca pentru instante de dimensiuni mici, dar si mari, Simulated Annealing da rezultate exacte sau aproape exacte, intr-un timp rezonabil.

6 Concluzii

Din toate punctele de vedere se poate observa ca Simulated Annealing are rezultate mult mai bune decat algoritmul genetic, atat pentru dimensiuni mici, cat si pentru dimensiuni mari. Imbunatatirile aduse celor doi algoritmi au adus rezultate mai bune, fiind observabile si din tabel.

Astfel, Simulated Annealing este mai recomandat de pus in practica decat un Algoritm Genetic.

7 Bibliografie

References

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

[2] https://gitlab.com/eugennc/teaching/-/blob/master/GA/texample.pdf

[3] https://gitlab.com/eugennc/teaching/-/blob/master/GA/texample.tex

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

[5] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/atsp/

[6] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ATSP.html

[7] https://www.academia.edu/6237245/algoritmi_genetici

[8] https://www.yumpu.com/ro/document/read/25070341/algoritmi-genetici-pentru-rezolvarea-problemelor-prin-sorin-

[9] https://profs.info.uaic.ro/~eugennc/teaching/ga/#Notions04

[10] https://www.victorneagoe.com/university/prai/lab5a.pdf

[11] https://staff.fmi.uvt.ro/~daniela.zaharie/cne2014/curs/cne2014_slides6.pdf

[12] https://core.ac.uk/download/pdf/208230471.pdf

[13] https://www.ijert.org/a-comparative-study-of-ga-and-sa-for-solving-travelling-salesman

[14] https://www.timvoelcker.de/genetic_algorithm.html

[15] https://www.geeksforgeeks.org/traveling-salesman-problem-using-genetic-algorithm/

[16] https://www.hindawi.com/journals/cin/2017/7430125/