Problema comis-voiajorului rezolvată de metode euristice

Lipșa Ondina-Patricia
grupa B1

Stanciu Victor-Nicolae
grupa B1

7 ianuarie 2021

Abstract. Acest raport surprinde modul de funcționare a metodei Simulated Annealing și a unui algoritm genetic pentru problema comis-voiajorului, precum și o analiză comparativă dintre cei 2 algoritmi, prin raportarea la rezultatele experimentale obținute. Experimentul a fost rulat de 30 de ori independent, pentru aceleași instanțe ale problemei și s-a putut observa că Simulated Annealing și algoritmul genetic returnează rezultate similare în cazul instanțelor cu 17, 36 și 124 de orașe, iar pentru testele cu peste 300 de orașe, rezultatele obținute prin metoda SA sunt mult mai apropiate de minimul cunoscut.

1 Introducere

Problema comis-voiajorului(PCV) este una dintre cele mai studiate probleme de optimizare. Ea presupune găsirea unui ciclu de cost minim într-un graf cu n orașe care trece exact o singură dată prin toate nodurile, cunoscând distanța dintre oricare 2 orașe. Varianta asimetrică, folosită în cadrul acestui studiu, se bazează pe aceeași idee, cu precizia că între 2 orașe A și B distanța de la nodul A la nodul B poate fi diferită de distanța de la nodul B la nodul A.

Aceasta este o problemă NP-dificilă, pentru care nu se cunoaște o rezolvare eficientă folosind algoritmi determiniști, din motive intuitive: având n orașe, există n! posibilități de construire a circuitelor care să viziteze toate orașele, deci pentru doar 10 orașe există peste 3,6 milioane de circuite diferite. Deoarece o căutare printre toate soluțiile posibile devine mult prea costisitoare pe măsură ce numărul de orașe crește, este preferabilă utilizarea algoritmilor euristici, ce aproximează soluția optimă și limitează spațiul de căutare.

2 Metode

2.1 Algoritmi genetici

Un algoritm genetic este o metodă meta-euristică inspirată din procesul natural de selecție al algoritmilor evolutivi, care menține o populație de indivizi(genetați

inițial aleator). Acești indivizi reprezintă soluțiile candidat ale problemei, iar scopul algoritmului este de avea o populație de indivizi din ce în ce mai adaptați pe măsură ce este iterat algoritmul (fiecare iterație se numește *generație*).

2.1.1 Reprezentarea

Indivizii sunt reprezentați printr-un vector ce conține toate cele n orașe, denumit *cromozom*, iar acesta conține mai multe *gene*, fiecare genă fiind un anumit oraș vizitat pe o anumită poziție din ordinea orașelor. Această poziție a vizitării unui oraș se numește *locus*. Ordinea orașelor în cromozom este chiar reprezentarea drumului, cu precizia că ultimul oraș este și primul, pentru a avea un circuit.

2.1.2 Pseudocod

```
begin
        generate_population();
        evaluate_population();
        while(nrGenerations<10000) do
                rouleteWheel();
                mutation();
                crossover();
                evaluate_population();
                nrGenerations=nrGenerations+1;
end
```

Algoritmul generează inițial o populație de 100 de indivizi, care sunt evaluați și ulterior selectați în funcție de calitatea lor, iar pe baza reprezentărilor existente se formează soluții noi prin aplicarea operatorilor genetici de mutație și crossover.

Procesul de evaluare, selecție și modificare este repetat (de 10000 de ori) pentru a obține mai multe generații de indivizi.

2.1.3 Evaluarea

Pentru fiecare generație, calitatea indivizilor este măsurată pe baza unei funcții *fitness*, pentru a-i putea selecta pe cei mai bine adaptați pentru supraviețuire. Funcția fitness (f) trebuie aleasă invers proporțional cu rezultatul fiecărui individ din cadrul populației, deci un cromozom care are o lungime a circuitului mai mică va avea o valoare mai mare în cadrul funcției fitness.

Am ales funcția f(x)= 1/pathLen, unde pathLen este lungimea circuitului în cadrul cromozomului x.

2.1.4 Selecția

Am folosit selecția bazată pe Roata norocului, unde probabilitatea unui individ de a fi selectat este proporțională cu fitness-ul acestuia. Pentru a implementa această metodă de selecție, sunt calculate probabilitățile indivizilor de

a fi aleși dupa formula p(i)=f(i)/T, unde T este suma totala a fitness-urilor, și prin adunarea probabilităților q(i+1)=q(i)+p(i) se observă o șansă mai mare ca un număr generat aleator să fie în intervalul (q(i),q(i+1]), pentru un cromozom cu fitness-ul mai mare, deci o șansă mai mare de selecție a acelui individ pentru supraviețuire.

2.1.5 Elitismul

Roata norocului are totuși o șansă mică să nu aleagă indivizii mai adaptați, de aceea utilizarea elitismului este foarte utilă. Acesta asigură păstrarea cromozomilor cu cele mai mari valori ale fitness-ului în noua generație și previne aplicarea operatorilor genetici asupra lor (am ales păstrarea primilor 2 cei mai adaptați cromozomi).

2.1.6 Mutația

Mutația modifică o genă(sau mai multe) aleasă aleator a unui cromozom dintre cei selectați(cu o probabilitare de 0.7%). Această probabilitate trebuie să fie scăzută, altfel algoritmul ar deveni un simplu algoritm de căutare aleatoare. Scopul mutației este de a introduce diversitate în populația curentă, pentru a evita oprirea într-un punct de minim local. Am folosit 2 tipuri de mutație: mutația TWORS, care inversează 2 gene alese pe poziții aleatoare și mutația RMS, care alege o secvență de gene S, delimitată de două poziții i și j alese aleator și inversează întreaga secvență.

2.1.7 Crossover

Este folosit pentru a combina informația genetică dintre 2 cromozomi(cu o probabilitate de 30%), iar rezultatul încrucișării este adăugat în populația din noua generație și astfel valoarea medie a fitness-ului crește (în general) deoarece doar cei mai adaptați indivizi sunt selectați pentru supraviețuire iar majoritatea celor care nu produc rezultate bune sunt eliminați (există o probabilitate mică de selecție a acestora, fiind utili pentru menținerea diversității).

3 Experiment

Experimentul constă în generarea unei populații inițiale de 100 de indivizi, din care sunt selectați indivizii cu cea mai bună rată de supraviețuire (prin utilizarea roții norocului), cărora li s-au aplicat operatorii de mutație (cu o probabilitate de 1%) și de încrucișare (probabilitate de 30%), coromozomii rezultați sunt adăugați în noua populație, iar procesul este repetat pentru 1000 de generații.

Cel mai adaptat cromozom al fiecărei generații este memorat, deoarece sunt șanse ca el să fie pierdut în urma procesului de selecție sau prin aplicarea operatorilor genetici.

Rezultatele experimentale au fost obținute după rularea algoritmului independent de 30 de ori.

4 Rezultate

5 dimensiuni

funcție	favg	fmin	fmax	minimul global	timp
De Jong	0.00044	0.00013	0.00582	0	9.24138 s
Schwefel	-2056.41715	-2093.16052	-1808.56421	-2094.9145	15.27951 s
Rastrigin	0.99527	0.02485	2.09382	0	9.48694 s
Michalewicz	-4.05729	-4.59322	-3.85206	-4.687	8.61081 s

10 dimensiuni

funcție	favg	fmin	fmax	minimul global	timp
De Jong	0.00137	0.00025	0.00449	0	18.48733 s
Schwefel	-4070.75375	-4123.15733	-3906.21062	-4189.829	29.18831 s
Rastrigin	5.32577	1.03988	11.20143	0	19.17876 s
Michalewicz	-9.19968	-9.60155	-7.26882	-9.66	17.85443 s

30 dimensiuni

funcție	favg	fmin	fmax	minimul global	timp
De Jong	0.01341	0.00376	0.02901	0	52.63290 s
Schwefel	-12170.82415	-12524.30573	-10378.77488	-12569.487	89.23895 s
Rastrigin	23.72290	16.80312	49.13765	0	55.21930 s
Michalewicz	-28.14372	-28.99019	-24.69863	?	84.62561 s

5 Comparații

În continuare vom analiza media rezultatelor obținute de cei 3 algoritmi pentru funcțiile pe 30 de dimensiuni:

funcție	FIHC	BIHC	SA	GA
De Jong	0.00075	0.00075	1.60874	0.01341
Schwefel	-9418.21075	-9865.00394	-12099.98071	-12170.82415
Rastrigin	73.81003	61.17492	31.27046	23.72290
Michalewicz	-25.70841	-26.57781	-27.08379	-28.14372

Se observă o îmbunătățire semnificativă a funcției De Jong în cazul utilizării unui algoritm genetic, față de rezultatul obținut cu Simulated Annealing, însă tot varianta Hill Climbing este mai potrivită pentru o astfel de funcție convexă și unimodală, deoarece algoritmul HC nu se poate bloca în minime locale (are un singur minim, global) și returnează de fiecare dată rezultatul exact.

În cazul funcției De Jong, algoritmul genetic, se apropie de minimul global, însă datorită măririi numărului de soluții candidat generate aleator și deoarece cromozomii sunt comparați doar cu alți cromozomi, și nu cu vecinii săi, nu poate parcurge un "traseu" în procesul de găsire a minimului similar cu Hill Climbing.

În schimb, pentru funcțiile multimodale, deoarece Hill Climbing se oprește atunci când nu mai sunt găsiți vecini mai buni decât soluția curentă și returnează doar minimele locale, este preferabilă folosirea unei metode care poate continua procesul de optimizare și atunci când se ajunge într-un minim local.

Simulated Annealing poate selecta și soluții mai slabe decât minimul curent, însă spațiul căutării nu este explorat la fel de mult precum în cazul algoritmilor genetici, pentru că sunt analizați doar vecinii unei soluții aleatoare, și nu o întreagă populație. Această diferență este foarte vizibilă în cazul rezultatelor pentru funcția Rastrigin, algoritmul genetic îmbunătățind semnificativ rezultatele obținute.

6 Concluzii

Pe baza rezultatelor experimentului s-a observat că algoritmii genetici produc rezultate mai bune decât metodele studiate anterior pentru funcții multimodale, însă pentru funcțiile care au un singur minim local este preferabilă folosirea unei variante de Hill Climbing, deoarece acestea returnează de fiecare dată rezultatul optim.

6.1 Cercetări viitoare

Algoritmul genetic pornește cu o populație aleatoare și, pe parcursul iterării, indivizii din populație se îmbunătățesc, însă spre final populația nu mai este la

fel de diversificată și șansa de găsire a unor soluții mai bune scade, deoarece sunt aplicați operatori genetici între cromozomi cu configurații asemăntoare. O idee de optimizare ar fi să eliminăm indivizii care au un anumit grad de similaritate între ei și să permitem adăugarea celor care au configurații diferite, sau chiar a unor noi indivizi generați aleator. Astfel se poate menține o diversitate mai mare a populației.

Bibliografie

[1] Pagina cursului:
https://profs.info.uaic.ro/~eugennc/teaching/ga/res/gaSlidesOld.pdf
https://profs.info.uaic.ro/~eugennc/teaching/ga/

[2] Algoritmi genetici
https://en.wikipedia.org/wiki/Genetic_algorithm
https://www.sciencedirect.com/topics/engineering/genetic-algorithm
https://en.wikipedia.org/wiki/Genetic_representation
https://en.wikipedia.org/wiki/Fitness_function
https://www.msi.umn.edu/sites/default/files/OptimizingWithGA.pdf

[3] Roulette Selection in Genetic Algorithms https://www.baeldung.com/cs/genetic-algorithms-roulette-selection

[4] Operatori genetici
https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)
https://www.sciencedirect.com/topics/engineering/crossover-probability
https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm)

[5] Genetic Algorithms vs. Simulated Annealing https://scholar.smu.edu/cgi/viewcontent.cgi?article=1000&context=engineering_compsci_research