Specializare: Calculatoare anul 3

Realizat de:

* Armand Kalisch
* Zaharie Dorin
* Dan Lorena
* Iacob Razvan

Problema Furnicii

O imagine care conține schiță, desen, desen animat, clipart

Descriere generată automatCuprins

Contents

[1.INTRODUCERE 2](#_Toc155978383)

[2.Descrierea Problemei 3](#_Toc155978384)

[3.Modelare Matematica 4](#_Toc155978385)

[4.Starea curentă a domeniului 5](#_Toc155978386)

[5.Modelarea teoretica 5](#_Toc155978387)

[6.Implementare 6](#_Toc155978388)

[7.Rezultate 7](#_Toc155978389)

[8.Concluzii 12](#_Toc155978390)

# 1.INTRODUCERE

Alegerea de a rezolva problema furnicii în cadrul acestui proiect a fost determinată de dorința de a găsi soluții pentru optimizarea traseelor în contexte complexe. Furnicile reale demonstrează un comportament colectiv remarcabil în găsirea celor mai scurte căi între mușuroaiele lor și sursele de hrană. Am ales să explorăm această strategie naturală într-un mediu de programare, sperând să descoperim modalități eficiente de optimizare a traseelor în diverse situații practice.

Furnicile din natură pot găsi drumul cel mai scurt către sursa de hrană fără a vedea în detaliu mediul din jur și fără să comunice direct între ele.

Ele folosesc feromoni pentru a marca traseele și pentru a ghida furnicile următoare către sursa de hrană.

În loc de feromoni reali, furnicile din cod ajustează traseele lor și calculează un fitness în funcție de cât de bine se descurcă pe traseu.

Furnicile "lasă urme virtuale" prin ajustarea traseului, iar cele mai bune trasee sunt menținute în timp, similar cu cum feromonii persistă în mediul natural.

Furnicile din cod se adaptează la mediu fără a avea o imagine directă a matricei sau a altor detalii vizuale.

Ele iau decizii pe baza informațiilor indirecte, cum ar fi feedback-ul dat de fitness și ajustările aleatoare ale traseelor.

# 2.Descrierea Problemei

Configurația Inițială:

Furnicile încep de la un punct de start într-o matrice care reprezintă un patratel.

În patratel există anumite celule care pot conține hrană (reprezentată de 1) sau pot fi goale (reprezentate de 0).

Scopul:

Scopul furnicilor este să parcurgă matricea iar apoi să găsească cel mai optim traseu de a culege toata mancarea.

Reprezentarea Traseului:

Traseul furnicilor este reprezentat printr-un șir de direcții, unde fiecare direcție indică următoarea mișcare a furnicii (Nord, Sud, Est sau Vest).La noi fiind cu insemnatatea:

0: Nord - Furnica se deplasează în sus, deci coordonata y scade cu 1.

1: Sud - Furnica se deplasează în jos, deci coordonata y crește cu 1.

2: Est - Furnica se deplasează spre dreapta, deci coordonata x crește cu 1.

3: Vest - Furnica se deplasează spre stânga, deci coordonata x scade cu 1.

Fitness-ul Furnicilor:

Fitness-ul furnicilor este calculat pe baza distanței parcurse și a cantității de hrană colectată.

Scopul este să se găsească furnici care să aibă un traseu eficient în a parcurge întregul teritoriu și să colecteze cât mai multă hrană posibil.(scopul e sa ajungem la colectarea in totalitate a hranei)

Algoritmul Evolutiv:

Furnicile trec printr-un proces evolutiv în care se creează noi generații pe baza furnicilor cu cel mai bun fitness.

Acest proces implică crossover :

Se alege aleator un punct de crossover pe traseul furnicilor părinți. Acest punct reprezintă poziția de pe traseu de la care se vor combina traseele părinților.

Copierea părții de traseu a părintelui 1:

Se copiază începutul traseului furnicii părinte 1 până la punctul de crossover în traseul copilului.

Concatenarea restului traseului de la părintele 2:

Se obține porțiunea rămasă a traseului de la părintele 2, începând de la punctul de crossover până la sfârșitul traseului său.

Crearea traseului copilului:

Traseul copilului este creat prin concatenarea părții copiate de la părintele 1 cu porțiunea rămasă de la părintele 2.

# 3.Modelare Matematica

1.Functia de fitness:

Distanța Totală: Fitness-ul include o componentă negativă legată de distanța totală parcursă. Scopul este să se favorizeze furnicile care parcurg trasee mai scurte.

Numărul de Hrane Colectate: Se adaugă un bonus pentru fiecare unitate de hrană colectată, încurajând furnicile să colecteze cât mai multă hrană.

Fitness=Distanța Totală−Penalizare+Bonus pentru Hrană Colectată

Generarea Traseului Initial:

Traseul furnicilor este generat aleatoriu, iar lungimea traseului depinde de dimensiunea matricei și a generației curente.

Crossover:

Crossover-ul combină traseele părinților la un punct ales aleator.

Mutație:

Mutația adaugă mici modificări ale traseului, selectând aleator noi direcții pentru anumite secțiuni ale traseului.

Local Search:

Pentru fiecare direcție posibilă, se verifică dacă schimbarea direcției îmbunătățește fitness-ul furnicii.

Selecția Părinților:

Părinții pentru crossover sunt selectați cu o probabilitate mai mare pentru furnicile cu fitness mai mare.

Matricea de Hrană și Matricea de Distanțe:

Aceste matrice sunt utilizate pentru a determina disponibilitatea hranei și distanțele dintre poziții în calculul fitness-ului furnicilor.

Penalizare și Bonus:

Penalizări pentru lungimea traseului furnicilor și bonusuri pentru hrana colectată ajustează fitness-ul în funcție de obiectivele problemei.

# 4.Starea curentă a domeniului

Problema furnicilor, cunoscută și sub denumirea de "Algoritmul furnicilor" sau "Ant Colony Optimization" (ACO), a fost propusă și dezvoltată în principal de Marco Dorigo, un cercetător în domeniul informaticii și inteligenței artificiale.

Algoritmul furnicilor a fost introdus în lucrarea sa de doctorat intitulată "Optimization, Learning and Natural Algorithms" (Optimizare, Învățare și Algoritmi Naturali), susținută în 1992. În această lucrare, Marco Dorigo prezintă ideea de a utiliza comportamentul furnicilor reale în găsirea de căi optime în rezolvarea problemelor de optimizare combinatorială. (Dorigo, 2006)

În contextul actual, optimizarea traseelor nu este doar o problemă teoretică, ci și un aspect esențial pentru eficiența operațională în diverse industrii, inclusiv transport, logistică, planificare urbană și tehnologii inteligente. Companiile și organizațiile sunt tot mai interesate să implementeze soluții inovatoare care să le permită să își optimizeze resursele și să reducă costurile.

S-au dezvoltat numeroase abordări, inclusiv algoritmi bazati pe inteligenta artificiala, precum rețele neuronale și algoritmi genetici, care au demonstrat eficacitate în rezolvarea problemelor complexe de rutare. În plus, simulările pe scară largă și tehnologiile de realitate virtuală au facilitat testarea și evaluarea performanței acestor algoritmi în medii diverse și realiste.

De asemenea, standardizarea schimbului de date și interoperabilitatea între diferite sisteme au devenit priorități importante în domeniu, asigurând integrarea optimă a acestor tehnologii în contextul aplicațiilor practice.

În cadrul acestei evoluții, explorarea comportamentului furnicilor în găsirea căilor optime a devenit un subiect de cercetare din ce în ce mai captivant. Abordarea biologică aduce nu doar o sursă de inspirație, ci și un cadru conceptual bogat pentru dezvoltarea de algoritmi și modele eficiente.

# 5.Modelarea teoretica

Avem o matrice bidimensională reprezentând un mediu în care furnicile trebuie să găsească hrană.

Fiecare element din matrice poate conține hrană (1) sau să fie gol (0).

Reprezentarea Cromozomială:

Fiecare furnică este reprezentată printr-un cromozom, care este o secvență de direcții (0 - Nord, 1 - Sud, 2 - Est, 3 - Vest).

Lungimea cromozomului este determinată de dimensiunea matricei și generația curentă.

Funcția de Fitness:

Fitness-ul furnicilor este calculat în funcție de eficiența traseului lor.

Distanța totală parcursă este luată în considerare, iar furnicile sunt penalizate pentru trasee mai lungi.

Se acordă bonusuri pentru hrana colectată.

Inițializarea Populației:

La început, o populație de furnici este creată aleatoriu, fiecare furnică având un traseu aleator.

Operatori Genetici:

Crossover (Recombinare): Furnicile selectate pentru reproducere își combină traseele la un punct aleatoriu, generând astfel furnici descendente.

Mutație: Se aplică mici modificări aleatoare asupra traseelor furnicilor pentru a explora noi soluții.

Evaluarea și Selecția:

Furnicile sunt evaluate în funcție de fitness-ul lor.

Furnicile cu un fitness mai mare au o probabilitate mai mare de a fi selectate pentru reproducere.

Evoluție:

Ciclul de reproducere, evaluare și selecție este repetat de mai multe ori (generații) pentru a permite populației să evolueze.

Convergență:

Algoritmul evolutiv are ca obiectiv găsirea unui traseu optim care maximizează fitness-ul, reprezentând furnicile care colectează hrana într-un mod eficient.

# 6.Implementare

In primul rand, am ales sa facem codul în Java, limbaj de programare orientat pe obiecte.

Am creat 2 clase principale: Ant si AntColony.

Clasa Ant:

Reprezintă furnica și include atribute precum path, fitness, și foodCollected.

Are metode pentru a genera un traseu aleator, a efectua căutarea locală, și a calcula fitness-ul furnicii.

Clasa AntColony:

Reprezintă colonia de furnici și include atribute precum ants, foodMatrix, distanceMatrix, și generation.

Are metode pentru a inițializa furnicile, a efectua operații de crossover și mutație, și a evolua colonia de furnici prin mai multe generații.

Algoritmul Evolutiv:

Generarea furnicilor inițiale se face cu trasee aleatoare și calcularea fitness-ului lor în funcție de matricea de hrană și matricea de distanțe.

Se aplică operații de crossover și mutație pentru a genera noi furnici și a explora spațiul soluțiilor.

La fiecare generație, furnicile sunt evaluate, iar cele mai bune sunt selectate pentru reproducere.

Furnicile rezultate sunt adăugate în noua generație.

Matricea de Hrană și Matricea de Distanțe:

foodMatrix este o matrice bidimensională reprezentând locurile unde există hrană (1) și unde nu există (0).

distanceMatrix este o matrice bidimensională inițializată cu valori constante (în exemplu, toate elementele sunt setate la 1).

Generarea Inițială a Furnicilor și Evoluția:

În fiecare generație, furnicile sunt mutate și încrucișate pentru a produce o nouă generație.

Se repetă aceste operații pentru un număr specific de generații.(Pe care il putem modifica oricand)

# 7.Rezultate

Pentru n=5, dimensiunea populației 100 și numarul de generații 100 se afisează:

A screenshot of a computer

Description automatically generated

A black background with many small numbers

Description automatically generated with medium confidence

Pentru n=5, dimensiunea populației 1000 și numarul de generații 200 se afisează:

A screenshot of a computer

Description automatically generated

A number on a black background

Description automatically generated

Pentru n=5, dimensiunea populației 1000 și numarul de generații 2000 se afisează:

A screenshot of a computer program

Description automatically generated

A black screen with many small numbers

Description automatically generated with medium confidence

Pentru n=6, dimensiunea populației 100 și numarul de generații 100 se afisează:

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

Pentru n=6, dimensiunea populației 500 și numarul de generații 1000 se afisează:

A screenshot of a computer program

Description automatically generated

Blur a blurry image of a computer screen

Description automatically generated

# 8.Concluzii

În această implementare a algoritmului genetic pentru rezolvarea problemei furnicilor, am conceput o evoluție pentru găsirea unui traseu optim într-o matrice de hrană. Prin generarea și evoluția unei colonii de furnici, am căutat să găsim trasee eficiente care să maximizeze fitness-ul, luând în considerare atât distanța parcursă, cât și hrana colectată de către furnici.

Contributia la proiect:

Armand Kalisch

25% Proiect + 25% Documentatie

Dorin Zaharie

25% Proiect + 25% Documentatie

Iacob Razvan

25% Proiect + 25% Documentatie

Dan Lorena

25% Proiect + 25% Documentatie