UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DOMENIUL: Calculatoare și tehnologia informației

SPECIALIZAREA: Sisteme distribuite și tehnologii web

**Algoritmul de optimizare a coloniilor de furnici**

Proiect la disciplina Sisteme Multi-agent

Coordonator științific

Ș.l.dr.ing. Marius Gavrilescu

Studenti,

Aeloaiei Vlad

Dumitriu Mariana

Surdu Cristina

Grupa SDTW II 1A

1. **Notiuni teoretice**

În informatică și cercetarea operațională, algoritmul de optimizare a coloniilor de furnici este o tehnică probabilistică pentru rezolvarea problemelor de calcul care pot fi reduse la găsirea unor căi bune prin grafice. Furnicile artificiale reprezintă metode multi-agent inspirate de comportamentul furnicilor reale.

1. **Descrierea problemei studiate**

Aplicația își propune să implementeze algoritmul de optimizare a coloniilor de furnici utilizând un sistem multi-agent. Lumea descrisă de agenți este un graf conectat. Atât caracteristicile grafului cât și pașii algoritmului sunt descriși mai jos:

* Un nod este **baza de origine**;
* **Celelalte** **noduri** pot avea **„**mâncare**”**. Cantitatea de mâncare este reprezentată de o valoare întreagă: nodurile cu valoarea 0 nu au hrană, în timp ce celelate vor avea unități de mâncare în diferite cantități, reprezentate de valori diferite de 0;
* **Muchiile** grafului au câte o valoare asignată care inițial va fi 0;
* Mai multe furnici agent încep de la baza de origine și vor începe să caute mâncare la întâmplare, se vor deplasa aleatoriu de la un nod la celălalt de-a lungul muchiilor grafului;
* Odată ce o furnică a găsit mâncare(a ajuns pe un nod care este diferit de 0), va lua o unitate de mâncare(valoarea nodului scade cu 1) și pot fi două versiuni care pot fi urmate:

1. Furnica va returna unitatea de mâncare la baza de origine;
2. Furnica va depune unitatea de mâncare pe un nod la jumătatea distante dintre nodul current si baza de origine.

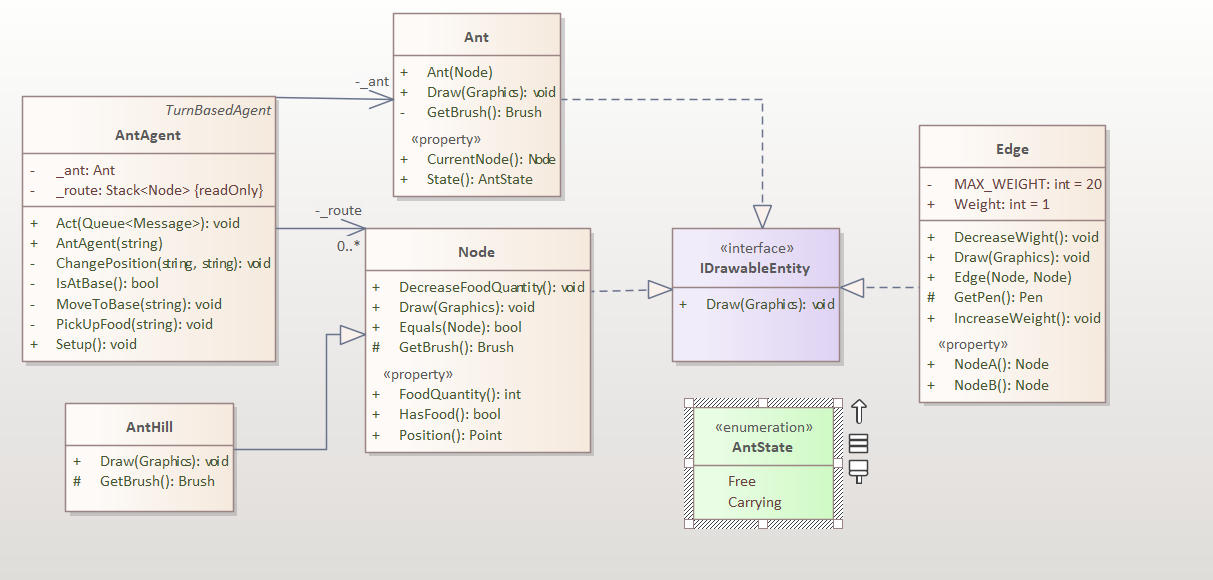
În ambele versiuni, furnica agent care transportă înapoi o unitate de mâncare crește greutatea fiecărei muchii pe care o parcurge cu 2. După ce depune unitatea de mâncare, va incepe căutarea din nou.

* Dacă o furnică care caută mâncare ajunge pe un nod cu muchie adiacentă diferită de 0, ea urmează această muchie și celelalte care sunt conectate și au valori diferite de 0, preferând întotdeauna muchiile cu cea mai mare valoare. Dacă ajunge pe un nod care are muchii cu valoarea 0(alta decât cea care a folosită pentru a ajunge pe nod), începe din nou căutarea aleatorie;
* Valorile nodurilor de descompun la un anumit interval de timp( de exemplu cu -0.1 la fiecare secundă).

1. **Implementarea algoritmului**

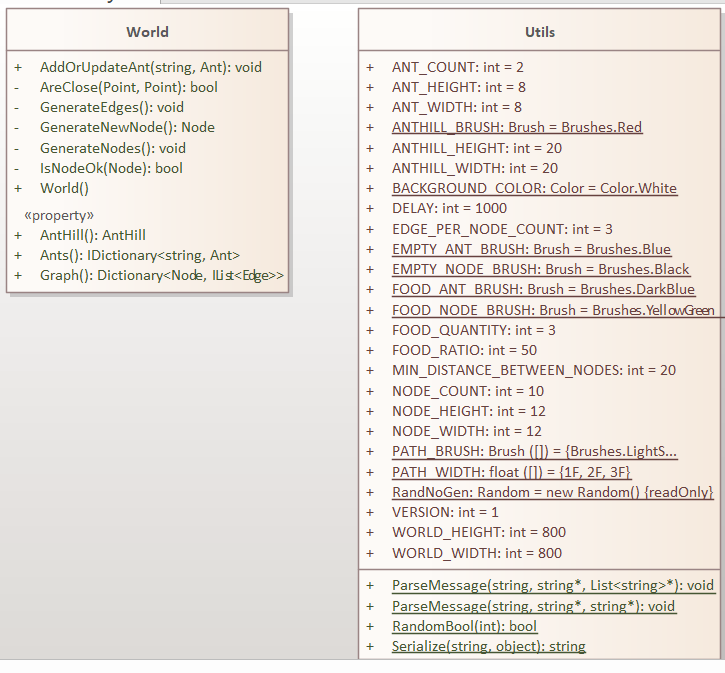
Proiectul a fost realizat utilizand limbajul de programare C# si Visual Studio 2019 ca mediu de dezvoltare.

* **Diagrama de clase:**

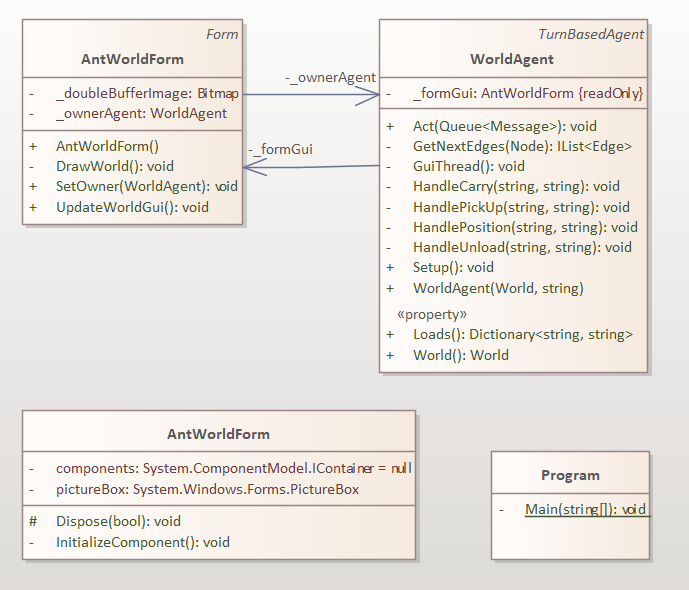


*Fig 1. Diagrama de clase(furnica agent, nod, muchie, baza)*

* Interfața *IDrawableEntity* conține funcția *Draw* care va fi folosită și implementată în clasele Node, Ant, și Edge pentru a desena atât nodurile și muchiile grafului, cât și furnicile;
* Clasa *Node* are următoarele proprietăți: poziția nodului, cantitatea de mâncare, iar ca metode, pe lângă cea care desenează nodul, metoda esențială este cea care descrește unitatea de mâncare;
* Clasa *Edge*, ca proprietăți, conține cele doua noduri de la capetele muchiei și valoarea asignată muchiei. De asemenea, metodele importante sunt *DecreaseWeight(),* *IncreaseWeight()* care vor descrește/ crește valoarea acesteia conform algoritmului;
* Clasa *AntHill* este baza de origine;
* Clasa *Ant* este o clasă care va fi folosita pentru a desena furnica si a regăsi poziția exactă în graf;
* Clasa *AntAgent* se folosește de clasa *Ant* pentru a reprezenta o furnică pe graf, iar metodele importante sunt: ridicarea unei unități de mâncare, schimbarea poziției pe graf, întoarcerea la bază.

*Fig 3. Diagrama de clase(lumea, clasa utilitara)*

* Clasa *Utils* creată și folosită pentru a parsa mesajele, serializarea obiectelor.
* Clasa *World* care conține graful care trebuie generat, lista de furnici și baza de origine. Metodele acestei clase,în principiu, sunt metode care generează nodurile, sau doar un singur nod, o muchie, adaugă o furnica sau updatează poziția unei furnici.



*Fig 2. Diagrama de clase(interfața grafică)*

* Clasa *WorldAgent* care se ocupă cu updatarea tuturor elementelor de pe graf, în special cu updatarea stării și poziției tuturor furnicile în funcție de ce acțiune desfașoară fiecare: caută mâncare, se află pe un nod unde găsește mâncare, ia mâncarea, cara mâncarea sau o lasă la bază sau pe un nod intermediar;
* Clasa *AntWorldForm* reprezintă interfața care se va updata în real time.
* **TODO: Alte aspect importante( implementarea versiunilor, metodele Setup/Act, cum facem serializarea etc):**

1. **Testare. Rezultate obtinute.**
2. **Concluzii**