Inferenţa prin enumerare în reţele bayesiene

Echipa: Atomei Georgiana (1410A)
Baciu H. Alexandru (1408B)

1. Descrierea problemei considerate

Se propune crearea unei aplicaţii care primeşte date despre anumite variabilele şi legăturile dintre ele, reprezentand structura şi parametrii unei reţele bayesiene, iar pe baza acestor date să se determine prin inferenţă prin enumerare rezultatele probabilistice dorite. De asemenea, se propune şi interogarea unui anumit nod din reţea pentru extragerea datelor în anumite situaţii în care acesta este implicat. Scopul principal al aplicaţiei este obţinerea de către utilizator a unor predicţii noi bazate pe date concrete, deja existente.

2. Aspecte teoretice privind algoritmul

Probabilităţi:

O probabilitate P(A) descrie numeric care sunt şansele ca un eveniment A să aibă loc sau indică procentul de adevăr al unei propoziţii. Aceasta poate lua valori între 0 şi 1, unde 0 indică imposibilitatea apariţiei evenimentului, iar 1 indică certitudinea aparitiei.

Probabilitatea conditionata P(A|B) este probabilitatea apariţiei evenimentului A, dată fiind apariţia evenimentului B.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Teorema lui Bayes

Teorema lui Bayes descrie probabilitatea unui eveniment date fiind condiţiile ce ar putea duce la apariţia evenimentului.

$$P(B|A) = P(A|B) * \frac{P(B)}{P(A)}$$

Independenţa şi independenţa condiţionată:

Doua evenimente sunt independente dacă unul nu influenţează rezultatele celuilalt eveniment.
Doua evenimente sunt independente conditionat dacă ele nu sunt independente şi rezultatele unuia pot influenţa cunoştinţele despre celălalt.

Reprezentarea distribuţiei comune de probabilitate

$$\mathrm{P}(x_1, x_2, .., x_n) = \prod_{i=1}^{n} P(x_i | \mathrm{parinti}(X_i))$$

Este adevărată doar dacă fiecare nod este independent condiţionat de predecesorii din şirul ordonat al nodurilor, daţi fiind părinţii nodului.

Reţele Bayesiane

Reţelele Bayesiane sunt modele grafice de probabilitate care pot fi folosite pentru crearea unor sisteme expert sau pentru extragerearea unor date precum predicţii, detecţii, diagnostice, etc. O retea bayesiana este un graf aciclic construit din noduri legate între ele. În majoritatea reţelelor un nod reprezintă o variabilă discretă sau continuă, iar o legătură dintre noduri indică că unul dintre acestea îl poate influenţa sau nu pe celălalt.

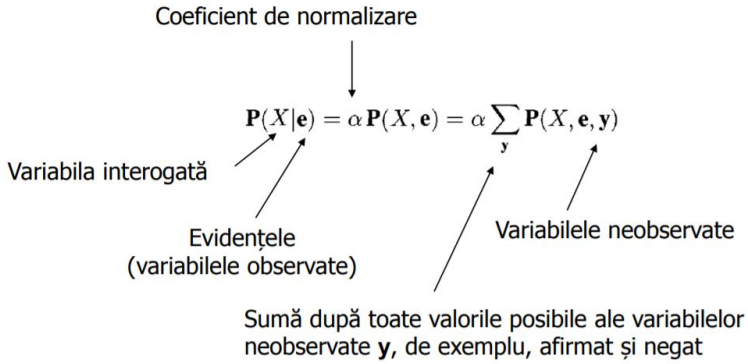
Sortarea topologică

Sortarea topologică a unui graf este o ordonare liniară a nodurilor sale astfel încât, pentru fiecare arc A → B, A apare înaintea lui B. Pentru o reţea bayesiană, sortarea topologică asigură faptul că nodurile părinte vor apărea înaintea nodurilor fiu.

3. Modalitatea de rezolvare

Pentru interogarea reţelei bayesiene oferită de utilizator prin noduri şi legături, acesta trebuie să construiască un fisier json care sa contina parametrii reţelei şi care urmează a fi interpretat şi analizat de aplicaţie cu scopul transformării acestuia într-un obiect necesar(graful reţelei) rulării corecte a programului. Pentru usurinta, aplicaţia oferă posibilitatea generării unui fişier tipar care poate fi ajustat de utilizator cu datele dorite.

Pe graful desenat pe interfaţă, utilizatorul poate interoga noduri pentru a afla diferite probabilitati. Prin interogarea unui nod se va apela algoritmul de inferenţă prin enumerare. Acest algoritm se bazează pe următoarea formulă de calcul:



Astfel algoritmul va calcula independent P(X_a, e, y), pentru fiecare valoare X_a pe care o poate avea nodul X, urmând ca la final aceste probabilitati sa fie normalizate astfel incat suma lor sa fie egala cu 1.

Pentru fiecare X_a, probabilitatea P(X_a, e, y) se calculeaza:

$$\mathrm{P}(X_a, \mathrm{e}, \mathrm{y}) = P(X_a | parinti(X_a)) * \prod_{i=1}^{n1} P(e_i | \mathrm{parinti}(e_i)) *$$

$$\prod_{j=1}^{n2} P(y_j | \text{parinti}(y_j))$$

4. Codul sursă, explicaţii şi comentarii

a. Sortarea topologică a nodurilor

```
/// <summary>
/// Method that sorts nodes ordered by their number of parents
/// </summary>
/// <returns></returns>
public List<Node> TopologicalSort()
{
    List<Node> networkNodes = new List<Node>();
    networkNodes.AddRange(Network.Nodes.Select(node=>node.DeepCopy()));

    List<Node> sorted = new List<Node>();
    Queue<Node> noParentsNodes = new Queue<Node>();

    networkNodes.ForEach(it =>
    {
        if (it.ParentsIds.Count == 0)
            noParentsNodes.Enqueue(it);
    });

    networkNodes.ForEach(node =>
    {
        if (noParentsNodes.Count == 0)
        {
            throw new Exception("Bayesian network's graph has at least one cycle.");
        }
        Node current = noParentsNodes.Dequeue();
        sorted.Add(Network.Nodes.Find(it=>it.Id==current.Id));
        networkNodes.ForEach(it =>
        {
            if (it.ParentsIds.Contains(current.Id))
            {
                it.ParentsIds.Remove(current.Id);
                if (it.ParentsIds.Count == 0)
                    noParentsNodes.Enqueue(it);
            }
        });

    });
    return sorted;
}
```

b. Metoda de calcul a probabilitatilor în urma interogării unui nod

Va returna lista de probabilitati normalizate calculate.
Se disting două cazuri: cand nodul este deja observat şi cand acesta nu este observat.
Dacă nodul a fost observat anterior, atunci lista va conţine pentru valoarea setată probabilitatea 1 şi 0 pentru restul valorilor posibile ale nodului.
Spre exemplu, pentru un nod X cu valori posibile [“A”, “B”, “C”], a cărui valoare observată a fost setată “A”, metoda va întoarce [1.0, 0.0, 0.0], unde 1.0 corespunde valorii “A”. Pentru cel de-al doilea caz, în care nodul nu are deja o valoare observată, se va parcurge lista valorilor posibile nodului şi pentru fiecare astfel de valoare se calculeaza probabilitatea prin regula de înmulţire a probabilitatilor în lanţ, înglobată în funcţia de calcul a probabilităţii unui nod dat prin index(descrisă mai jos).

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \text{parinti}(X_i))$$

```
/// <summary>
/// Method for calculating probabilities for the queried node
/// </summary>
/// <returns>List of all calculated probabilities, one for each possible
/// value for the queried node</returns>
public List<double> QueryNode()
{
    SortedNodes = TopologicalSort();
    List<double> probabilities = new List<double>();
    if (QueriedNode.ObservedValue != "None")            // if node was already observed, return observed probabilities
    {
        QueriedNode.NodeDomainValues.ForEach(val =>
        probabilities.Add(val != QueriedNode.ObservedValue ? 0.0 : 1.0));
    }

    else
    {
        //if queried node was not observed, than for each possible value, calculate its probability
        QueriedNode.NodeDomainValues.ForEach(value =>
        {
            QueriedNode.ObservedValue = value;
            var currentProb = CalculateProbability(0);
            probabilities.Add(currentProb);
        }
        );
        QueriedNode.ObservedValue = "None";
    }

    var probsSum = probabilities.Sum();

    var alpha = probsSum!=0?1.0 / probsSum:0.0;
    //normalize probabilities so that their sum will be 1
    var normalizedProbs = probabilities.Select(prob => (prob * alpha)).ToList();
    return normalizedProbs;
}
```

c. Metoda de calcul a probabilităţii unui nod dat prin index:

Metoda de calcul recursiv a probabilităţii prin regula de înmulţire (chain rule).

Şi aici se disting două cazuri: cel în care nodul curent este observat şi cel în care nodul nu este observat.
Pentru cazul în care nodul este observat, probabilitatea sa este preluată din tabelul de probabilitati cu care a fost iniţializat nodul, urmand a fi înmulţită cu probabilitatea următorului nod. Funcţia utilizată este cea de preluare a valorii unei probabilităţi din tabel descrisă mai jos.

În cazul în care nodul nu este observat, modalitatea de calcul a nodului curent şi implicit a tuturor nodurilor următoare.
Astfel, probabilitatea se va transforma într-o sumă de probabilităţi corespunzătoare pentru fiecare valoare posibilă din domeniul de valori a nodului curent, fapt reflectat în funcţia de calcul a probabilităţii unui nod neobservat.

```
/// <summary>
/// Recursive function for calculating probability for a node
/// </summary>
/// <param name="idx">Nodes's index in SortedNodes list</param>
/// <returns></returns>
double CalculateProbability(int idx)
{
    if (idx >= SortedNodes.Count)
        return 1.0;

    var node = SortedNodes[idx];

    if (node.ObservedValue != "None") //if node was observed
    {
        return CalculateProbForNode(node) * CalculateProbability(idx + 1);
    }
    else
    {
        return CalculateProbForNodeInSum(idx);
    }

}
```

d. Metoda de preluare a valorii unei probabilităţi din tabel:

Metoda primește ca parametru nodul pentru care se vrea preluarea probabilităţi şi returnează probabilitatea.
Dacă nodul nu are părinţi, probabilitatea preluată este una simpla, altfel dacă nodul are părinți probabilitatea va fi una conditionata de valorile acestora.

```
/// <summary>
/// Method that returns probability value from input table of probabilities
/// </summary>
/// <param name="node"></param>
/// <returns>Probability taken from input table of probabilities</returns>
double CalculateProbForNode(Node node)          //probability is taken from input probabilities' table
{
    var currentProb = 1.0;
    if (node.ParentsIds.Count == 0)     //if node has no parents
    {
        var value = $" {node.Name}={node.ObservedValue}";
        currentProb = node.Probabilities[value][0];
    }
    else
    {
        //if node has parents, the probability is a conditioned
        var value = "";
        node.ParentsIds.ForEach(parentId =>
        {
            var parentFound = Network.Nodes.Find(parent => parent.Id == parentId);
            value += $" {parentFound.Name}={parentFound.ObservedValue}";
        }
        );
        int idx = node.NodeDomainValues.FindIndex(it => it == node.ObservedValue);
        currentProb = node.Probabilities[value][idx];
    }
    return currentProb;
}
```

e. Metoda de calcul a probabilităţii unui nod incepand cu un nod neobservat:

Se disting două cazuri: nodul curent este observat sau nodul curent nu este observat.
În primul caz, probabilitatea se preia din tabel şi se inmulteste cu probabilitatea următorului nod.
În cel de-al doilea caz, pentru fiecare valoare din domeniu se preia valoarea din tabel şi se inmulteste cu probabilitatile nodurilor următoare.Toate aceste probabilitati rezultate se insumeaza, şi valoarea finală va reprezenta probabilitatea nodurilor, plecând de la nodul curent.

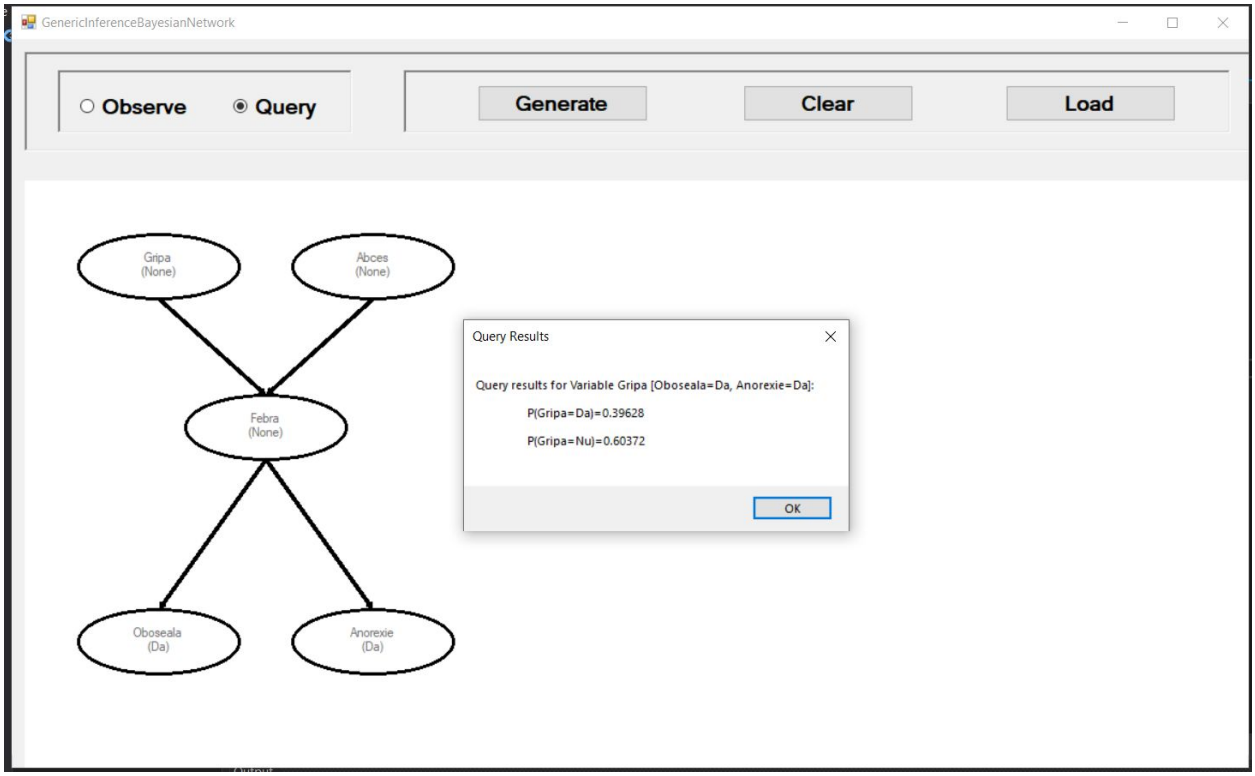
```
/// <summary>
/// Method for calculating probbaility when a node is not observed
/// For this node, every possible domain value is taken into consideration
/// </summary>
/// <param name="idx">Node's index in sortedNodes list </param>
/// <returns></returns>

double CalculateProbForNodeInSum(int idx)
{
    if (idx >= SortedNodes.Count)
        return 1.0;

    Node node = SortedNodes[idx];
    if(node.ObservedValue != "None")
    {
        return CalculateProbForNode(node)*CalculateProbForNodeInSum(idx+1);
    }

    double Sum = 0.0;

    node.NodeDomainValues.ForEach(value =>
    {
        node.ObservedValue = value;
        var currentProb = CalculateProbForNode(node);
        Sum += currentProb * CalculateProbForNodeInSum(idx + 1);
    });
    node.ObservedValue = "None";
    return Sum;
}
```

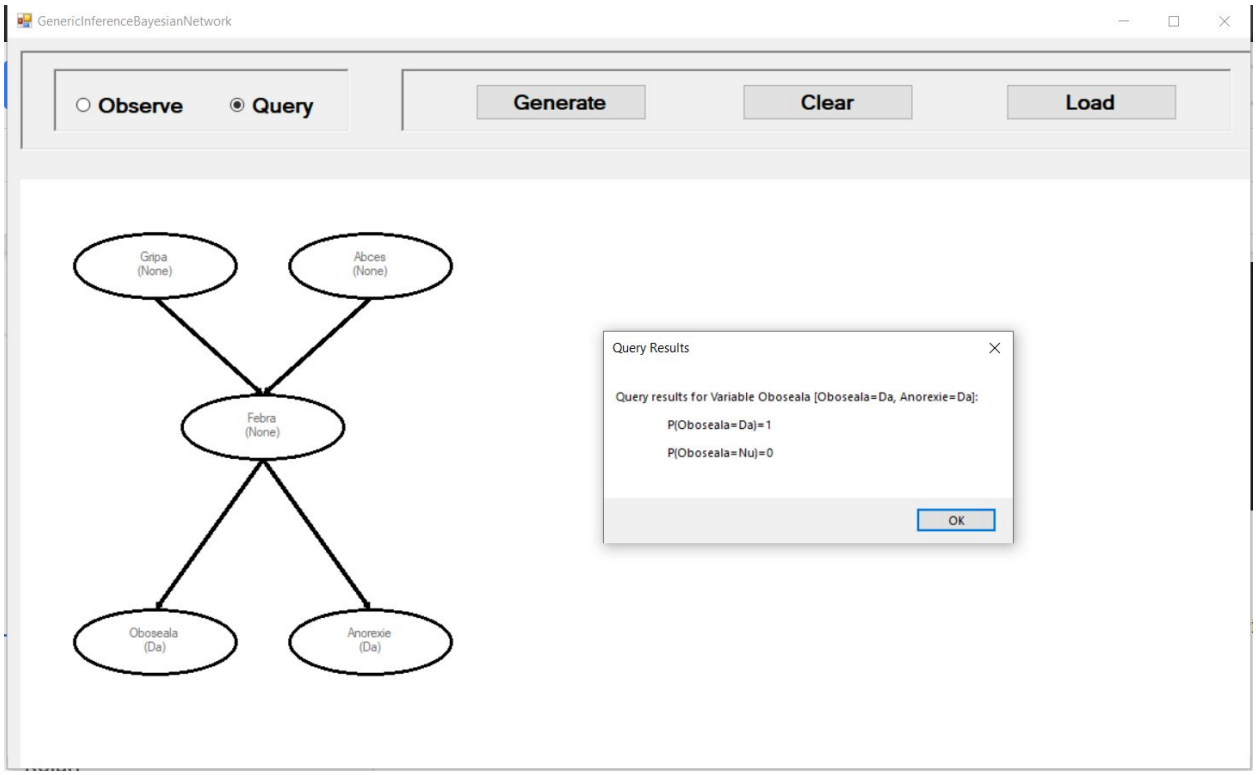
f. Structuri de date utilizate

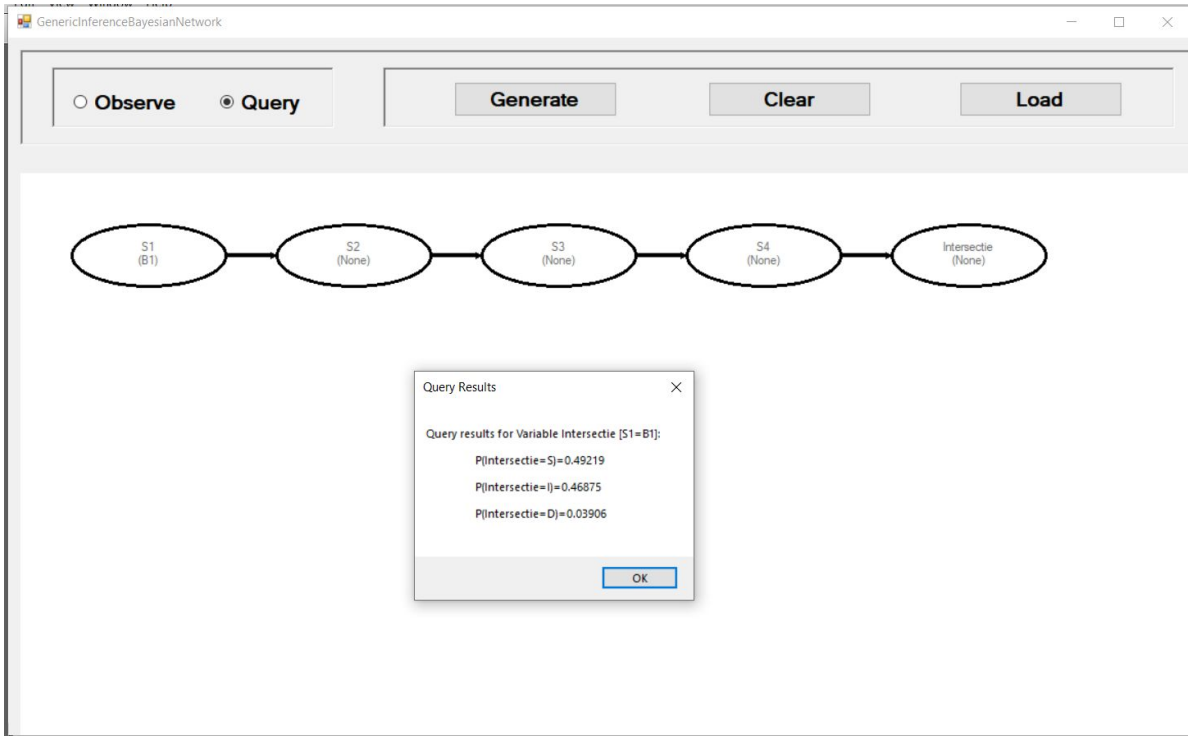
```
public class Node
{
    public long Id { get; set; }

    public String Name { get; set; }

    public int PosX { get; set; }

    public int PosY { get; set; }

    public List<long> ParentsIds { get; set; }

    public List<String> NodeDomainValues { get; set; }

    public String ObservedValue { get; set; }

    public Dictionary<String, List<Double>> Probabilities { get; set; }

    public Node DeepCopy()...

    public void GenerateProbabilities(Network network)...

    private static IEnumerable<IEnumerable<String>> CartesianProduct<String>
        (IEnumerable<IEnumerable<String>> sequences)...

}
```

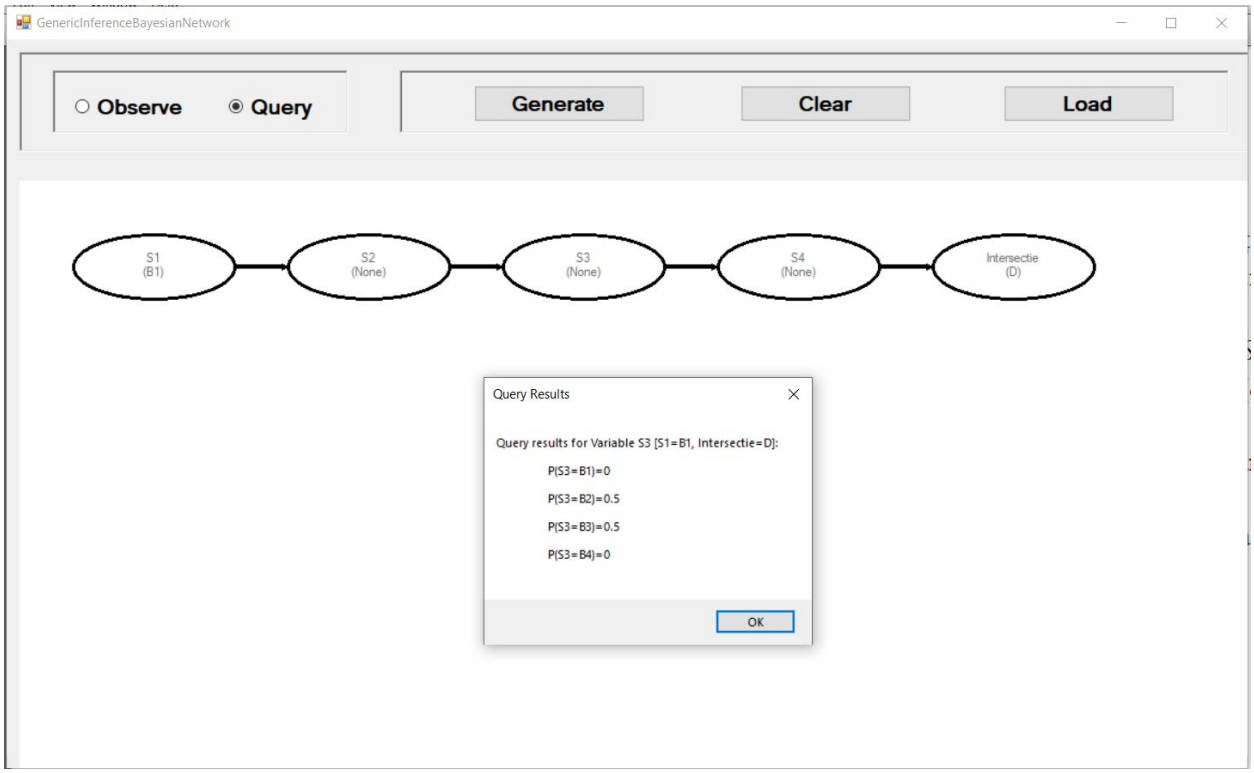
```
public class Network
{
    public String Name { get; set; }

    public List<Node> Nodes { get; set; }

}
```

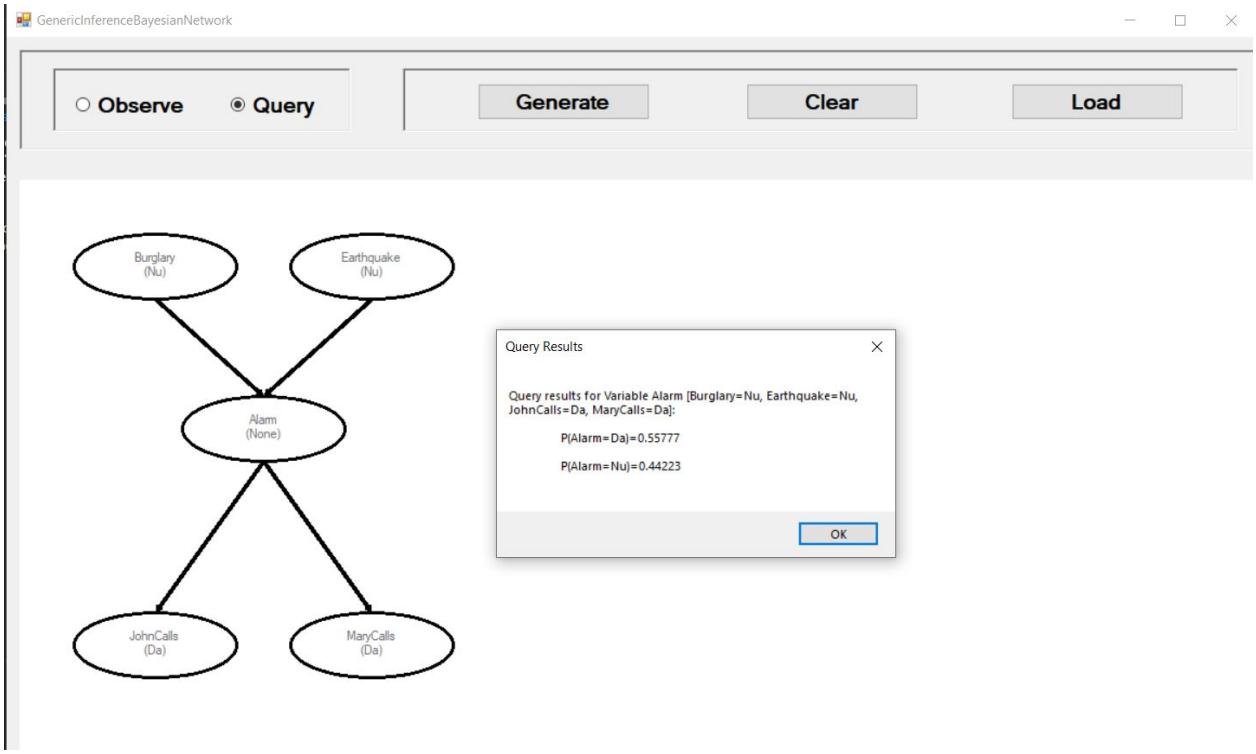
5. Rezultatele obţinute în diverse situaţii, capturi ecran şi comentarii asupra rezultatelor


GenericInferenceBayesianNetwork
Observe
Query
Generate
Clear
Load
Gripa
(None)
Abces
(None)
Febra
(None)
Oboseala
(Da)
Anorexie
(Da)
Query Results
Query results for Variable Gripa [Oboseala=Da, Anorexie=Da]:
P(Gripa=Da)=0.39628
P(Gripa=Nu)=0.60372
OK


GenericInferenceBayesianNetwork
Observe
Query
Generate
Clear
Load
Gripa
(None)
Abces
(None)
Febra
(None)
Oboseala
(Da)
Anorexie
(Da)
Query Results
Query results for Variable Oboseala [Oboseala=Da, Anorexie=Da]:
P(Oboseala=Da)=1
P(Oboseala=Nu)=0
OK


GenericInferenceBayesianNetwork
Observe
Query
Generate
Clear
Load
S1
(B1)
S2
(None)
S3
(None)
S4
(None)
Intersectie
(None)
Query Results
Query results for Variable Intersectie [S1=B1]:
P(Intersectie=S)=0.49219
P(Intersectie=I)=0.46875
P(Intersectie=D)=0.03906
OK


GenericInferenceBayesianNetwork
Observe
Query
Generate
Clear
Load
S1
(B1)
S2
(None)
S3
(None)
S4
(None)
Intersectie
(D)
Query Results
Query results for Variable S3 [S1=B1, Intersectie=D]:
P(S3=B1)=0
P(S3=B2)=0.5
P(S3=B3)=0.5
P(S3=B4)=0
OK



6. Concluzii

Aşa cum a fost dorit, programul rezolva interogari asupra oricărei reţele, cu condiţia respectării tipului de format a fişierului de intrare.

Imbunatatiri:

Printre îmbunătăţirile care ar putea fi aduse se numără implementarea sortarii topologice cu ajutorul unei cozi cu prioritate în schimbul uneia normale pentru a îmbunătăţi calculul probabilitatii.
În acest sens, amintit exemplul în care două noduri cu acelaşi număr de părinţi, unul avand valoare observată şi celălalt nu, ar fi optim la calcul, ca nodul observat sa fie considerat primul în cazul calculului, fapt ce nu este garantat deocamdată de sortarea topologică.

7. Bibliografie

a. https://en.wikipedia.org/wiki/Probability
b. http://florinleon.byethost24.com/Curs_IA/IA10_ReteleBayesiene.pdf
c. https://www.bayesserver.com/docs/introduction/bayesian-networks

8. Roluri

Atomei Georgiana	**Baciu H. Alexandru**
Desenarea grafică a reţelei	Creare interfaţă, generare structură reţea şi generarea fişier tipar
Metode de calcul a probabilitatilor prin inferenţă prin enumerare	Sortare topologică