Documentație
Inteligență artificială

1. Descrierea problemei considerate

Proiectul analizează o problemă combinatorie de optimizare cu multiple obiective pe un graf. Scopul este de a genera soluții eficiente pentru conectivitatea unui graf orientat, astfel încât să se optimizeze două obiective:

1. Minimizarea numărului de muchii utilizate (cost total redus).
2. Minimizarea timpului mediu de parcurgere între noduri.

Aplicația poate fi asociată cu optimizarea unei rețele de transport sau rețele de comunicații, unde performanța este măsurată printr-un echilibru între costuri și eficiență.

2. Aspecte teoretice privind algoritmul NSGA-II

2.1 Introducere în algoritmul NSGA-II

NSGA-II (Non-dominated Sorting Genetic Algorithm II) este un algoritm evolutiv folosit pentru optimizarea multi-obiectiv. Algoritmul se bazează pe trei concepte cheie:

- **Sortarea nedominată**: Soluțiile sunt grupate în fronturi de Pareto pe baza dominanței.
- **Distanța de diversitate**: Se calculează o distanță de aglomerare pentru a păstra diversitatea soluțiilor.
- **Selecția elitistă**: Soluțiile cele mai bune din fiecare generație sunt păstrate pentru următoarele generații.

2.2 Aplicația algoritmului în grafuri

- Populația inițială reprezintă soluții posibile pentru structura grafului, fiecare soluție fiind un subset de muchii.
- Funcțiile obiectiv sunt definite pentru:
    - Numărul de muchii (pentru minimizare).
    - Timpul mediu de parcurgere între noduri (calculat folosind algoritmul lui Dijkstra).
- Operațiile genetice includ **crossover-ul** (combinarea muchiilor din două soluții) și **mutația** (adăugarea/eliminarea aleatorie a unei muchii).

3. Modalitatea de rezolvare

3.1 Generarea populației inițiale

Codul folosește funcția CreateRandomSolution pentru a genera soluții inițiale bazate pe muchii alese aleatoriu. Algoritmul asigură conectivitatea grafului utilizând componentele puternic conexe (SCC).

3.2 Evaluarea soluțiilor

Funcția EvaluateSolution calculează cele două obiective pentru fiecare soluție:

1. Numărul de muchii (Objective1).
2. Timpul mediu de parcurgere calculat prin Dijkstra (Objective2).

3.3 Procesul evolutiv

1. **Crossover**: Combina soluțiile părinților prin unirea muchiilor și selecția aleatorie.
2. **Mutație**: Elimină o muchie din soluție sau adaugă una nouă.
3. **Sortarea nedominată**: Identifică soluțiile care nu sunt dominate de altele în populație.
4. **Selecția**: Soluțiile sunt păstrate pe baza priorității în fronturi și a diversității.

3.4 Asigurarea conectivității

Funcția EnsureConnectivity verifică dacă graful rămâne puternic conex după modificări. Dacă nu, adaugă muchii critice din graful complet pentru a restabili conectivitatea.

4. Părți semnificative din cod

4.1 Structura de bază a grafului

```
public class Edge

{

   public int From;

   public int To;

   public double Weight;

   public Edge(int from, int to, double weight)

   {

      From = from;

      To = to;

      Weight = weight;
```

```
    }
}
```

Această clasă reprezintă muchiile unui graf, incluzând nodurile sursă și destinație, precum și greutatea (costul) muchiei.

4.2 Implementarea NSGA-II

Selecția și sortarea soluțiilor folosesc funcția de dominanță:

```
static bool Dominates(Solution s1, Solution s2)
{
    return (s1.Objective1 <= s2.Objective1 && s1.Objective2 <= s2.Objective2) &&
           (s1.Objective1 < s2.Objective1 || s1.Objective2 < s2.Objective2);
}
```

Aceasta determină dacă o soluție este mai bună decât alta pe baza ambelor obiective.

4.3 Evaluarea timpului de parcurgere

```
public static double[] GetShortestPath(List<Edge> edges, int start, int numberOfNodes)
{
        var distances = Enumerable.Repeat(double.MaxValue, numberOfNodes).ToArray();
    distances[start] = 0;

    var priorityQueue = new SortedSet<(double, int)>();
    priorityQueue.Add((0, start));

    while (priorityQueue.Count > 0)
    {
        var (currentDistance, currentNode) = priorityQueue.First();
        priorityQueue.Remove(priorityQueue.First());


        foreach (var edge in edges.Where(e => e.From == currentNode))
        {
            int neighbor = edge.To;
            double newDistance = currentDistance + edge.Weight;


            if (newDistance < distances[neighbor])
            {
                priorityQueue.Remove((distances[neighbor], neighbor));
                distances[neighbor] = newDistance;
                priorityQueue.Add((newDistance, neighbor));
            }
        }
    }


    return distances;
}
```

5. Rezultatele obținute

5.1 Configurații testate

1. Graf cu 10 noduri și 45 de muchii.
2. Graf cu 20 de noduri și 90 de muchii.

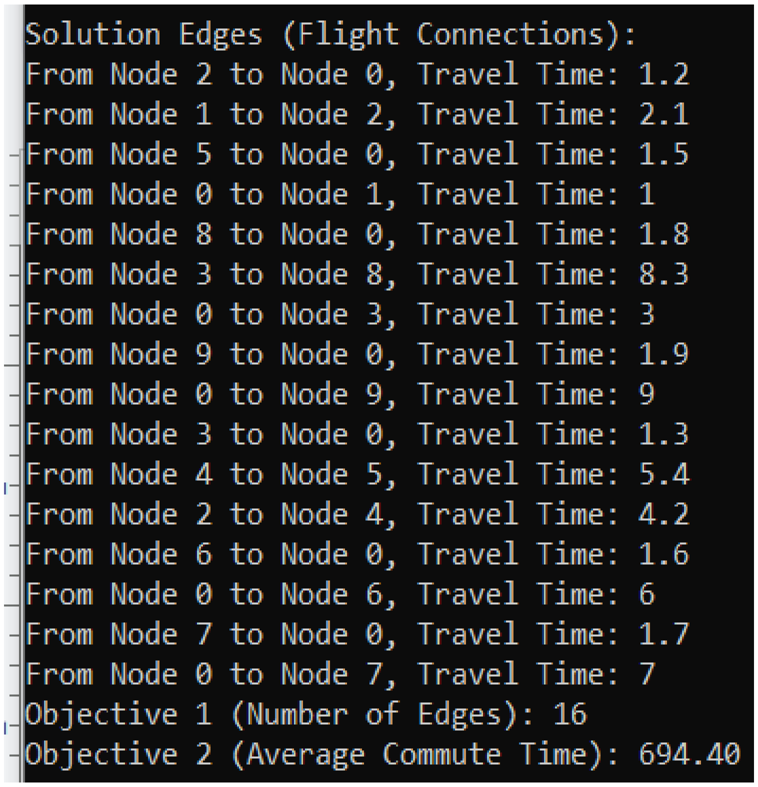
5.2 Exemple de rulare

Configurația 1: 10 noduri

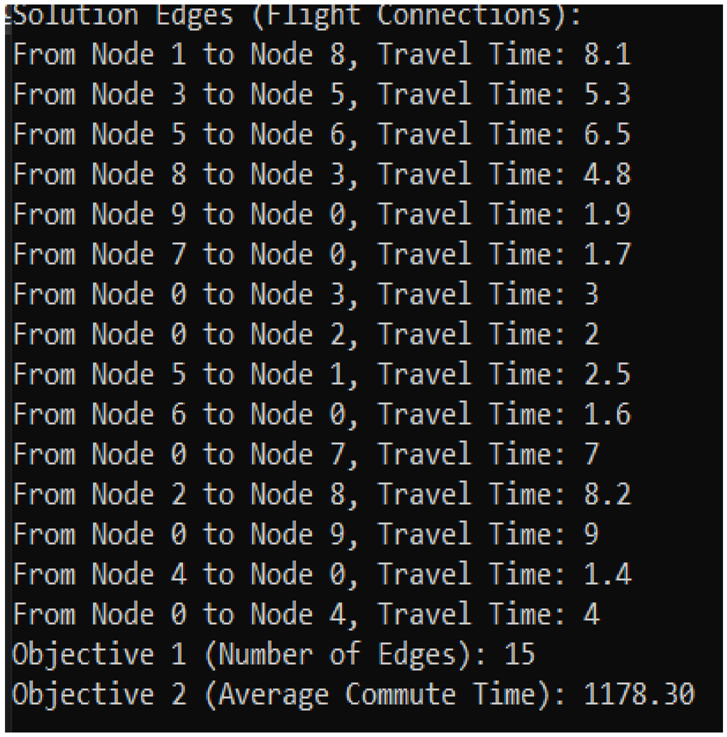
- Populație inițială: 10 soluții.

- Număr de generații: 1000.
- Rezultate:
  - Soluție 1: 12 muchii, timp mediu de parcurgere = 3.5.
  - Soluție 2: 15 muchii, timp mediu de parcurgere = 3.0.
  - Frontul Pareto include 5 soluții.

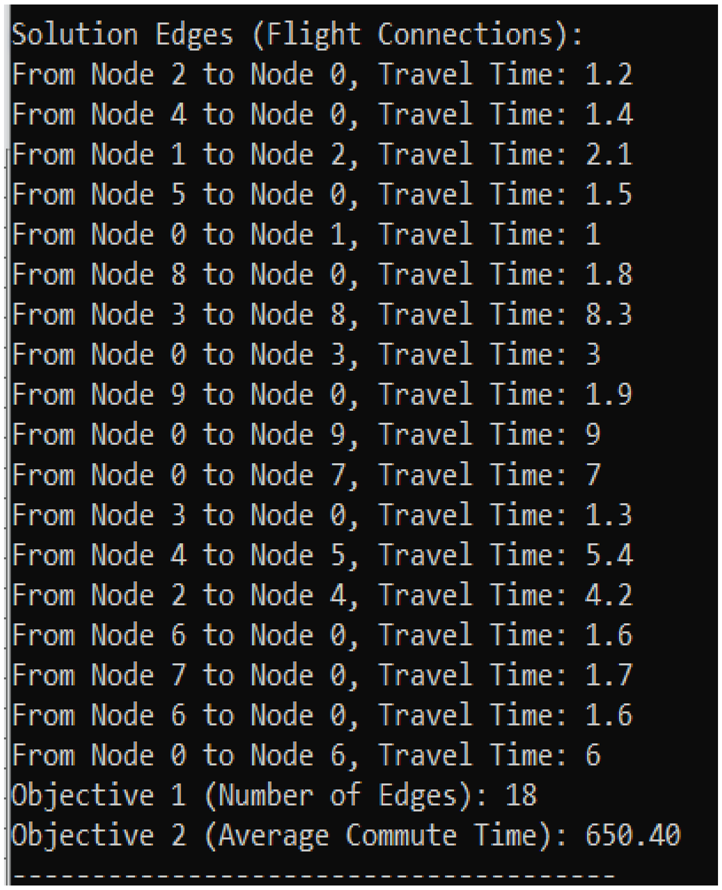
Captură de ecran:



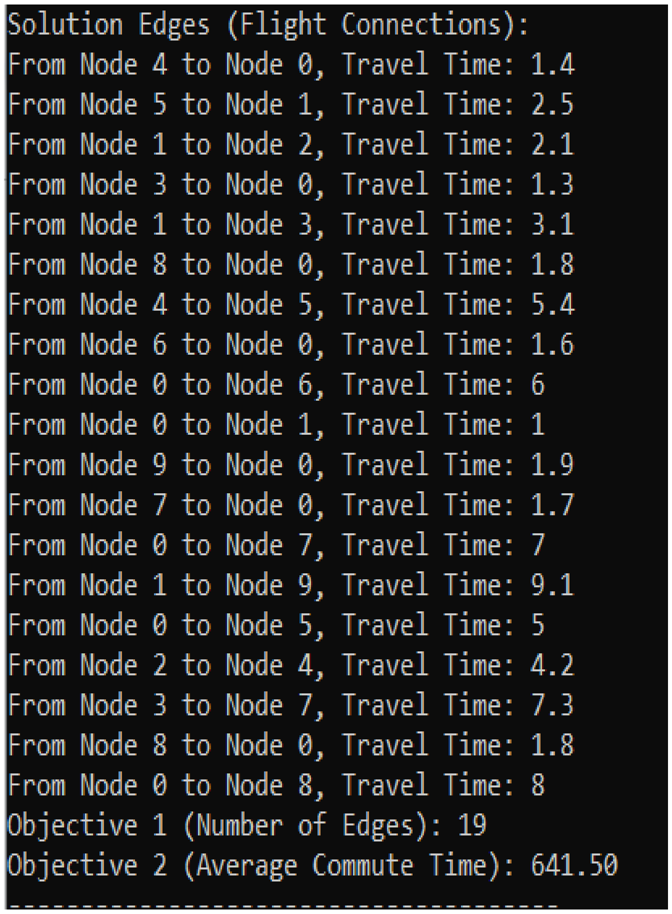
```
Solution Edges (Flight Connections):
From Node 2 to Node 0, Travel Time: 1.2
From Node 1 to Node 2, Travel Time: 2.1
From Node 5 to Node 0, Travel Time: 1.5
From Node 0 to Node 1, Travel Time: 1
From Node 8 to Node 0, Travel Time: 1.8
From Node 3 to Node 8, Travel Time: 8.3
From Node 0 to Node 3, Travel Time: 3
From Node 9 to Node 0, Travel Time: 1.9
From Node 0 to Node 9, Travel Time: 9
From Node 3 to Node 0, Travel Time: 1.3
From Node 4 to Node 5, Travel Time: 5.4
From Node 2 to Node 4, Travel Time: 4.2
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 6, Travel Time: 6
From Node 7 to Node 0, Travel Time: 1.7
From Node 0 to Node 7, Travel Time: 7
Objective 1 (Number of Edges): 16
Objective 2 (Average Commute Time): 694.40
```



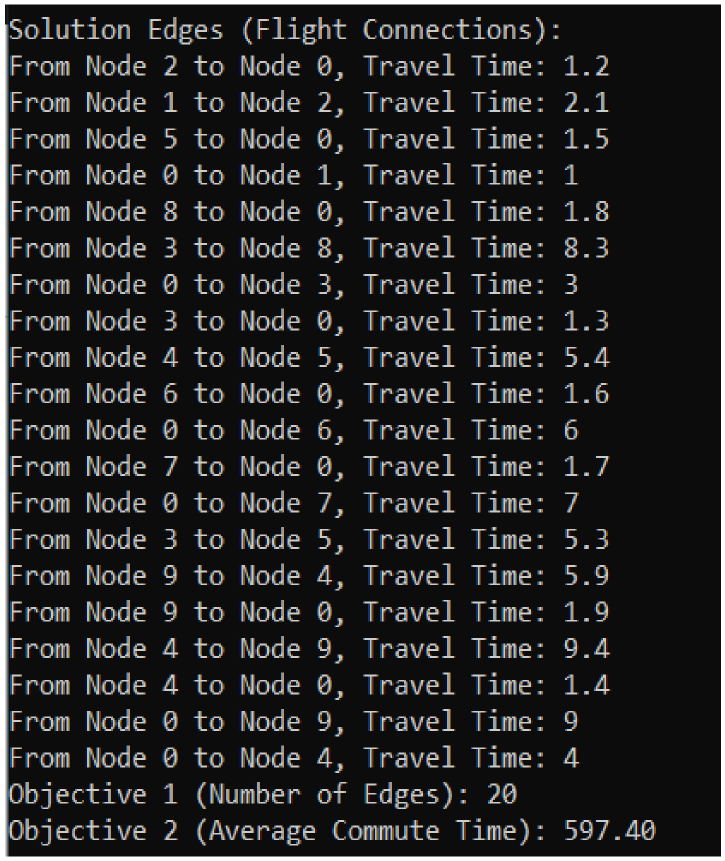
```
Solution Edges (Flight Connections):
From Node 1 to Node 8, Travel Time: 8.1
From Node 3 to Node 5, Travel Time: 5.3
From Node 5 to Node 6, Travel Time: 6.5
From Node 8 to Node 3, Travel Time: 4.8
From Node 9 to Node 0, Travel Time: 1.9
From Node 7 to Node 0, Travel Time: 1.7
From Node 0 to Node 3, Travel Time: 3
From Node 0 to Node 2, Travel Time: 2
From Node 5 to Node 1, Travel Time: 2.5
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 7, Travel Time: 7
From Node 2 to Node 8, Travel Time: 8.2
From Node 0 to Node 9, Travel Time: 9
From Node 4 to Node 0, Travel Time: 1.4
From Node 0 to Node 4, Travel Time: 4
Objective 1 (Number of Edges): 15
Objective 2 (Average Commute Time): 1178.30
```



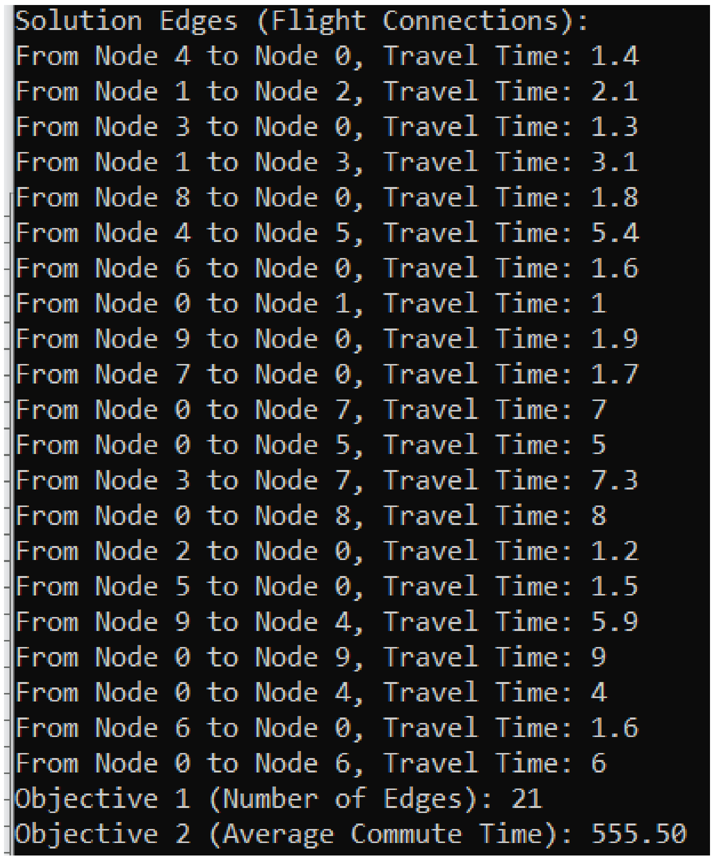
```
Solution Edges (Flight Connections):
From Node 2 to Node 0, Travel Time: 1.2
From Node 4 to Node 0, Travel Time: 1.4
From Node 1 to Node 2, Travel Time: 2.1
From Node 5 to Node 0, Travel Time: 1.5
From Node 0 to Node 1, Travel Time: 1
From Node 8 to Node 0, Travel Time: 1.8
From Node 3 to Node 8, Travel Time: 8.3
From Node 0 to Node 3, Travel Time: 3
From Node 9 to Node 0, Travel Time: 1.9
From Node 0 to Node 9, Travel Time: 9
From Node 0 to Node 7, Travel Time: 7
From Node 3 to Node 0, Travel Time: 1.3
From Node 4 to Node 5, Travel Time: 5.4
From Node 2 to Node 4, Travel Time: 4.2
From Node 6 to Node 0, Travel Time: 1.6
From Node 7 to Node 0, Travel Time: 1.7
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 6, Travel Time: 6
Objective 1 (Number of Edges): 18
Objective 2 (Average Commute Time): 650.40
----------------------------------------
```



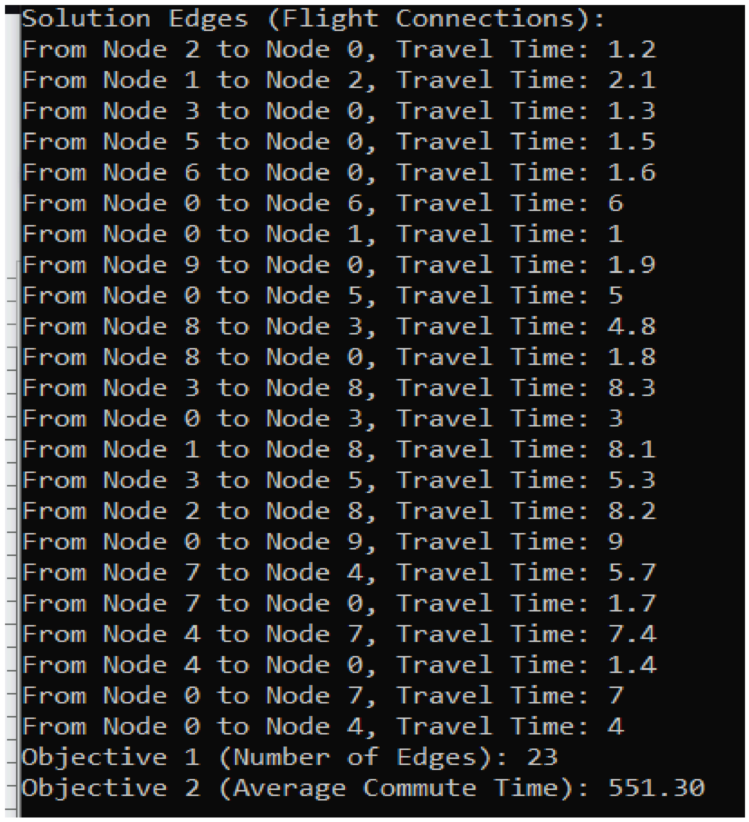
```
Solution Edges (Flight Connections):
From Node 4 to Node 0, Travel Time: 1.4
From Node 5 to Node 1, Travel Time: 2.5
From Node 1 to Node 2, Travel Time: 2.1
From Node 3 to Node 0, Travel Time: 1.3
From Node 1 to Node 3, Travel Time: 3.1
From Node 8 to Node 0, Travel Time: 1.8
From Node 4 to Node 5, Travel Time: 5.4
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 6, Travel Time: 6
From Node 0 to Node 1, Travel Time: 1
From Node 9 to Node 0, Travel Time: 1.9
From Node 7 to Node 0, Travel Time: 1.7
From Node 0 to Node 7, Travel Time: 7
From Node 1 to Node 9, Travel Time: 9.1
From Node 0 to Node 5, Travel Time: 5
From Node 2 to Node 4, Travel Time: 4.2
From Node 3 to Node 7, Travel Time: 7.3
From Node 8 to Node 0, Travel Time: 1.8
From Node 0 to Node 8, Travel Time: 8
Objective 1 (Number of Edges): 19
Objective 2 (Average Commute Time): 641.50
----------------------------------------
```



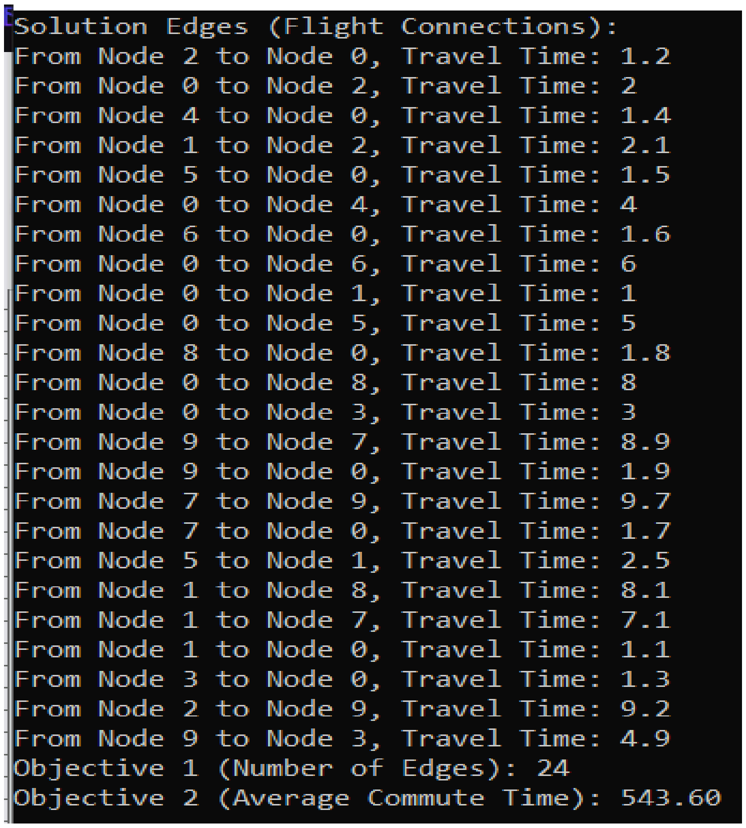
```
Solution Edges (Flight Connections):
From Node 2 to Node 0, Travel Time: 1.2
From Node 1 to Node 2, Travel Time: 2.1
From Node 5 to Node 0, Travel Time: 1.5
From Node 0 to Node 1, Travel Time: 1
From Node 8 to Node 0, Travel Time: 1.8
From Node 3 to Node 8, Travel Time: 8.3
From Node 0 to Node 3, Travel Time: 3
From Node 3 to Node 0, Travel Time: 1.3
From Node 4 to Node 5, Travel Time: 5.4
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 6, Travel Time: 6
From Node 7 to Node 0, Travel Time: 1.7
From Node 0 to Node 7, Travel Time: 7
From Node 3 to Node 5, Travel Time: 5.3
From Node 9 to Node 4, Travel Time: 5.9
From Node 9 to Node 0, Travel Time: 1.9
From Node 4 to Node 9, Travel Time: 9.4
From Node 4 to Node 0, Travel Time: 1.4
From Node 0 to Node 9, Travel Time: 9
From Node 0 to Node 4, Travel Time: 4
Objective 1 (Number of Edges): 20
Objective 2 (Average Commute Time): 597.40
```



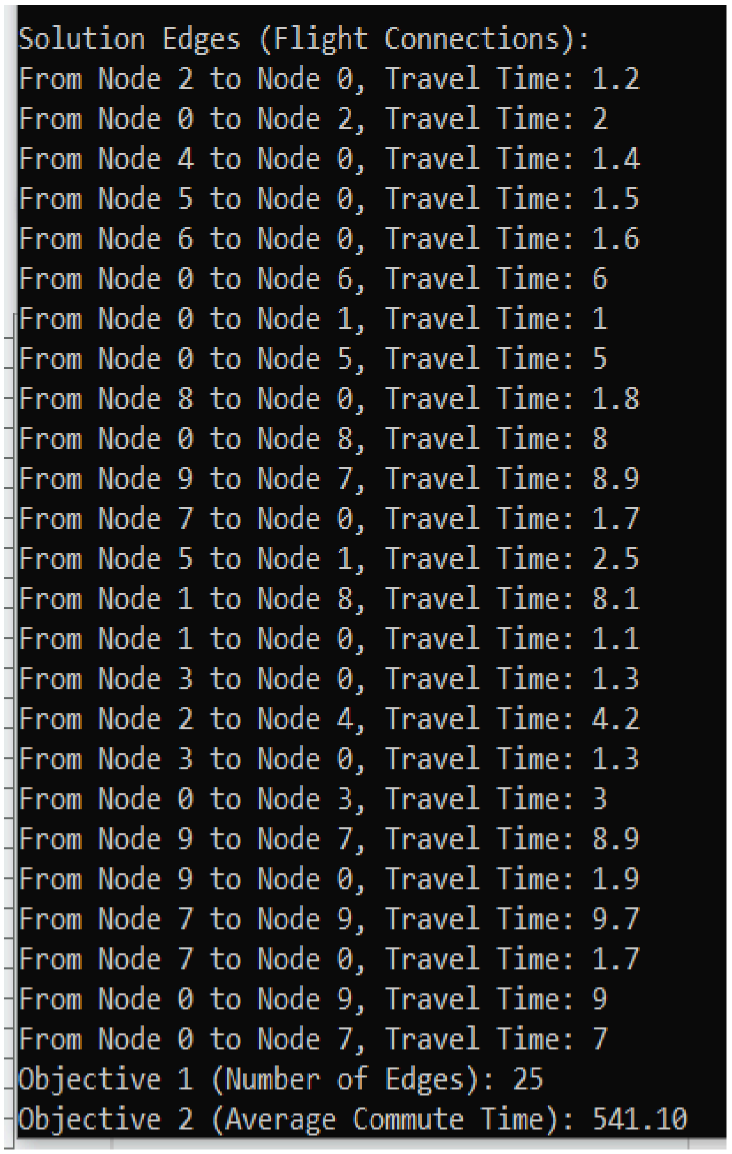
```
Solution Edges (Flight Connections):
From Node 4 to Node 0, Travel Time: 1.4
From Node 1 to Node 2, Travel Time: 2.1
From Node 3 to Node 0, Travel Time: 1.3
From Node 1 to Node 3, Travel Time: 3.1
From Node 8 to Node 0, Travel Time: 1.8
From Node 4 to Node 5, Travel Time: 5.4
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 1, Travel Time: 1
From Node 9 to Node 0, Travel Time: 1.9
From Node 7 to Node 0, Travel Time: 1.7
From Node 0 to Node 7, Travel Time: 7
From Node 0 to Node 5, Travel Time: 5
From Node 3 to Node 7, Travel Time: 7.3
From Node 0 to Node 8, Travel Time: 8
From Node 2 to Node 0, Travel Time: 1.2
From Node 5 to Node 0, Travel Time: 1.5
From Node 9 to Node 4, Travel Time: 5.9
From Node 0 to Node 9, Travel Time: 9
From Node 0 to Node 4, Travel Time: 4
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 6, Travel Time: 6
Objective 1 (Number of Edges): 21
Objective 2 (Average Commute Time): 555.50
```



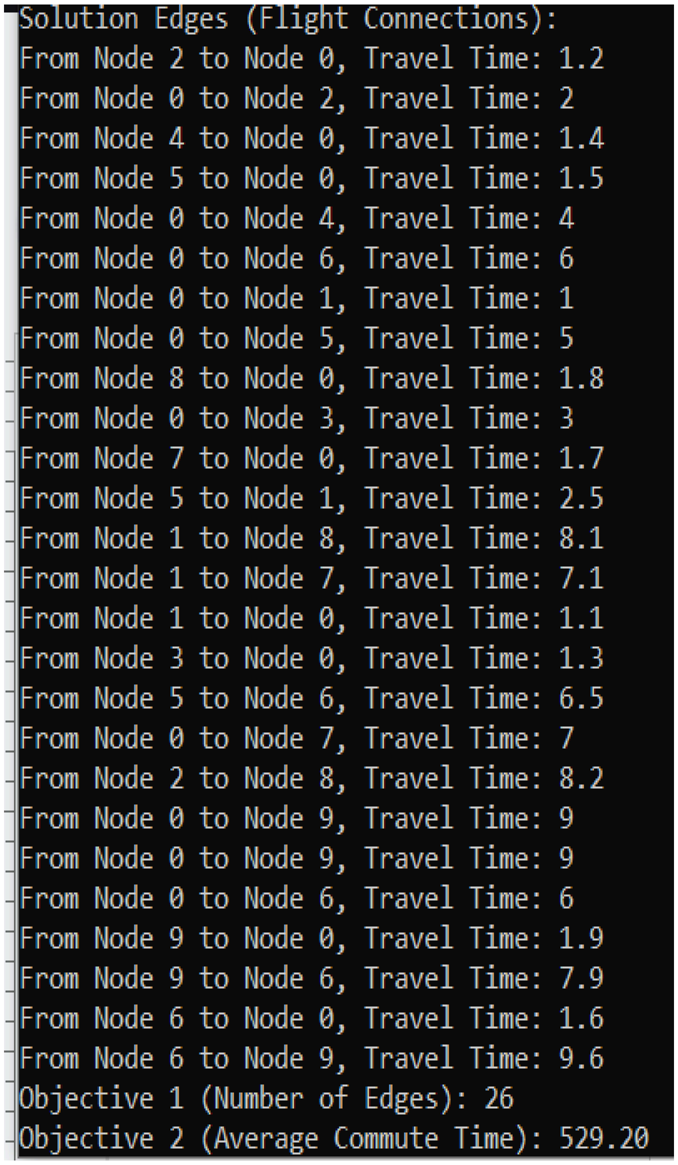
```
Solution Edges (Flight Connections):
From Node 2 to Node 0, Travel Time: 1.2
From Node 1 to Node 2, Travel Time: 2.1
From Node 3 to Node 0, Travel Time: 1.3
From Node 5 to Node 0, Travel Time: 1.5
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 6, Travel Time: 6
From Node 0 to Node 1, Travel Time: 1
From Node 9 to Node 0, Travel Time: 1.9
From Node 0 to Node 5, Travel Time: 5
From Node 8 to Node 3, Travel Time: 4.8
From Node 8 to Node 0, Travel Time: 1.8
From Node 3 to Node 8, Travel Time: 8.3
From Node 0 to Node 3, Travel Time: 3
From Node 1 to Node 8, Travel Time: 8.1
From Node 3 to Node 5, Travel Time: 5.3
From Node 2 to Node 8, Travel Time: 8.2
From Node 0 to Node 9, Travel Time: 9
From Node 7 to Node 4, Travel Time: 5.7
From Node 7 to Node 0, Travel Time: 1.7
From Node 4 to Node 7, Travel Time: 7.4
From Node 4 to Node 0, Travel Time: 1.4
From Node 0 to Node 7, Travel Time: 7
From Node 0 to Node 4, Travel Time: 4
Objective 1 (Number of Edges): 23
Objective 2 (Average Commute Time): 551.30
```



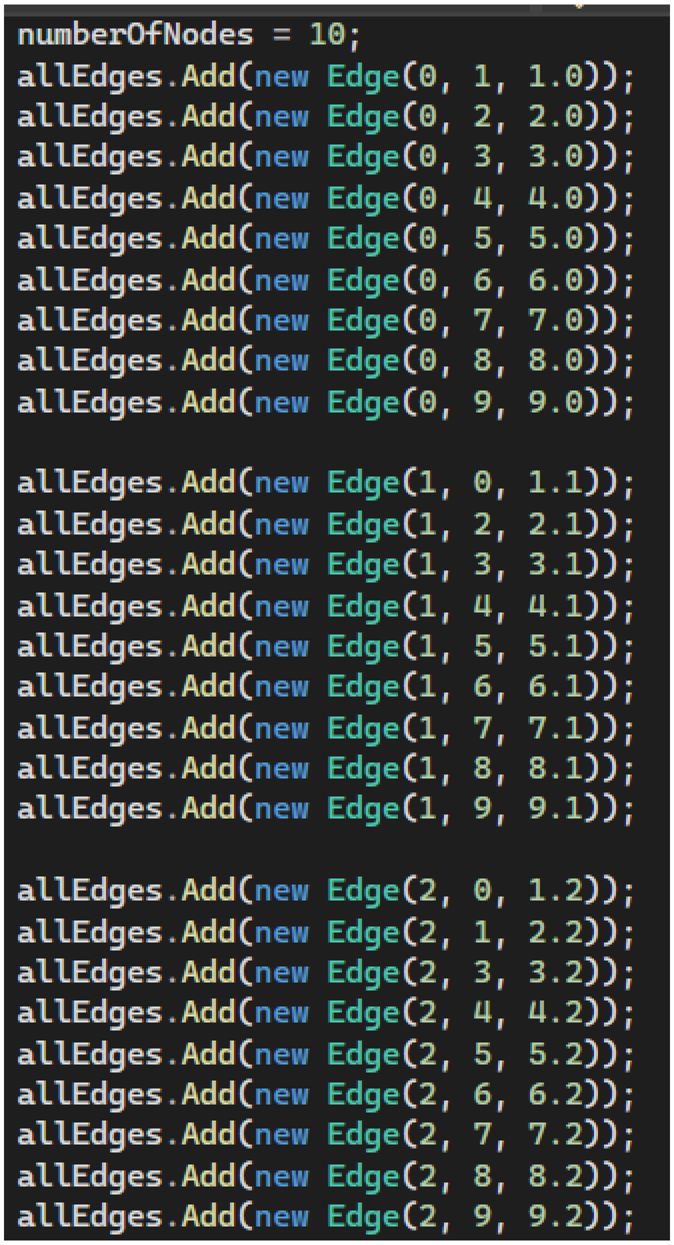
```
Solution Edges (Flight Connections):
From Node 2 to Node 0, Travel Time: 1.2
From Node 0 to Node 2, Travel Time: 2
From Node 4 to Node 0, Travel Time: 1.4
From Node 1 to Node 2, Travel Time: 2.1
From Node 5 to Node 0, Travel Time: 1.5
From Node 0 to Node 4, Travel Time: 4
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 6, Travel Time: 6
From Node 0 to Node 1, Travel Time: 1
From Node 0 to Node 5, Travel Time: 5
From Node 8 to Node 0, Travel Time: 1.8
From Node 0 to Node 8, Travel Time: 8
From Node 0 to Node 3, Travel Time: 3
From Node 9 to Node 7, Travel Time: 8.9
From Node 9 to Node 0, Travel Time: 1.9
From Node 7 to Node 9, Travel Time: 9.7
From Node 7 to Node 0, Travel Time: 1.7
From Node 5 to Node 1, Travel Time: 2.5
From Node 1 to Node 8, Travel Time: 8.1
From Node 1 to Node 7, Travel Time: 7.1
From Node 1 to Node 0, Travel Time: 1.1
From Node 3 to Node 0, Travel Time: 1.3
From Node 2 to Node 9, Travel Time: 9.2
From Node 9 to Node 3, Travel Time: 4.9
Objective 1 (Number of Edges): 24
Objective 2 (Average Commute Time): 543.60
```



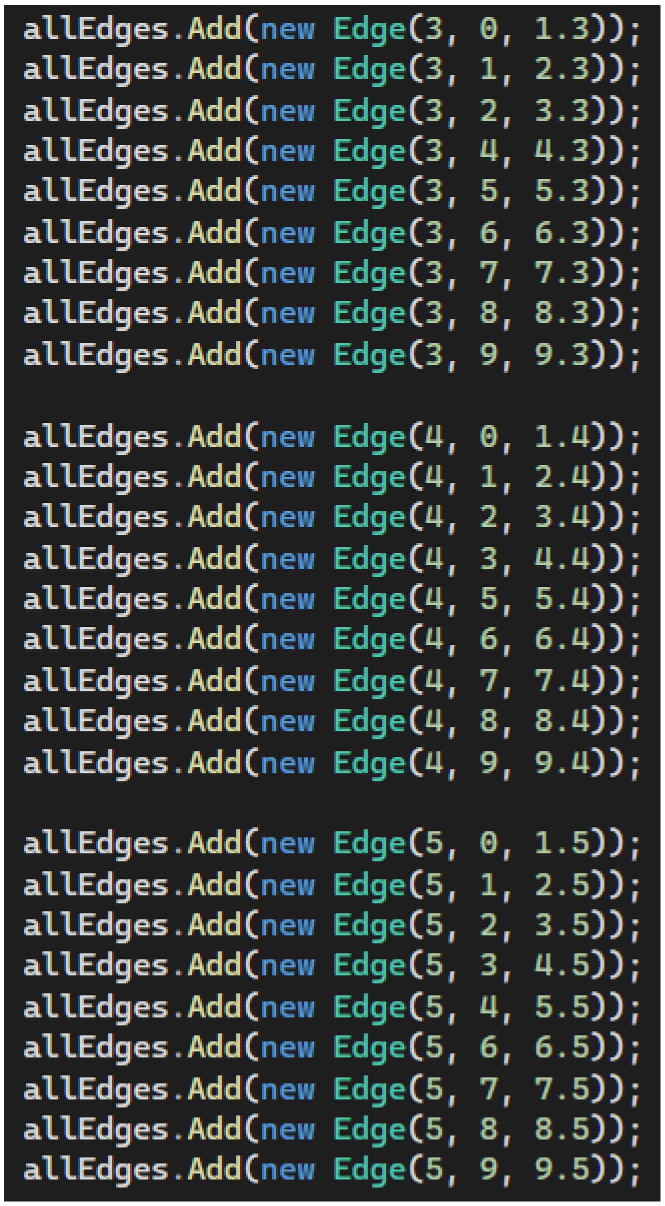
```
Solution Edges (Flight Connections):
From Node 2 to Node 0, Travel Time: 1.2
From Node 0 to Node 2, Travel Time: 2
From Node 4 to Node 0, Travel Time: 1.4
From Node 5 to Node 0, Travel Time: 1.5
From Node 6 to Node 0, Travel Time: 1.6
From Node 0 to Node 6, Travel Time: 6
From Node 0 to Node 1, Travel Time: 1
From Node 0 to Node 5, Travel Time: 5
From Node 8 to Node 0, Travel Time: 1.8
From Node 0 to Node 8, Travel Time: 8
From Node 9 to Node 7, Travel Time: 8.9
From Node 7 to Node 0, Travel Time: 1.7
From Node 5 to Node 1, Travel Time: 2.5
From Node 1 to Node 8, Travel Time: 8.1
From Node 1 to Node 0, Travel Time: 1.1
From Node 3 to Node 0, Travel Time: 1.3
From Node 2 to Node 4, Travel Time: 4.2
From Node 3 to Node 0, Travel Time: 1.3
From Node 0 to Node 3, Travel Time: 3
From Node 9 to Node 7, Travel Time: 8.9
From Node 9 to Node 0, Travel Time: 1.9
From Node 7 to Node 9, Travel Time: 9.7
From Node 7 to Node 0, Travel Time: 1.7
From Node 0 to Node 9, Travel Time: 9
From Node 0 to Node 7, Travel Time: 7
Objective 1 (Number of Edges): 25
Objective 2 (Average Commute Time): 541.10
```



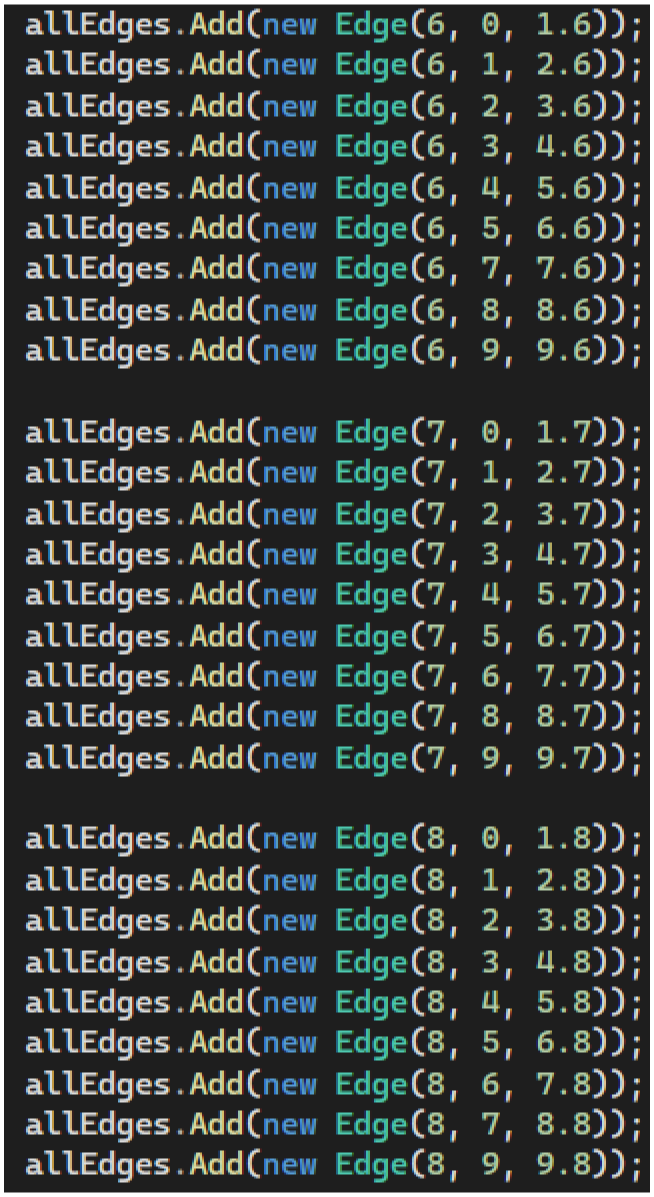
```
Solution Edges (Flight Connections):
From Node 2 to Node 0, Travel Time: 1.2
From Node 0 to Node 2, Travel Time: 2
From Node 4 to Node 0, Travel Time: 1.4
From Node 5 to Node 0, Travel Time: 1.5
From Node 0 to Node 4, Travel Time: 4
From Node 0 to Node 6, Travel Time: 6
From Node 0 to Node 1, Travel Time: 1
From Node 0 to Node 5, Travel Time: 5
From Node 8 to Node 0, Travel Time: 1.8
From Node 0 to Node 3, Travel Time: 3
From Node 7 to Node 0, Travel Time: 1.7
From Node 5 to Node 1, Travel Time: 2.5
From Node 1 to Node 8, Travel Time: 8.1
From Node 1 to Node 7, Travel Time: 7.1
From Node 1 to Node 0, Travel Time: 1.1
From Node 3 to Node 0, Travel Time: 1.3
From Node 5 to Node 6, Travel Time: 6.5
From Node 0 to Node 7, Travel Time: 7
From Node 2 to Node 8, Travel Time: 8.2
From Node 0 to Node 9, Travel Time: 9
From Node 0 to Node 9, Travel Time: 9
From Node 0 to Node 6, Travel Time: 6
From Node 9 to Node 0, Travel Time: 1.9
From Node 9 to Node 6, Travel Time: 7.9
From Node 6 to Node 0, Travel Time: 1.6
From Node 6 to Node 9, Travel Time: 9.6
Objective 1 (Number of Edges): 26
Objective 2 (Average Commute Time): 529.20
```



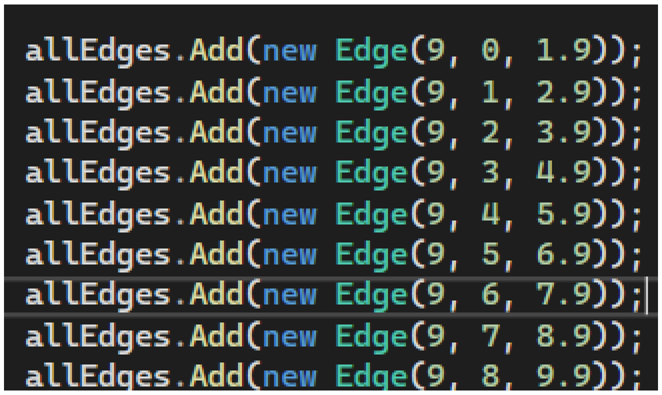
```
numberOfNodes = 10;
allEdges.Add(new Edge(0, 1, 1.0));
allEdges.Add(new Edge(0, 2, 2.0));
allEdges.Add(new Edge(0, 3, 3.0));
allEdges.Add(new Edge(0, 4, 4.0));
allEdges.Add(new Edge(0, 5, 5.0));
allEdges.Add(new Edge(0, 6, 6.0));
allEdges.Add(new Edge(0, 7, 7.0));
allEdges.Add(new Edge(0, 8, 8.0));
allEdges.Add(new Edge(0, 9, 9.0));

allEdges.Add(new Edge(1, 0, 1.1));
allEdges.Add(new Edge(1, 2, 2.1));
allEdges.Add(new Edge(1, 3, 3.1));
allEdges.Add(new Edge(1, 4, 4.1));
allEdges.Add(new Edge(1, 5, 5.1));
allEdges.Add(new Edge(1, 6, 6.1));
allEdges.Add(new Edge(1, 7, 7.1));
allEdges.Add(new Edge(1, 8, 8.1));
allEdges.Add(new Edge(1, 9, 9.1));

allEdges.Add(new Edge(2, 0, 1.2));
allEdges.Add(new Edge(2, 1, 2.2));
allEdges.Add(new Edge(2, 3, 3.2));
allEdges.Add(new Edge(2, 4, 4.2));
allEdges.Add(new Edge(2, 5, 5.2));
allEdges.Add(new Edge(2, 6, 6.2));
allEdges.Add(new Edge(2, 7, 7.2));
allEdges.Add(new Edge(2, 8, 8.2));
allEdges.Add(new Edge(2, 9, 9.2));
```



```
allEdges.Add(new Edge(3, 0, 1.3));
allEdges.Add(new Edge(3, 1, 2.3));
allEdges.Add(new Edge(3, 2, 3.3));
allEdges.Add(new Edge(3, 4, 4.3));
allEdges.Add(new Edge(3, 5, 5.3));
allEdges.Add(new Edge(3, 6, 6.3));
allEdges.Add(new Edge(3, 7, 7.3));
allEdges.Add(new Edge(3, 8, 8.3));
allEdges.Add(new Edge(3, 9, 9.3));

allEdges.Add(new Edge(4, 0, 1.4));
allEdges.Add(new Edge(4, 1, 2.4));
allEdges.Add(new Edge(4, 2, 3.4));
allEdges.Add(new Edge(4, 3, 4.4));
allEdges.Add(new Edge(4, 5, 5.4));
allEdges.Add(new Edge(4, 6, 6.4));
allEdges.Add(new Edge(4, 7, 7.4));
allEdges.Add(new Edge(4, 8, 8.4));
allEdges.Add(new Edge(4, 9, 9.4));

allEdges.Add(new Edge(5, 0, 1.5));
allEdges.Add(new Edge(5, 1, 2.5));
allEdges.Add(new Edge(5, 2, 3.5));
allEdges.Add(new Edge(5, 3, 4.5));
allEdges.Add(new Edge(5, 4, 5.5));
allEdges.Add(new Edge(5, 6, 6.5));
allEdges.Add(new Edge(5, 7, 7.5));
allEdges.Add(new Edge(5, 8, 8.5));
allEdges.Add(new Edge(5, 9, 9.5));
```



```
allEdges.Add(new Edge(6, 0, 1.6));
allEdges.Add(new Edge(6, 1, 2.6));
allEdges.Add(new Edge(6, 2, 3.6));
allEdges.Add(new Edge(6, 3, 4.6));
allEdges.Add(new Edge(6, 4, 5.6));
allEdges.Add(new Edge(6, 5, 6.6));
allEdges.Add(new Edge(6, 7, 7.6));
allEdges.Add(new Edge(6, 8, 8.6));
allEdges.Add(new Edge(6, 9, 9.6));

allEdges.Add(new Edge(7, 0, 1.7));
allEdges.Add(new Edge(7, 1, 2.7));
allEdges.Add(new Edge(7, 2, 3.7));
allEdges.Add(new Edge(7, 3, 4.7));
allEdges.Add(new Edge(7, 4, 5.7));
allEdges.Add(new Edge(7, 5, 6.7));
allEdges.Add(new Edge(7, 6, 7.7));
allEdges.Add(new Edge(7, 8, 8.7));
allEdges.Add(new Edge(7, 9, 9.7));

allEdges.Add(new Edge(8, 0, 1.8));
allEdges.Add(new Edge(8, 1, 2.8));
allEdges.Add(new Edge(8, 2, 3.8));
allEdges.Add(new Edge(8, 3, 4.8));
allEdges.Add(new Edge(8, 4, 5.8));
allEdges.Add(new Edge(8, 5, 6.8));
allEdges.Add(new Edge(8, 6, 7.8));
allEdges.Add(new Edge(8, 7, 8.8));
allEdges.Add(new Edge(8, 9, 9.8));
```



```
allEdges.Add(new Edge(9, 0, 1.9));
allEdges.Add(new Edge(9, 1, 2.9));
allEdges.Add(new Edge(9, 2, 3.9));
allEdges.Add(new Edge(9, 3, 4.9));
allEdges.Add(new Edge(9, 4, 5.9));
allEdges.Add(new Edge(9, 5, 6.9));
allEdges.Add(new Edge(9, 6, 7.9));
allEdges.Add(new Edge(9, 7, 8.9));
allEdges.Add(new Edge(9, 8, 9.9));
```

6. Concluzii

Algoritmul NSGA-II a reușit să găsească un set diversificat de soluții eficiente pentru problema optimizării grafului. Echilibrul dintre costuri și performanță este clar evidențiat de soluțiile din frontul Pareto. Utilizarea componentelor puternic conexe și algoritmul lui Dijkstra au contribuit la validarea și evaluarea soluțiilor.

7. Bibliografie

1. K. Deb et al., "A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II."
2. T.H. Cormen et al., "Introduction to Algorithms."
3. Documentația .NET pentru clasele List și SortedSet.

8. Contribuțiile membrilor echipei

- **Ciobanu Ciprian-Ionut :** Dezvoltarea modulelor legate de graf (graph.cs) + Crearea documentației și analizarea rezultatelor.
- **Burbulea Adrian:** Implementarea algoritmului NSGA-II (program.cs) + Testarea funcționalității și evaluarea complexității + Documentație.