Wektorowanie statków powietrznych przy zastosowaniu metody programowania matematycznego

Autor: **Mateusz Oleszek, nr. 144608**

Kod

Solvera ILOG CPLEX użyłem przy pomocy udostępnionego API do języka programowania C#

Import danych

```
static int[][] ImportData(int planes, int maneuvers)
        {
            var textFile = $"CM/CM_n={planes}_m={maneuvers}.txt";
            var matrix = File.ReadAllLines(textFile)
                .Select(line => line.Split(' ')
                    .Select(letter => int.Parse(letter)).ToArray()
                    ).ToArray();

            return matrix;
        }
```

Tworzenie równań

```
static (INumVar[], IRange[]) PopulateByRow(IMPModeler model, int planes, int maneuvers)
        {
            var variableNames = new List<string>(maneuvers * planes);
            for (int i = 0; i < planes; i++)
            {
                for (int j = 0; j < maneuvers; j++)
                {
                    variableNames.Add($"samolot_{i}_manewr_{j}");
                }
            }

            var variables = model.NumVarArray(planes * maneuvers, 0, 1, NumVarType.Int, variableNames.ToArray());

            var singlePlaneConstraints = new List<INumExpr>();
            for(int plane = 0; plane < planes; plane++)
            {
                var plane_sum = new List<INumExpr>();
                for(int maneuver = 0; maneuver < maneuvers; maneuver++)
                {
                    var idx = plane * maneuvers + maneuver;
                    plane_sum.Add(variables[idx]);
                }
                singlePlaneConstraints.Add(model.Sum(plane_sum.ToArray()));
            }

            var matrix = ImportData(planes, maneuvers);

            var collisionConstraints = new List<INumExpr>();
            for(int plane1 = 0; plane1 < planes; plane1++)
            {
                for(int manuver1 = 0; manuver1 < maneuvers; manuver1++)
                {
                    for (int plane2 = plane1 + 1; plane2 < planes; plane2++)
                    {
                        for (int manuver2 = 0; manuver2 < maneuvers; manuver2++)
```

```
                        {
                            var plane1idx = plane1 * maneuvers + manuver1;
                            var plane2idx = plane2 * maneuvers + manuver2;

                            if (matrix[plane1idx][plane2idx] == 1)
                            {
                                collisionConstraints.Add(model.Sum(variables[plane1idx], variables[plane2idx]));
                            }
                        }
                    }
                }
            }

            var rng = new IRange[singlePlaneConstraints.Count + collisionConstraints.Count];
            for(int i = 0; i < singlePlaneConstraints.Count; i++)
            {
                rng[i] = model.AddEq(singlePlaneConstraints[i], 1, $"sp_{i}");
            }

            for (int i = 0; i < collisionConstraints.Count; i++)
            {
                rng[i] = model.AddLe(collisionConstraints[i], 1, $"col_{i}");
            }

            model.AddMinimize();
            return (variables, rng);
        }
```

Rozwiązywanie równań

```
static void SolvePlanes(int planes, int maneuvers)
        {
            Console.WriteLine($"\nSolving MIP for n={planes} and m={maneuvers}");
            Cplex cplex = new Cplex();

            var (var, rng) = PopulateByRow(cplex, planes, maneuvers);

            cplex.ExportModel($"lpex_{planes}_{maneuvers}.lp");

            if (cplex.Solve())
            {
                double[] x = cplex.GetValues(var);
                cplex.Output().WriteLine("Solution status = " + cplex.GetStatus());
                Console.WriteLine("\nRows - planes.\nColumns - maneuvers");

                for (int plane = 0; plane < planes; plane++)
                {
                    Console.Write(plane + ": ");
                    for (int maneuver = 0; maneuver < maneuvers; maneuver++)
                    {
                        var idx = plane * maneuvers + maneuver;
                        Console.Write(x[idx] + " ");
                    }
                    Console.WriteLine();
                }
            }
            cplex.End();
        }
```

Główna funkcja

```
public static void Main(string[] args)
        {
            var planes = new int[]{ 10, 20, 30, 40};
            var maneuvers = 7;

            foreach (var plane in planes)
            {
                SolvePlanes(plane, maneuvers);
            }
        }
```

Czasy i Wyniki

Wektor wynikowy jest sformatowany tak, że każdy rząd pokazuje wybrany manewr dla jednego samolotu.

N = 10

```
Found incumbent of value 0.000000 after 0.00 sec. (0.01 ticks)

Root node processing (before b&c):
  Real time             =    0.00 sec. (0.01 ticks)
Parallel b&c, 12 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.00 sec. (0.01 ticks)
Solution status = Optimal

0: 1 0 0 0 0 0 0
1: 1 0 0 0 0 0 0
2: 1 0 0 0 0 0 0
3: 0 0 1 0 0 0 0
4: 0 1 0 0 0 0 0
5: 1 0 0 0 0 0 0
6: 1 0 0 0 0 0 0
7: 0 1 0 0 0 0 0
8: 0 0 1 0 0 0 0
9: 1 0 0 0 0 0 0
```

N = 20

```
Presolve time = 0.00 sec. (0.93 ticks)
Found incumbent of value 0.000000 after 0.00 sec. (1.88 ticks)

Root node processing (before b&c):
  Real time             =    0.00 sec. (1.89 ticks)
Parallel b&c, 12 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.00 sec. (1.89 ticks)

0: 0 0 0 0 0 1 0
1: 0 0 0 1 0 0 0
2: 0 0 0 0 1 0 0
3: 0 0 0 1 0 0 0
```

```
4: 0 0 1 0 0 0 0
5: 1 0 0 0 0 0 0
6: 1 0 0 0 0 0 0
7: 1 0 0 0 0 0 0
8: 1 0 0 0 0 0 0
9: 0 1 0 0 0 0 0
10: 0 0 0 0 0 0 1
11: 0 0 0 0 1 0 0
12: 0 0 0 0 1 0 0
13: 0 0 0 0 1 0 0
14: 0 0 1 0 0 0 0
15: 0 0 0 1 0 0 0
16: 0 0 0 0 1 0 0
17: 0 0 0 0 0 1 0
18: 0 0 0 1 0 0 0
19: 0 0 0 0 1 0 0
```

N = 30

Nie mogłem obliczyć rozwiązań dla macierzy dla 30 i 40 samolotów, ponieważ liczba ograniczeń wynikających z tablicy CM przekracza 1000, co jest odgórnie narzuconym limitem darmowej wersji. Jest tak nawet z uwzględnieniem tylko jednej połowy tablicy.