Wektorowanie statków powietrznych przy zastosowaniu metody programowania matematycznego, cz.2

Optymalizacja

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

Kod

Solvera ILOG CPLEX użyłem przy pomocy udostępnionego API do języka programowania C#.
Kod jest identyczny jak w części pierwszej, oprócz funkcji dodającej równanie optymalizacyjne.

Zamiast pustej funkcji optymalizacyjnej

```
model.AddMinimize();
```

Jest wywoływane

```
static void AddOptimization(IMPModeler model, int planes, int maneuvers, INumVar[] variables)
{
    var coefficients = new List<int>();
    for (int plane = 0; plane1 < planes; plane++)
    {
        for (int manuver = 0; manuver < maneuvers; manuver++) {
            coefficients.Add(manuver + 1);
        }
    }

    var minimization = model.ScalProd(variables, coefficients.ToArray());
    model.AddMinimize(minimization);
}
```

Czasy i Wyniki

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

N = 10

```
Total (root+branch&cut) =    0.00 sec. (0.74 ticks)
Solution status = Optimal
Solution value  = 14

Rows - planes.
Columns - maneuvers
0: 0 0 0 1 0 0 0
1: 0 1 0 0 0 0 0
2: 1 0 0 0 0 0 0
3: 1 0 0 0 0 0 0
4: 1 0 0 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: 1 0 0 0 0 0 0
```

N = 20

```
Total (root+branch&cut) =    0.01 sec. (3.94 ticks)
Solution status = Optimal
Solution value  = 28

Rows - planes.
Columns - maneuvers
0: 1 0 0 0 0 0 0
1: 0 1 0 0 0 0 0
2: 1 0 0 0 0 0 0
3: 1 0 0 0 0 0 0
4: 1 0 0 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: 1 0 0 0 0 0 0
10: 0 1 0 0 0 0 0
11: 0 1 0 0 0 0 0
12: 1 0 0 0 0 0 0
13: 1 0 0 0 0 0 0
14: 1 0 0 0 0 0 0
15: 1 0 0 0 0 0 0
16: 1 0 0 0 0 0 0
17: 0 0 1 0 0 0 0
18: 0 1 0 0 0 0 0
19: 0 0 1 0 0 0 0
```

N = 30

```
Total (root+branch&cut) =    0.25 sec. (100.67 ticks)
Solution status = Optimal
Solution value  = 66

Rows - planes.
Columns - maneuvers
0: 0 0 0 0 1 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: 1 0 0 0 0 0 0
5: 0 0 0 1 0 0 0
6: 0 0 0 1 0 0 0
7: 0 0 0 1 0 0 0
8: 0 1 0 0 0 0 0
9: 0 0 1 0 0 0 0
10: 1 0 0 0 0 0 0
11: 0 0 1 0 0 0 0
12: 1 0 0 0 0 0 0
13: 1 0 0 0 0 0 0
14: 0 0 1 0 0 0 0
15: 1 0 0 0 0 0 0
16: 0 1 0 0 0 0 0
17: 1 0 0 0 0 0 0
18: 0 0 1 0 0 0 0
19: 0 1 0 0 0 0 0
20: 1 0 0 0 0 0 0
21: 0 1 0 0 0 0 0
22: 1 0 0 0 0 0 0
```

```
23: 0 0 1 0 0 0 0
24: 0 0 0 0 1 0 0
25: 0 0 0 1 0 0 0
26: 1 0 0 0 0 0 0
27: 1 0 0 0 0 0 0
28: 1 0 0 0 0 0 0
29: 1 0 0 0 0 0 0
```

N = 40

```
Total (root+branch&cut) =    0.61 sec. (102.58 ticks)
Solution status = Optimal
Solution value  = 91

Rows - planes.
Columns - maneuvers
0: 0 1 0 0 0 0 0
1: 1 0 0 0 0 0 0
2: 0 1 0 0 0 0 0
3: 1 0 0 0 0 0 0
4: 1 0 0 0 0 0 0
5: 1 0 0 0 0 0 0
6: 0 0 0 0 0 1 0
7: 0 0 0 1 0 0 0
8: 0 1 0 0 0 0 0
9: 1 0 0 0 0 0 0
10: 1 0 0 0 0 0 0
11: 0 1 0 0 0 0 0
12: 1 0 0 0 0 0 0
13: 0 1 0 0 0 0 0
14: 1 0 0 0 0 0 0
15: 0 0 0 1 0 0 0
16: 0 0 0 1 0 0 0
17: 1 0 0 0 0 0 0
18: 0 1 0 0 0 0 0
19: 1 0 0 0 0 0 0
20: 0 0 1 0 0 0 0
21: 0 1 0 0 0 0 0
22: 1 0 0 0 0 0 0
23: 0 0 0 0 0 1 0
24: 0 0 1 0 0 0 0
25: 1 0 0 0 0 0 0
26: 1 0 0 0 0 0 0
27: 1 0 0 0 0 0 0
28: 0 1 0 0 0 0 0
29: 0 0 0 0 0 1 0
30: 0 1 0 0 0 0 0
31: 0 1 0 0 0 0 0
32: 0 1 0 0 0 0 0
33: 0 1 0 0 0 0 0
34: 1 0 0 0 0 0 0
35: 0 0 1 0 0 0 0
36: 0 1 0 0 0 0 0
37: 0 0 0 0 1 0 0
38: 1 0 0 0 0 0 0
39: 0 0 0 0 1 0 0
```