Assignment 4. Linear programming.

Student name: Evgenii Dudkin
e-mail: e.dudkin@stud.uis.no

A C# project called «LinearProgramming» was developed to address the tasks outlined in the assignment. This project utilizes the Google.OrTools NuGet package as a linear programming tool. Google OR-Tools is a collection of optimization algorithms and tools developed by Google, offering a range of functionalities for solving various optimization problems [2]. Detailed information about each problem solver, along with code samples and components, will be presented in the following sections of the report.

For access to the implementations, code, comments, and other related materials, please refer to the GitHub repository [1].

1 Exam problem

The problem would be formulated as a Linear Programming problem, as follows:

Decision Variables:

- Let x be the quantity of product X produced per week.
- Let y be the quantity of product Y produced per week.

Objective Function:

Maximize the total profit, which is the revenue minus the cost. Given that the production of product X requires 15 minutes (1/4 hour) of machine time and 20 minutes (1/3 hour) of craftsman time, and the production of product Y requires 20 minutes (1/3 hour) of machine time and 30 minutes (1/2 hour) of craftsman time, taking into account the cost of the resources, we obtain:

$Maximize\ Z = (200x + 300y) - (\frac{100}{4}x + \frac{100}{3}y + \frac{20}{3}x + \frac{20}{2}y)) = \frac{505}{3}x + \frac{770}{3}y$

Constraints:

- $\frac{1}{4}x + \frac{1}{3}y \leq 40$
- $\frac{1}{3}x + \frac{1}{2}y \leq 35$
- $x \geq 10$
- $y \geq 0$

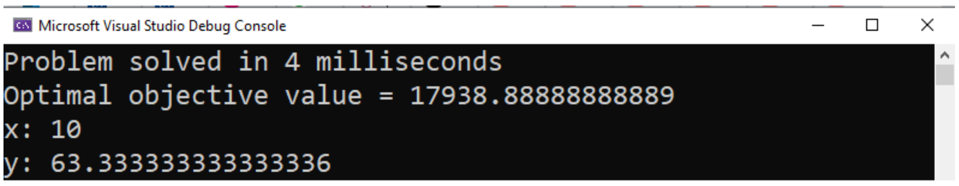
Putting it all together, the complete linear programming model is: $Maximize\ Z = \frac{505}{3}x + \frac{770}{3}y$ subject to

$$\begin{cases} \frac{1}{4}x + \frac{1}{3}y \leq 40 \\ \frac{1}{3}x + \frac{1}{2}y \leq 35 \\ x \geq 10 \\ y \geq 0 \end{cases}$$

LP problem was solved programmatically (Listing 1). Figure 1 illustrates the output of the program. Thus, 10 products of X and 63 products of Y should be produced per week to maximize the total profit.

```
Solver solver = Solver.CreateSolver( "GLOP" );
Variable x = solver.MakeNumVar( 0.0, double.PositiveInfinity, "x" );
Variable y = solver.MakeNumVar( 0.0, double.PositiveInfinity, "y" );

// Maximize 505/3 x + 770 / 3 y
Objective objective = solver.Objective();
objective.SetCoefficient( x, 505.0 / 3.0 );
objective.SetCoefficient( y, 770.0 / 3.0 );
objective.SetMaximization();

// x >= 10
Constraint c0 = solver.MakeConstraint( 10, double.PositiveInfinity );
c0.SetCoefficient( x, 1 );

// y >= 0
Constraint c1 = solver.MakeConstraint( 0.0, double.PositiveInfinity );
c1.SetCoefficient( y, 1 );

//1/4 x + 1/3 y <= 40
Constraint c2 = solver.MakeConstraint( 0, 40 );
c2.SetCoefficient( x, 1.0 / 4.0 );
c2.SetCoefficient( y, 1.0 / 3.0 );

//1/3 x + 1/2 y <= 35
Constraint c3 = solver.MakeConstraint( 0, 35 );
c3.SetCoefficient( x, 1.0 / 3.0 );
c3.SetCoefficient( y, 1.0 / 2.0 );

SolveProblem( solver );
```

Listing 1: LP solution



```
Problem solved in 4 milliseconds
Optimal objective value = 17938.88888888889
x: 10
y: 63.333333333333336
```

Figure 1: Exam problem's solution

2 Maximum Flow

2.1 Bottleneck network cut

The problem would be formulated as an optimization problem, as follows:
Decision Variables:

- Let d_{uv} $\forall(u,v) \in E$ be the variables that indicate whether the edges belong to the min-cut:

$$d_{uv} = \begin{cases} 1, \text{ if u } \in \text{ S and v } \in \text{ T} \\ 0, \text{otherwise} \end{cases}$$

- Let z_v $\forall v \in V \setminus \{s,t\}$ be the variables that indicate whether the node in S-set:

$$z_v = \begin{cases} 1, \text{ if v } \in \text{ S} \\ 0, \text{otherwise} \end{cases}$$

Objective Function:
Since the objective is to identify the bottleneck network cut, it is necessary to minimize the total capacity of edges in the cut:
$Minimize\ Z = \sum_{(u,v)\in E} c_{uv} d_{uv}$
Constraints:

- $d_{uv} - z_u + z_v \geq 0$ $\forall(u,v) \in E, u \neq s, v \neq t$
  $d_{uv} \geq z_u + z_v$ guarantees that, for non-terminal nodes u,v, if u is in S and v is in T, then the edge (u,v) is counted in the cut ($d_{uv} \geq 1$).

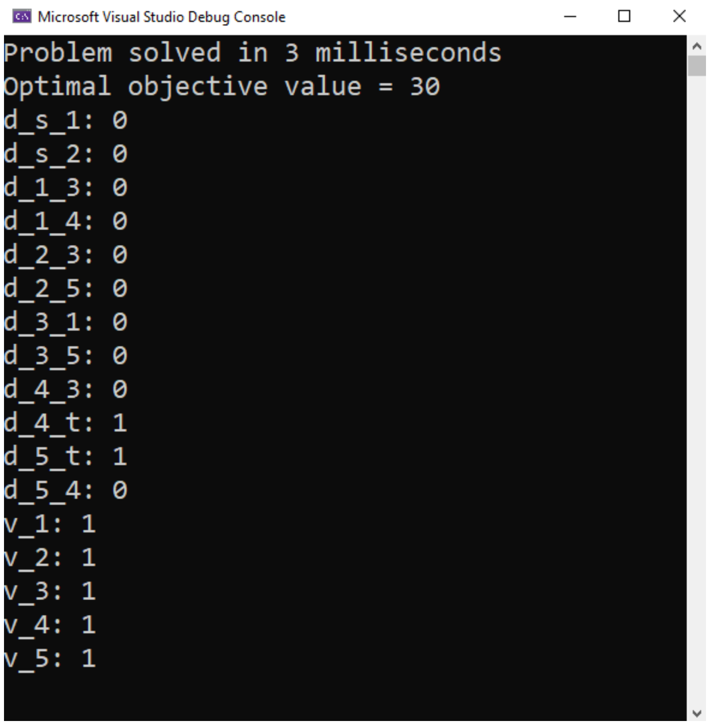
- $d_{sv} + z_v \geq 1$ $\forall(s,v) \in E, v \neq t$
  $d_{sv} \geq 1 - z_v$ guarantees that, if v is in T, then the edge (s,v) is counted in the cut (since s is by definition in S)

- $d_{ut} - z_u \geq 0 \; \forall (u,t) \; inE, u \neq s$
  $d_{ut} \geq z_u$ guarantees that, if u is in S, then the edge (u,t) is counted in the cut (since t is by definition in T)
- $d_{st} \geq 1$ if $(s,t) \in E$
- $d_{uv} \geq 0 \; \forall (u,v) \in E$
- $z_v \geq 0 \; \forall v \in V \setminus \{s,t\}$

Putting it all together, the complete linear programming model is: $Minimize \; Z = \sum_{(u,v)\in E} c_{uv} d_{uv}$ subject to

$$\begin{cases} d_{uv} - z_u + z_v \geq 0 \; \forall (u,v) \in E, u \neq s, v \neq t \\ d_{sv} + z_v \geq 1 \; \forall (s,v) \in E, v \neq t \\ d_{ut} - z_u \geq 0 \; \forall (u,t) \; inE, u \neq s \\ d_{st} \geq 1 \text{ if } (s,t) \in E \\ d_{uv} \geq 0 \; \forall (u,v) \in E \\ z_v \geq 0 \; \forall v \in V \setminus \{s,t\} \end{cases}$$

This optimization problem was solved programmatically. Figure 2 illustrates the output of the program. Thus, the bottleneck of the flow network is 30 and the cut is $S = \{s, V_1, V_2, V_3, V_4, V_5\}$, $T = \{t\}$



```
Problem solved in 3 milliseconds
Optimal objective value = 30
d_s_1: 0
d_s_2: 0
d_1_3: 0
d_1_4: 0
d_2_3: 0
d_2_5: 0
d_3_1: 0
d_3_5: 0
d_4_3: 0
d_4_t: 1
d_5_t: 1
d_5_4: 0
v_1: 1
v_2: 1
v_3: 1
v_4: 1
v_5: 1
```

Figure 2: Bottleneck network cut problem's solution

2.2 Maximum flow

The problem would be formulated as an optimization problem, as follows:
Decision Variables:

- Let x_{ij} $\forall (i,j) \in E$ be the flow from i to j in the max flow netwrok.

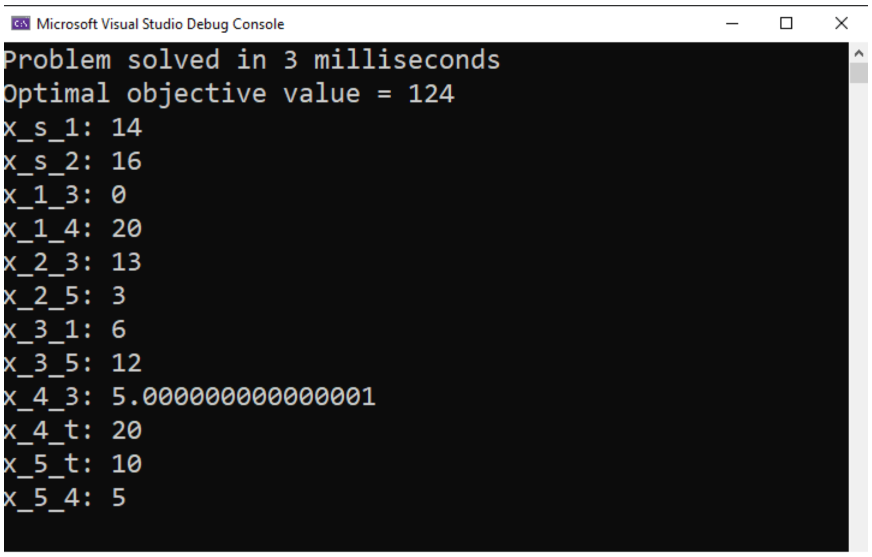
Objective Function:
$Maximize\ Z = \sum_{(i,j)\in E} x_{ij}$
Constraints:

- $0 \leq x_{ij} \leq c_{ij}$ $\forall (i,j) \in E$ (capacity constraint)
- $\sum_u x_{uv} - \sum_w x_{vw} = 0$ $\forall v \in V \setminus \{s,t\}$ (flow conservation)

Putting it all together, the complete linear programming model is: $Maximize\ Z = \sum_{(i,j)\in E} x_{ij}$ subject to

$$\begin{cases} 0 \leq x_{ij} \leq c_{ij}\ \forall (i,j) \in E \\ \sum_u x_{uv} - \sum_w x_{vw} = 0\ \forall v \in V \setminus \{s,t\} \end{cases}$$

This optimization problem was solved programmatically. Figure 3 illustrates the output of the program, representing the maximum flow at the edges of the network.



```
Problem solved in 3 milliseconds
Optimal objective value = 124
x_s_1: 14
x_s_2: 16
x_1_3: 0
x_1_4: 20
x_2_3: 13
x_2_5: 3
x_3_1: 6
x_3_5: 12
x_4_3: 5.000000000000001
x_4_t: 20
x_5_t: 10
x_5_4: 5
```

Figure 3: Maximum flow problem's solution

References

[1] Github Evgenii Dudkin. — URL: `https://github.com/EvgeneDudkin/dat600` (online; accessed: 07.04.2024).

[2] Tools Google Optimization. — URL: `https://github.com/google/or-tools` (online; accessed: 07.04.2024).