Solving Economic Problems by Using OML Modeling Language

Radim Remeš

Abstract: The aim of this article is to introduce an algebraic modeling language called Optimization Modeling Language (OML) to solve economic optimization problems. How to use this language is demonstrated on the example using the methods of mixed linear programming in OML language. The solution is found by using the Microsoft Solver Foundation tool, which can be used for mathematical simulation, optimization, and modeling. The example solution is demonstrated by using the Microsoft Visual Studio programming tool.

The benefits of using this tool for mathematics include easy usage and concurrently wide scope of the problems solution.

Key words: Business Analytics · Solver Foundation Services · Optimization Modeling Language · OML · Linear programming · Mixed integer linear programming

JEL Classification: C61 · C63

1 Introduction

This article introduces an algebraic modeling language called Optimization Modeling Language (OML). Optimizing methods are used to solve a whole spectrum of problems.

Optimizing methods are mainly used for covering conventional optimization (Gen, Cheng, 2000), simulation and risk analysis (Goswami, Mishra, 2016), decision analysis (Palvia, Gordon, 1992), simulation optimization (Carson, Maria, 1997), stochastic optimization (Houda, Klicnarova, 2015), robust optimization, or food processing industry (Rieger et al., 1989).

Nowadays there are many software for solving optimization problems, frequently used are XPRESS-MP (Fico, 2016), ILOG CPLEX (IBM, 2016, LINGO (Lindo, 2016), lp_solve (Eikland, Notebaert, 2016), or Gurobi (Gurobi Optimization, 2016).

How to use OML language is demonstrated on the example by using the methods of linear programming. The solution is found by using the Microsoft Solver Foundation tool (Microsoft, 2013), which can be used for mathematical simulation, optimization, and modeling. The example solution is demonstrated by using the Microsoft Visual Studio programming tool (Microsoft, 2016). These tools were also used by students in the economic course to calculate the optimization of e-commerce (Beranek, Remes, 2013).

2 Optimal solution finding

To demonstrate this, we used an example for which the inspiration from books of authors Asllani (2015) and Kreyszig, Kreyszig, Norminton (2011) was obtained.

A small company produces scooters for children. The sales forecast in units for the coming year are given in the table 1. The company has a capacity of 300 scooters per month. It is possible to augment the production by up to 25% through overtime working, but this increases the production cost for a scooter from the usual 400 CZK to 500 CZK. Currently there are 50 scooters in stock. The storage costs have been calculated as 100 CZK per unit held in stock at the end of a month. We assume that the storage capacity at the company is unlimited (the real capacity does not impose any limits in our case). We are at the first of January. Which quantities need to be produced and stored in the course of the next twelve months in order to satisfy the forecast demand and minimize the total cost?
Table 1 Sales forecasts for the coming year

<table>
<thead>
<tr>
<th>Month</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forecast items</td>
<td>300</td>
<td>150</td>
<td>250</td>
<td>250</td>
<td>600</td>
<td>700</td>
<td>420</td>
<td>500</td>
<td>290</td>
<td>120</td>
<td>200</td>
<td>300</td>
</tr>
</tbody>
</table>

Source: Own processing

First, we need to define decision variables:

- $s_i$: quantity stored in $i$-th month,
- $p_i$: quantity produced in $i$-th month,
- $p'_i$: quantity produced in overtime work in $i$-th month.

Next we define parameters:

- $d_i$: demand forecast in $i$-th month (see table 1).

Now we can formulate the objective function:

$$\min C = \sum_{i=1}^{12} (s_i \cdot 100 + p_i \cdot 400 + p'_i \cdot 500)$$

where:

- $C$: the total production cost,
- $s_i$: quantity stored in $i$-th month,
- $p_i$: quantity produced in $i$-th month,
- $p'_i$: quantity produced in overtime work in $i$-th month,
- $d_i$: demand forecast in $i$-th month.

For objective function we have to define the set of constraint:

$$s_i \geq 0 \text{ and } p_i \geq 0 \text{ and } p'_i \geq 0$$

$$p_i \leq 300 \text{ and } p'_i \leq 375$$

$$p_i + p'_i + 50 = d_i + s_i$$

$$p_i + p'_i + s_{i-1} = d_i + s_i$$

And the set of non-negativity constraints:

$$s_i \geq 0 \text{ and } p_i \geq 0 \text{ and } p'_i \geq 0 \text{ for all } i = (1, 12)$$

Finally, overall formulation:

$$\min C = \sum_{i=1}^{12} (s_i \cdot 100 + p_i \cdot 400 + p'_i \cdot 500)$$

subject to:

- $p_i \leq 300 \text{ and } p'_i \leq 375$ for all $i = (1, 12)$ (constraint maximum amount of scooters)
- $p_i + p'_i + 50 = d_i + s_i$ for $i = 1$ (constraint for January)
- $p_i + p'_i + s_{i-1} = d_i + s_i$ for all $i = (2, 12)$ (constraint for other months than January)
- $s_i \geq 0$ and $p_i \geq 0$ and $p'_i \geq 0$ for all $i = (1, 12)$ (non-negativity constraint)
3 Results - Solving the problem in Microsoft Visual Studio

For solution of our problem we utilize the library for mathematic programming and optimization Microsoft Solver Foundation (Microsoft, 2013). We use this library alongside with Microsoft Visual Studio development environment (Microsoft, 2016).

Both Microsoft Visual Studio development environment Community edition and Microsoft Solver Foundation libraries Express edition are distributed free, it is possible to download them from Microsoft website.

3.1 Developing Models

Microsoft Solver Foundation offers three developing models in Microsoft Visual Studio environment, which leads to these possible ways to the final solution:

- Solver Foundation Services,
- Solver Foundation Solvers,
- Optimization Modeling Language.

Developing model Solver Foundation Services (SFS) is suitable for novice developers, because desired algorithm (solver) is chosen automatically.

Solver Foundation Solvers developing model accesses to specific solvers (e.g. solver for linear programming, quadratic programming, integer programming solver for quasi-Newton methods) and is more appropriate for advanced developers.

Optimization Modeling Language (OML) is an algebraic modeling language that is designed exclusively for modeling and problem solving. Source code and designed model are mutually separate, each of them is in a separated file. This leads to the transparency of the program and the designed model can thus be easily edited later and modified for resolving the given problem. We will use this developer model in our problem solving.

3.2 Optimization Modeling Language

The example solution of our chosen task will be presented on a model designing by using Optimization Modeling Language (OML). The model will be used together with developing model Solver Foundation Services.

The model design is prepared with the use of equations 1-7 consisting of these stages:

- set of parameters (months demand forecast),
- decision variables (productions),
- constraint,
- goals.

On listing 1 is shown frame of a whole programme in C# language, the commented lines will be further completed with the code of solution.

Listing 1 Frame of a programme for SFS model

```csharp
using System;
using Microsoft.SolverFoundation.Services;

namespace ScooterPlanning
{
    class ScooterProduction
    {
        public void Solve()
        {
            // actual solution
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            ScooterProduction sProd = new ScooterProduction();
            sProd.Solve(); // call solution
        }
    }
}
```
Creating the object of the designed model is implemented by loading it into solvers SolverContext class instance context. Custom model is loaded from an external file scooter.oml, the source code is shown in listing 2.

Listing 2 Object Model Loading

```csharp
SolverContext context = SolverContext.GetContext();
context.LoadModel(FileFormat.OML, "scooter.oml");
```

### 3.3 Model in OML

The whole model is described in an external file using OML. The parameters in this example represents individual months. The model decision determines that we need variables, each variable will be a whole set of non-negative integers. Constraints in the model are determined by the example assignment. Section of the model in which is described the desired target states that we are trying to get the minimum total cost. Listing 3 shows the entire model designed using the OML.

Listing 3 Model described in OML file

```xml
Model[
    Parameters[
        Sets[Integers[1, 12]],
        Months
    ],
    Parameters[
        Integers[0, Infinity],
        DemandForecast[Months]
    ],
    Decisions[
        Integers[0, Infinity],
        ProductionNormal[Months],
        ProductionOT[Months],
        store[Months]
    ],
    Constraints[
        Caps -> Foreach[
            {iter1, Months},
            ProductionNormal[iter1] <= 300
        ],
        OT_Caps -> Foreach[
            {iter2, Months},
            ProductionOT[iter2] <= 375
        ],
        SupplyDemandJan -> Foreach[
            {iter3, Months},
        ],
        SupplyDemandRest -> Foreach[
            {iter4, Months},
        ],
    ],
    Goals[
```
3.4 Founded solution

We already have everything ready for the calculation of the final solution (calling Solve method from the Model context and writing solution report, listing 4).

Listing 4 The solution calculation and report display

```csharp
Solution solution = context.Solve();
Report r = solution.GetReport();
Console.WriteLine(r.ToString());
```

After running the programme it displays the resulting message on the calculation and shows the founded optimal solution (listing 5).

Listing 5 Report with received results

```plaintext
===Solver Foundation Service Report===

Date: 1.11.2016 11:21:43
Version: Microsoft Solver Foundation 3.0.2.10889 Enterprise Edition
Model Name: DefaultModel
Capabilities Applied: MILP
Solve Time (ms): 1099
Total Time (ms): 2674
Solve Completion Status: Optimal
Solver Selected: Microsoft.SolverFoundation.Solvers.SimplexSolver
Directives:
Simplex(TimeLimit = -1, MaximumGoalCount = -1, Arithmetic = Default,
Pricing = Default, IterationLimit = -1, Algorithm = Default, Basis =
Default, GetSensitivity = False)
Algorithm: Primal
Arithmetic: Double
Variables: 36 -> 36 + 37
Rows: 37 -> 37
Nonzeros: 107
Eliminated Slack Variables: 0
Pricing (double): SteepestEdge
Basis: Slack
Pivot Count: 16
Phase 1 Pivots: 16 + 0
Phase 2 Pivots: 0 + 0
Factorings: 4 + 0
Degenerate Pivots: 0 (0,00 %)
Branches: 0

===Solution Details===

Goals:
Cost: 1716500
Decisions:
ProductionNormal(1): 250
ProductionNormal(2): 150
ProductionNormal(3): 250
ProductionNormal(4): 250
```
Table 2 Production decisions for the coming year

<table>
<thead>
<tr>
<th>Month</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales forecast</td>
<td>300</td>
<td>150</td>
<td>250</td>
<td>250</td>
<td>600</td>
<td>700</td>
<td>420</td>
<td>500</td>
<td>290</td>
<td>120</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>Normal production</td>
<td>250</td>
<td>150</td>
<td>250</td>
<td>250</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>290</td>
<td>120</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>Overtime production</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>325</td>
<td>375</td>
<td>120</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Stored items</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Source: Own processing

4 Conclusions

The model according to the specified data finds the optimal solution. Total production cost is calculated to 1,716,500 CZK. Production will work overtime only for four months (May to August) and we only need a warehouse in May. Detailed final solution is shown in table 2.

Article showed the use of programming tools in Microsoft Visual C# library with Microsoft Solver Foundation for solving linear programming problems, the solution was specified model using algebraic language OML in a separate text file. The benefit of using this software is for the non-programmers who will use the ready-made programme in advance and design only model in an external OML file.
References


