Optimisation Modelling with Excel and CMPL2

Mike Steglich
Technical University of Applied Sciences Wildau
15745 Wildau, Germany
e-mail: mike.steglich@th-wildau.de

***Abstract* – In companies and other organisations, spreadsheet programs are essential tools for preparing and supporting decisions, as they are easy to use and available in most workplaces. For complex problems, optimisation software is used. This offers a wide range of modelling capabilities but relies on external data, such as that maintained in spreadsheets. It therefore makes sense to combine spreadsheets and optimisation software. Add-ins in spreadsheet programs such as Excel solver are relatively widespread. They allow interactive work, although the method of modelling using cell ranges does not seem to be suitable for complex models. Another possibility is to use the spreadsheet interfaces of algebraic modelling languages, which are excellent for modelling complex problems. Unfortunately, as pure data interfaces, they do not allow interactive work. There are some approaches that combine modelling languages with Excel in the form of an Excel add-in, thus combining interactive work with the modelling possibilities of the modelling languages. Unfortunately, these solutions are only available for Windows and some of them seem to have been discontinued. The consideration of all the advantages and disadvantages of the available tools led to the motivation to create an easy-to-use interface between the open-source modelling language CMPL and Excel, which allows interactive work and is available for Windows and macOS. This paper describes this interface.**

Keywords – spreadsheet optimisation; algebraic modelling language; interactive decision-making process; optimisation.

I. INTRODUCTION

To solve optimisation problems, the optimisation routines must be addressed, as well as the provision and organisation of the required data. This is often done in companies or other organisations with spreadsheet programs. This is the reason why a variety of software solutions have emerged that combine spreadsheet programs and optimisation environments. These solutions can be divided into spreadsheet add-ins and data interfaces.

The best-known spreadsheet add-in is the freely available Excel solver [1] and its commercial version by Frontline [2]. Similar solutions include the solver in LibreOffice/Calc [3], the open-source solution OpenSolver [4] [5], Frontline's add-in for Google Sheets [6], the Excel add-in Evolver by Palisade [7], Lindo's Whats'sBest! [8] and XLOPTIM by Addinsoft and LocalSolver [9].

In all these approaches, after organising the data, the user has to define the objective function, the variables and the constraints in a user dialogue, as shown in Figure 1. These definitions are made in the form of cell references. After optimisation, the solution is written in the spreadsheet cells defined for the variables. Further outputs, such as reduced costs and shadow prices can be written in separate tables.

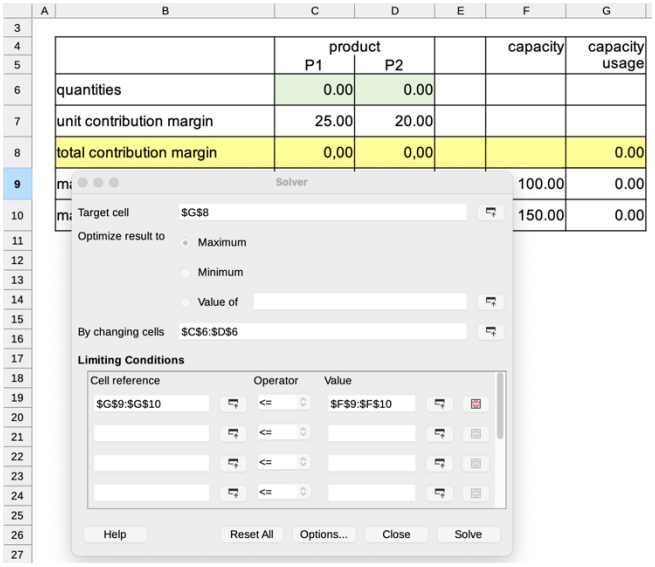


Figure 1. Solver add-in in LibreOffice.

With such solver add-ins, data and models can be easily combined and shared as needed. Since changes to the data lead to new solutions after new optimisation, interactive work is possible. On the other hand, the formulation of model relationships in the form of cell references is not suitable for complex models. Debugging models is also rather complicated [10]. Another disadvantage is that some of these add-ins are only available for Windows (e.g., What'sBest! and Evolver) and not for macOS.

Another widely used approach is algebraic modelling languages, which are more suitable for modelling and solving optimisation problems in terms of their functionality and flexibility than the solver add-ins. Most of these optimisation environments, such as AMPL [11], MPL [12], AIMMS [13], GAMS [14], OPL [15], MOSEL [16] and SAS [17] , offer an interface with which data can be read from spreadsheet files and results can be written to it. These interfaces allow the user to combine the capabilities of the languages with a widely available data source and to use the possibilities of a spreadsheet program to further process a solution that has been found. Unfortunately, these interfaces do not usually allow interactive work, as the spreadsheet files cannot be used by other processes while they are being written and thus

 ISBN: 978-1-61208-898-3

cannot be opened. Some of these software solutions (e.g., MPL, MOSEL) offer a VBA library for Microsoft Excel that allows these languages to be used within Excel [18] [19]. But such approaches are more suitable for programmers than for typical corporate decision-makers.

The combination of both an algebraic modelling language and Excel in an interactive mode seems to be a good approach to many real decision-making problems in companies and other organisations. In this context, it is worth mentioning the commercially available AIMMS Excel add-in [20] and SolverStudio [10] [21], which is available free of charge. Microsoft's Solver Foundation was an interesting offer which is evidently not being continued [22].

After choosing certain settings like the project file and licence server, a user of the AIMMS Excel add-in has to define so-called execution sequences to determine the sets and parameters to be read into the AIMMS project, the execution of the problem and the reading back of the results into the Excel spreadsheet. This facilitates interactivity in the process of formulating, solving and interpreting an optimisation problem, albeit in a rather complex way. SolverStudio offers a simpler approach. This Excel add-in allows several algebraic modelling languages (PuLP, Pyomo, AMPL, GMPL, GAMS, CMPL and Gurobi via its Python modelling interface) to be used within Excel [21]. The first step is to select one of the modelling languages and formulate the optimisation model. Then, as shown in Figure 2, the Data Item Editor is used to define the sets and parameters that are to be read into the optimisation model and the solution elements to be written into the Excel spreadsheet after the optimisation is completed. The optimisation is started either by clicking the smiley in the toolbar or via the language menu [10].

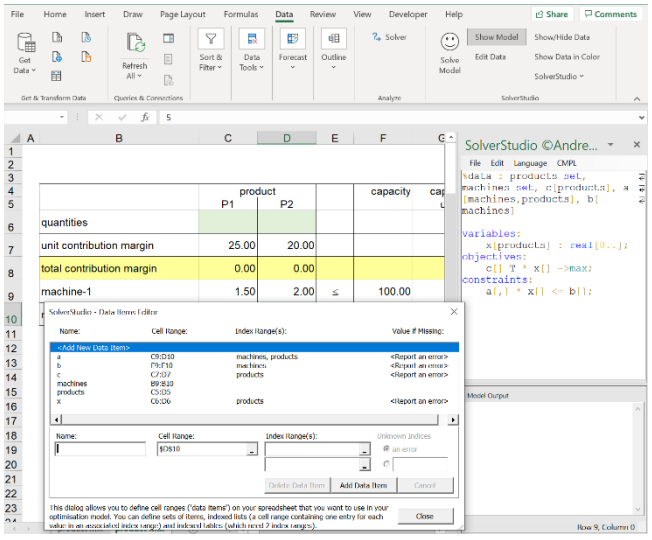


Figure 2. SolverStudio.

SolverStudio is an excellent and very convenient tool, but unfortunately the project does not seem to have been continued seriously, as the last update was in 2016. The interfaces to the languages depend on IronPython 2.7 which is no longer up to date. They would have to be redeveloped for IronPython 3.4, which is currently only available as an alpha version [23]. Another disadvantage is that both SolverStudio and the AIMMS Excel add-in are only available for Windows and not for macOS.

One of the languages supported by SolverStudio is CMPL [24], whose interface to SolverStudio was developed by the author of this paper. The problem with the non-updated SolverStudio and the consideration of all the advantages and disadvantages of all available tools led to the motivation to create an easy-to-use interface between this modelling language and Excel, which allows interactive work and is available for Windows and macOS.

This paper describes this interface. After a short introduction to CMPL, the interface CmplXlsData is explained, followed by an example.

II. CMPL

<Coliop|Coin> Mathematical Programming Language (CMPL) is a mathematical programming language and a system for the mathematical programming and optimisation of linear and quadratic optimisation problems. The CMPL syntax is similar in formulation to the original mathematical model but also includes syntactic elements from modern programming languages. CMPL is intended to combine the clarity of mathematical models with the flexibility of programming languages [25].

A typical LP problem is the product-mix problem. The aim is to find an optimal quantity for the products, depending on given capacities. The objective function is defined by the profit contributions per unit c and the variable quantity of the products x. The constraints consist of the use of the capacities and the ranges for the decision variables. The use of the capacities is given by the product of the coefficient matrix a and the vector of the decision variables x and restricted by the vector of the available capacities b. The simple example

$$25x_1 + 20x_2 \rightarrow \max!$$
$$s.t.$$
$$1.5x_1 + 2x_2 \leq 100$$
$$12x_1 + 5x_2 \leq 150$$
$$x_1, x_2 \geq 0$$

can be formulated in CMPL as follows:

```
01 par:
02   c := (25, 20);
03   a := ((1.5,  2), (12, 5));
04   b := (100, 150);
05 var:
06   x[defset(c)] : real[0..];
07 obj:
08   c^T * x ->max;
09 con:
10   a * x <=b;
```

Listing 1. The product-mix problem in CMPL

A CMPL model usually consists of four sections. In the `par` section (lines 01–04), sets and parameters (here the vectors c and b and the matrix a) must be specified. The `var` section (lines 05–06) is used to define the variables of the problem. In line 06, a vector `x` of non-negative continuous variables is defined using the definition set of the parameter vector `c`. The objective function in the `obj` section (lines 07–08) and the constraints in the `con` section (lines 09–10) are specified by vector and matrix multiplications.

CMPL executes CBC, GLPK, Gurobi, SCIP or CPLEX directly to solve the generated model instance. Because it is also possible to transform the mathematical problem into MPS or Free-MPS, alternative solvers can be used.

CMPL is a COIN-OR [26] open-source project initiated by the Technical University of Applied Sciences Wildau. Binaries for Windows, macOS and Linux can be downloaded free of charge from http://coliop.org/.

The CMPL distribution contains Coliop, which is CMPL's Integrated Development Environment (IDE), application programming interfaces (APIs) for Python3 and Java (pyCmpl and jCmpl) and, in CMPLServer, [27] an XML-RPC-based web service for distributed and grid optimisation.

III. CMPLXLSDATA

CmplXlsData was introduced with CMPL version 2.0 and is CMPL's interface for reading sets and parameters from an Excel file and for writing optimisation results to an open Excel file. If the Excel file is not open, CMPL will open it automatically and the results of the optimisation can be seen immediately. Please note, this feature is only available on Windows and macOS if Microsoft Excel is installed. CmplXlsData is mainly implemented with Python3 using the (open-source) Python for Excel library by xlwings [28].

As in SolverStudio or the AIMMS Excel add-in, a user must specify which data from an Excel file should be read into a CMPL model and which results should be written back. These specifications are made in a CmplXlsData file. A CmplXlsData file is a plain text file that contains the definition of parameters and sets with the cell addresses of their values in the specified Excel file in a particular syntax. Additionally, the optimisation results to be written to Excel with their cell addresses can be specified in this file.

A CmplXlsData file contains usually the three sections `@source`, `@input` and `@output`.

The `@source` section is intended to specify the Excel file and optionally the sheet to be used to read sets and parameters and to write the optimisation results.

@source	Section for specifying the Excel file and the default sheet
%file <*fileName***>**	Name of the Excel file
*[***%sheet <***sheetName***>***]*	Optional argument to specify the name of the active sheet In each entry for the inputs and the outputs, the sheet can be specified directly.

Listing 2. Source section

In the `@input` section, the sets and parameters to be read into the CMPL model have to be specified with their cell ranges.

@input	Section for specifying sets and parameters to be read into CMPL
%name **<***cell***>**	A scalar parameter `name` is assigned a single string or number available in Excel at the specified cell.
%name **set***[***[***rank***]***]* ↵ **<***cellRange***>**	Definition of an n-tuple set A set definition starts with the name followed by the keyword set. For n-tuple sets with $n > 1$ the rank of the set is to be specified enclosed by square brackets. The set is assigned the entries available in Excel in the cells specified in the cell range reference.
*%name***[***set[,set1,* ↵ *...]*** **<***cellRange***>**	Definition of a parameter array The specification of a parameter array starts with the name followed by one or more sets, over which the array is defined. The data entries can be strings or numbers and have to be found at the specified cell range reference in Excel.

Listing 3. Input section

The `@output` section specifies the optimisation result elements to be written to the Excel file. These results are displayed directly in the Excel file.

@output	Section for specifying the optimisation results to be written to Excel
*%name***.activity** ↵ **<***cell***>** *%name***.type <***cell***>** *%name***.lowerBound** ↵ **<***cell***>** *%name***.upperBound** ↵ **<***cell***>** *%name***.marginal** ↵ **<***cell***>**	Singleton variable or constraint For a singleton variable or constraint named `name`, the activity, type, limits and dual values can be written to Excel in the `cell`. The name is followed by a dot and one of the keywords (`activity`, `type`, `lowerbound`, `upperbound`, `marginal`) for the information to be written to Excel.
*%name***[***set[,set1,* ↵ *...]***].activity** ↵ **<***cellRange***>**	Arrays of variables or constraints A complete array of variables or constraints named `name`, the activity, type, limits and dual

%*name*[*set*[,*set1*, ...]].**type** <*cellRange*> **%***name*[*set*[,*set1*, ...]].**lowerBound** <*cellRange*> **%***name*[*set*[,*set1*, ...]].**upperBound** <*cellRange*> **%***name*[*set*[,*set1*, ...]].**marginal** <*cellRange*>	values can be written to Excel in the specified cell range. The specification of an array of variables or constraints starts with the name followed by one or more sets, over which the array is defined. This is followed by a dot and one of the keywords for the attributes `activity`, `type`, `lowerbound`, `upperbound`, `marginal` of the result information to be written to Excel.
%objName <*cell*>	Writes the name of the objective function to Excel in the specified cell
%objSense <*cell*>	Writes the objective sense
%objValue <*cell*>	Writes the objective function value
%objStatus <*cell*>	Writes the status of the objective function
%nrOfVars <*cell*>	Writes the number of the variables
%nrOfCons <*cell*>	Writes the number of the constraints
%solverName <*cell*>	Writes the name of the solver
%solverMsg <*cell*>	Writes a message of the solver

Listing 4. Output section

To connect a CmplDataFile with the CMPL model, the command line option `xlsdata` is used. The arguments of this command line option define parameters and sets for CMPL, whose source Excel file and the corresponding cell ranges are specified in a CmplXlsData file. It is recommended that this command line option be used in the CMPL header.

```
%xlsdata [filename] : [set1 set[[rank]]] ↵
        [, set2 set[[rank]] , … ]

%xlsdata [filename] : [param1] ↵
        [, param2 , … ]

%xlsdata [filename] : [paramArray1[set]] ↵
        [, paramArray2[set] , … ]
```

Listing 5. CmplXlsData in CMPL header

The first argument is the file name. If the file name contains white spaces, the name must be enclosed in double quotes. If *filename* is not specified, the generic name *model*`.xdat` is used, where *model*`.cmpl` is the name of the CMPL file. After the colon, the sets, scalar parameters and parameter arrays to be read can be specified and separated by commas.

IV. CMPLXLSDATA EXAMPLE

In this section, a transhipment problem is used to illustrate the functionalities of CmplXlsData. A transhipment model is intended to organise an optimal supply of a homogeneous good between a set of sources (origins, suppliers), a set of transhipment nodes and a set of sinks (destinations, customers) in order to minimise the total transportation cost (or distances, times, etc.) [29].

In this example, a transport plan between three plants, two warehouses and four distribution centres is to be determined to minimise the total transport costs. The unit transport costs are shown in the picture below as weights at the edges. The capacity of each possible road (edge) is restricted to 500 units due to the vehicle pool.

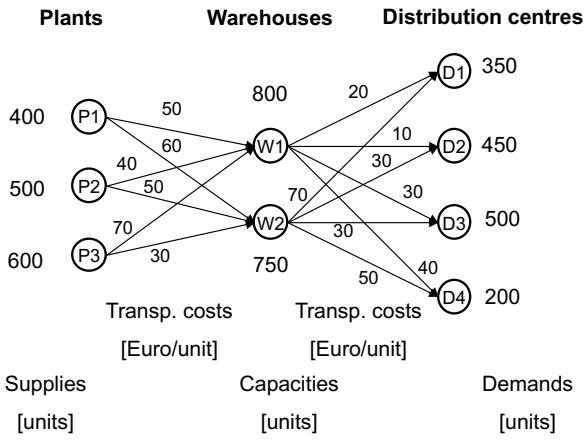
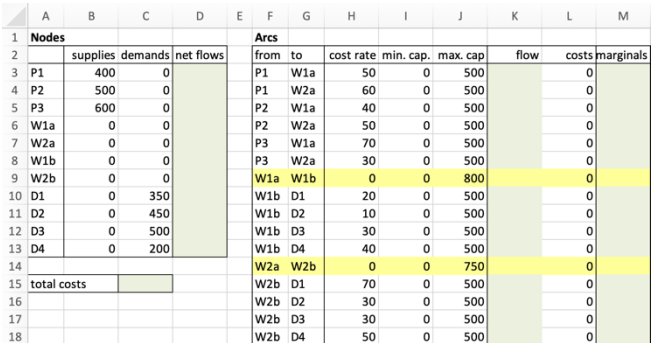


Figure 3. Transhipment problem example.

In the first step, an Excel file `transhipment.xlsx` containing the sheet `transhipment` is created. As shown in Figure 4, the IDs, supplies and demands of the nodes are given in the columns A to C. Please note that the transhipment nodes W1 and W2 have to be split (W1a, W1b, W2a, W2b) owing to their capacities and the fact that the min-cost flow model does not allow capacities for nodes [30]. Therefore, each transhipment node must be split into two nodes, with a cost-free edge connecting the two nodes. The maximum flow on such an edge equals the capacity of the transhipment node. Consequently, the definition of the 2-tuple set of the edges in the columns F and G also contains these two auxiliary edges W1a to W1b and W2a to W2b in addition to the normal edges.

The corresponding cost rates, minimum and maximum capacities of these edges are given in columns H to J. The columns K and M are for the activities and marginal values of the flow variables. The costs in column L yield the product of the activities in column K and the cost rates in column H. These values are displayed after the optimisation in addition to the objective function value in cell C15 and the activities of `netFlow` constraints of the nodes in column D.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Nodes					Arcs							
2		supplies	demands	net flows		from	to	cost rate	min. cap.	max. cap	flow	costs	marginals
3	P1	400	0			P1	W1a	50	0	500		0	
4	P2	500	0			P1	W2a	60	0	500		0	
5	P3	600	0			P2	W1a	40	0	500		0	
6	W1a	0	0			P2	W2a	50	0	500		0	
7	W2a	0	0			P3	W1a	70	0	500		0	
8	W1b	0	0			P3	W2a	30	0	500		0	
9	W2b	0	0			W1a	W1b	0	0	800		0	
10	D1	0	350			W1b	D1	20	0	500		0	
11	D2	0	450			W1b	D2	10	0	500		0	
12	D3	0	500			W1b	D3	30	0	500		0	
13	D4	0	200			W1b	D4	40	0	500		0	
14						W2a	W2b	0	0	750		0	
15	total costs					W2b	D1	70	0	500		0	
16						W2b	D2	30	0	500		0	
17						W2b	D3	30	0	500		0	
18						W2b	D4	50	0	500		0	

Figure 4. Transhipment problem in Excel.

The CmplXlsData `transhipment.xdat` file starts in the `source` section with the entry for the Excel file `transhipment.xlsx` and the sheet `transhipment` from which the data is to be read and into which the results are to be written.

```
01 @source
02   %file < transhipment.xlsx >
03   %sheet < transhipment>
04
05 @input
06   %edges set[2] < F3:G18 >
07   %nodes set < A3:A13 >
08
09   %c[edges] < H3:H18 >
10   %d[nodes] < C3:C13 >
11   %s[nodes] < B3:B13 >
12
13   %minCap[edges] < I3:I18  >
14   %maxCap[edges] < J3:J18 >
15
16 @output
17   %x[edges].activity < K3:K18 >
18   %x[edges].marginal < M3:M18 >
19
20   %netFlow[nodes].activity < D3:D13 >
21
22   %objValue < C15 >
```

Listing 6. CmplXlsData file of the transhipment problem

The following `input` section usually starts with the definition of index sets that will later be used for parameter arrays. In line 06, a 2-tuple set `edges` is defined, to which the IDs of the edges stored in cells `F3:G18` are assigned. The following line defines the set `nodes` and assigns the IDs given in the Excel sheet in the cell range `A3:A13`. These sets are used to define the parameter arrays for the cost rates `c` of the edges (line 09), as well as the supplies `s` and demands `d` of the nodes (lines 10 and 11) and assigns the data stored in the cell ranges indicated in the angle brackets. The minimum and maximum capacities (`minCap` and `maxCap`) of the edges are given in lines 13 and 14.

The `output` section is designed to enable all the requested results of the optimisation to be written into the specified Excel sheet. In lines 17 and 18 it is specified that the activities and marginals of the flow variables `x` must be written in the cell ranges `K3:K18` and `M3:M18`. The activities of the `netFlow` constraints of the nodes should be displayed in `D3:D13` and the objective function value in cell `C15`.

These specifications are connected with the corresponding CMPL model using the CMPL header entry `%xlsdata` in the first line of the model.

```
01 %xlsdata : nodes set, s[nodes],
   d[nodes], edges set[2], c[edges],
   maxCap[edges]
02
03 var:
04   { [i,j] in edges: x[i,j] :
05     real[minCap[i, j]..maxCap[i, j]];
06   }
07
08 obj:
09   sum{[i,j] in edges:
10       c[i,j]*x[i,j]} ->min;
11
12 con:
13   {i in nodes : netFlow[i]:
14     sum{j in edges *> [i,*] : x[i,j]}-
15     sum{j in edges *> [*,i] : x[j,i]}=
16     s[i] - d[i];
17   }
```

Listing 7. CMPL file of the transhipment problem

The variables of the model are organised in an array `x`, which is defined by using the 2-tuple set `edges`. They are all continuous variables with lower and upper bounds defined in the vectors `minCap` and `maxCap`. These variables are the flows of the uniform good on the edges (lines 04–06). The objective function to be minimised is defined in the `obj` section (lines 09–10) as the sum over all edges of the product of the unit transport costs `c[i,j]` and the flow `x[i,j]` on the edge. For all nodes, a flow balance constraint `netFlow[i]` has to be created in which the difference between the outgoing and incoming flow on the left-hand side must be equal to the difference between the supply `s[i]` and the demand `d[i]` of this node on the right-hand side (lines 13–17).

The results can be found after the optimisation in the cells specified in the CmplXlsData file as shown in Figure 5.

The planned quantities on the edges can be seen in column K. A few of the edges are unused, whereby the marginal values in column M show the reduced cost of these non-basic variables. For fully utilised edges, the marginals show the shadow prices. For example, the auxiliary edge for the transhipment node W1 has a shadow price of 20 due to the fully used capacity of 800 units. The activities of the `netFlow` constraints written in column D show that all the supply, demand and flow balance constraints of the nodes are satisfied. This transport plan results in a minimum transport cost of 100,500 which is shown in cell `C15`.

 ISBN: 978-1-61208-898-3

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Nodes**					**Arcs**							
2		supplies	demands	net flows		from	to	cost rate	min. cap.	max. cap	flow	costs	marginals
3	P1	400	0	400		P1	W1a	50	0	500	200	10.000	0
4	P2	500	0	500		P1	W2a	60	0	500	200	12.000	0
5	P3	600	0	600		P2	W1a	40	0	500	500	20.000	0
6	W1a	0	0	0		P2	W2a	50	0	500	0	0	0
7	W2a	0	0	0		P3	W1a	70	0	500	100	7.000	0
8	W1b	0	0	0		P3	W2a	30	0	500	500	15.000	-50
9	W2b	0	0	0		W1a	W1b	0	0	800	800	0	-20
10	D1	0	350	-350		W1b	D1	20	0	500	350	7.000	0
11	D2	0	450	-450		W1b	D2	10	0	500	450	4.500	0
12	D3	0	500	-500		W1b	D3	30	0	500	0	0	10
13	D4	0	200	-200		W1b	D4	40	0	500	0	0	0
14						W2a	W2b	0	0	750	700	0	0
15	total costs		100.500			W2b	D1	70	0	500	0	0	40
16						W2b	D2	30	0	500	0	0	10
17						W2b	D3	30	0	500	500	15.000	0
18						W2b	D4	50	0	500	200	10.000	0

Figure 5. Results for the transhipment problem in Excel

CMPL's Excel interface CmplXlsData is not an add-in, but it enables interactive working, as both Excel and CMPL's IDE Coliop can be run simultaneously (Figure 6). Each change to the data organised in Excel results in new solutions when CMPL is restarted, which are immediately displayed in the specified cell ranges in Excel. If the Excel file is not open, CMPL opens it automatically.

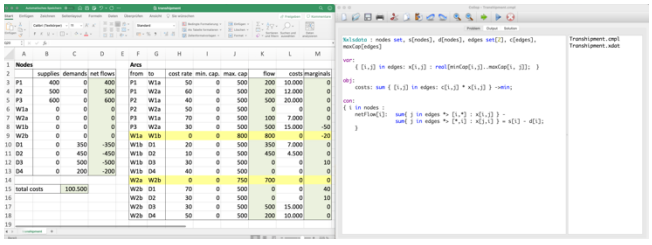


Figure 6. Interactive mode with Excel and Coliop

Unlike the AIMMS Excel add-in and SolverStudio, CmplXlsData is available for Windows and macOS. An additional installation routine to connect Excel and CMPL is not required. A user only needs to install Excel and CMPL. The connection between the two is established automatically by CmplXlsData.

As shown in the example, this interface is easy to use as it provides a simple and structured syntax similar to CmplData, which is another data interface of CMPL.

V. SUMMARY

This paper deals with the combination of spreadsheet programs and optimisation software.

Spreadsheet programs, which are easy to use and available at most workplaces, are essential for preparing and supporting decisions. It is reasonable to connect spreadsheet programs with optimisation software to combine the modelling capabilities of optimisation software with the data maintained in spreadsheets. Such software solutions can be divided into spreadsheet add-ins and data interfaces, which are investigated in this work. Add-ins in spreadsheet programs such as the Excel solver add-in allow interactive work, although modelling with cell ranges does not seem suitable for complex models. Data interfaces to spreadsheets of algebraic modelling languages, which are excellent for modelling complex problems, do not allow interactive work. In addition, there are some approaches that combine modelling languages with Excel in the form of an Excel add-in and thus combine interactive work with excellent modelling possibilities. Unfortunately, these are only available for Windows and some of them seem to have been discontinued.

The consideration of all the advantages and disadvantages of the available tools led to the motivation to create CmplXlsData, which is CMPL's interface to Excel. It is an easy-to-use interface between this modelling language and Excel, which allows interactive work and is available for Windows and macOS. This paper describes this interface with its main functionalities and an illustrative example.

REFERENCES

[1] Microsoft, "Define and solve a problem by using Solver," 2021. [Online]. Available: https://support.microsoft.com/en-us/office/define-and-solve-a-problem-by-using-solver-5d1a388f-079d-43ac-a7eb-f63e45925040. [retrieved: July 2021].

[2] Frontline, "Excel Solver – Overview and Example" 2021. [Online]. Available: https://www. solver.com/excel-solver-overview-and-example. [retrieved: July 2021].

[3] LibreOffice, "Solver," 2021. [Online]. Available: https://help.libreoffice.org/7.0/en-US/text/scalc/01/solver.html?DbPAR=CALC. [retrieved: July 2021].

[4] OpenSolver, "About OpenSolver," 2021. [Online]. Available: https://opensolver.org. [retrieved: July 2021].

[5] A. J. Mason, "OpenSolver – An Open Source Add-in to Solve Linear and Integer Progammes in Excel," in *Operations Research Proceedings 2011*, Berlin and Heidelberg, pp. 401-406, 2012.

[6] Frontline, "Solver - Add-on for Google Sheets," 2021. [Online]. Available: https://workspace.google.com/marketplace/app/solver/539454054595. [retrieved: July 2021].

[7] Palisade, "Evolver - Innovative Optimization for Spreadsheets," 2021. [Online]. Available: https://www.palisade.com/evolver/default.asp. [retrieved: July 2021].

[8] L. S. Inc., " What'sBest! 17.0 - Excel Add-In for Linear, Nonlinear, and Integer Modeling and Optimization," 2021. [Online]. Available: https://www.lindo.com/index.php/products/what-sbest-and-excel-optimization. [retrieved: July 2021].

[9] Addinsoft, "The leading Optimization Solver for Microsoft Excel®," 2021. [Online]. Available: https://www.xloptim.com/en. [retrieved: July 2021].

[10] A. J. Mason, "SolverStudio: A New Tool for Better Optimisation and Simulation Modelling in Excel," *INFORMS Transactions on Education,* vol. 14, no. 1, pp. 45-52 , 2013.

[11] AMPL, "AMPL Direct Spreadsheet Interface," 2021. [Online]. Available: https://ampl.com/resources/new-features/spreadsheets/. [retrieved: July 2021].

[12] M. Software, "MPL for Windows Manual - Import Data from Excel Spreadsheet," 2021. [Online]. Available: http://www.maximalsoftware.com/mplman/mpw07060.html. [retrieved: July 2021].

 ISBN: 978-1-61208-898-3

[13] A. B.V., “AIMMS Excel Library - AXLL,” 2021. [Online]. Available: https://how-to.aimms.com/Articles/85/85-using-axll-library.html. [retrieved: July 2021].

[14] GAMS, “Data Exchange with Microsoft Excel,” 2021. [Online]. Available: https://www.gams.com/latest/docs/UG_DataExchange_Excel.html. [retrieved: July 2021].

[15] IBM, “ILOG CPLEX Optimization Studio/ 20.1.0 / Spreadsheet Input/Output,” 2021. [Online]. Available: https://www.ibm.com/docs/en/icos/20.1.0?topic=sources-spreadsheet-inputoutput. [retrieved: July 2021].

[16] FICO, “The Excel interface,” 2021. [Online]. Available: https://www.fico.com/fico-xpress-optimization/docs/latest/mosel/mosel_data/dhtml/secsetup_sec_secexcelsetup.html. [retrieved: July 2021].

[17] V. DelGobbo, “Integrating SAS® and Microsoft Excel: Exploring the Many Options Available to You,” SAS Institute Inc., https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/2991-2019.pdf, 2019 [retrieved: July 2021].

[18] M. Software, “OptiMax Component Library,” 2021. [Online]. Available: http://www.maximalsoftware.com/optimax/. [retrieved: July 2021].

[19] FICO, “Launching Mosel from Excel using VBAFICO Xpress Optimization Examples Repository /,” 2021. [Online]. Available: https://examples.xpress.fico.com/example.pl?id=excelmosel1. [retrieved: July 2021].

[20] AIMMS B.V., “AIMMS - The Excel Add-In User’s Guide,” https://download.aimms.com/aimms/download/references/AIMMS_excel.pdf, 2016. [retrieved: July 2021]

[21] A. J. Mason, “SolverStudio,” 2021. [Online]. Available: https://solverstudio.org. [retrieved: July 2021].

[22] Microsoft, “Solver Foundation,” 2014. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/msdn10/hh145003(v=msdn.10). [retrieved: July 2021].

[23] .NET Foundation, “IronPython - the Python programming language for .NET,” 2021. [Online]. Available: https://ironpython.net. [retrieved: July 2021]

[24] M. Steglich and T. Schleiff, “CMPL,” 2021. [Online]. Available: http://coliop.org. [retrieved: July 2021]

[25] M. Steglich and T. Schleiff, “CMPL: Coliop Mathematical Programming Language,” *Wildauer Schriftenreihe - Entscheidungsunterstützung und Operations Research,* vol. 1, 2010.

[26] COIN-OR, 2021. [Online]. Available: https://www.coin-or.org. [retrieved: Sep 2021]

[27] M. Steglich, “CMPLServer - An open source approach for distributed and grid optimisation,” *AKWI Anwendungen und Konzepte der Wirtschaftsinformatik,* no. 4, pp. 9-21, 2016.

[28] xlwings, “xlwings - Python for Excel,” Zoomer Analytics GmbH, 2021. [Online]. Available: https://www.xlwings.org. [retrieved: July 2021]

[29] F. Hillier and G. Lieberman, Introduction to Operations Research (International edition), vol. 10th ed., New York et al.: McGraw-Hill, 2015.

[30] M. Steglich, D. Feige, and P. Klaus, Logistik-Entscheidungen: Modellbasierte Entscheidungsunterstützung in der Logistik mit LogisticsLab, Berlin and Boston: De Gruyter, 2016.

 ISBN: 978-1-61208-898-3