Vilnius University
Institute of Mathematics and Informatics
L I T H U A N I A

INFORMATICS (09 P)

RESEARCH AND DEVELOPMENT OF AN OPEN SOURCE GLOBAL OPTIMIZATION SYSTEM

Vaidas Jusevičius

October 2018

Technical Report MII-DS-09P-18-1 October 2017 - 30 September 2021

VU Institute of Data Science and Digital Technologies, Akademijos str. 4, Vilnius LT-08663,
Lithuania
www.mii.lt

Abstract

We consider a research and development of an open source global optimization system. There exists a quite large and solid selection of the commercial optimization modeling systems available for end users, however open source alternatives are basically non existent or significantly lacking in the features offered by their commercial counterparts. Most of the missing functionality is observed in the features related to interaction with modeling system and feedback from the solvers. To bridge this gap we consider building an open source modeling system based on top of the best open source optimization modeling elements currently existing in the world. In the first part we investigate current state and issues in the optimization modeling field by selecting most prominent algebraic modeling languages and modeling systems and performing an extensive both theoretical and experimental investigation of them.

Keywords: Modeling systems, Modeling languages, Optimization, `AIMMS`**,** `AMPL`**,** `GAMS`**,** `PYOMO`

Contents

1 Introduction

In this review we explore and compare existing algebraic modeling languages most frequently used for describing mathematical optimization problems [NEO18]. In the last few decades, there were multiple algebraic modeling languages created, and new ones are still being developed (e.g., `Pyomo`).

Therefore there is a need to review and summarize the current state of the optimization modeling languages by comparing characteristics and performance of the most widely used algebraic modeling languages and expect it to provide insights on what is still missing and where the research and development might be focused in the year to come.

Numerous real-world problems are solved routinely by optimization tools. Many decision problems can benefit from an optimization-type approach. Indeed, the paradigm of optimization seems to be well adapted to needs in different domains such as, economics, engineering, management or physics. In many applications, it is quite natural to formulate the problem as a choice of one solution from a set of possible solutions [FG02].

Mathematical modeling is the art of translating problems from an application area into tractable mathematical formulations whose theoretical and numerical analysis provides insight, answers, and guidance useful for the originating application [KPH04]. Optimization modeling languages bridge the gap between model formulation and the appropriate solution technique [FG02].

Algebraic modeling languages, being a type of optimization modeling languages, store the knowledge about a model, they define the problem and usually do not specify how to solve it. They are declarative languages. The problem is flattened, i.e., all variables and constraints become essentially one-dimensional, and the model is written in an index-based formulation, using algebraic expressions in a way which is close to the mathematical notation [KPH04]. The similarity of the model written in an algebraic modeling language to the mathematical formulation of a problem is a very important aspect which distinguishes algebraic modeling languages from other types of modeling languages, like object-oriented (e.g. `OptimJ`), solver specific (e.g. `LINGO`), general purpose (e.g. `TOMLAB`) modeling languages. This algebraic design approach allows mathematicians without a specific programming or modeling knowledge to be efficient in describing problems to be solved.

It is also important to note that the algebraic modeling language is then responsible for creating a problem instance that a solution algorithm can work on [KPH04]. Since majority of a algebraic modeling languages are proprietary and integral part of a specific modeling system it is important to isolate the responsibilities of a modeling language from an overall modeling system.

So in general we can conclude that algebraic modeling languages are sophisticated software packages that provide a key link between an analyst's mathematical conception

of an optimization model and the complex algorithmic routines that seek out optimal solutions. Algebraic modeling language software automatically reads and interprets a model and data, generates an instance, and conveys the instance to a solver in the required form [Fou13].

2 Algebraic modeling languages

Algebraic modeling languages were game changers because they allowed optimizers to separate model formulation from implementation details [KPH04] while keeping notation close to the mathematical formulation of problem [FG02].

Since the data appears to be more volatile than the problem structure, most modeling languages designers insist on data and model structure being separated [H99]. So central idea in modern algebraic modeling languages is differentiation between abstract models and concrete problem instances [HWW11].

An abstract model separates the declaration of a model from the data used to generate a specific model instance [HWW11]. Model and data together specify a particular instance of an optimization problem for which a solution can be sought. This is realized by replicating every entity of an abstract model over the different elements of the data set. This is often referred to as a set-indexing ability of the algebraic modeling language [FG02].

Model instances are usually defined in `MPS` format [FG02]. `MPS` file [MIP11] format allows to standardize the formulation of linear programs. MPS format can be used today to formulate mixed integer, quadratic or stochastic optimization problems [FG02]. Virtually all LP solvers accept input of model instances expressed in simple text file formats, especially the `MPS` format dating back many decades and various linear LP formats that resemble textbook examples complete with + and = signs [Fou17a].

So basic characteristics of a modern algebraic modeling language could be defined in a following way [KPH04]:

- problems are represented in a declarative way;
- there is a clear separation between problem definition and the solution process;
- there is a clear separation between the problem structure and its data.

Here it must be noted that one of the most important characteristics of modern algebraic modeling languages is their ability to describe non-linear models [KPH04]. So we consider the support for mathematical expressions and operations needed for describing non-linear models an important feature of an algebraic modeling language.

Also it is worth to observe that most interpreters included in todays algebraic modeling languages are based on automatic differentiation [FG02], a process in which the

modeling language can generate derived information from the model description without assistance of the user [KPH04]. This motivates us to include automatic differentiation as an important feature of a modern algebraic modeling language too.

The algebraic expressions that are useful in describing individual objectives and constraints are also useful in describing manipulations of models and transformations of data. Thus almost as soon as modeling languages became available, users started finding ways to adapt model notations to implement sophisticated solution strategies and iterative schemes. These efforts stimulated the evolution within algebraic modeling languages of scripting features, which include statements for looping, testing, and assignment [Fou13]. So at least a minimal support for scripting capabilities is another interesting aspect of a algebraic modeling language to investigate.

For this review we have chosen four different algebraic modeling languages: `AIMMS`, `AMPL`, `GAMS` and `Pyomo`. The selection was based on language recognition in scientific community and it's usage popularity. So the winners of 2012 INFORMS Impact Prize [INF12], a prize awarded to the originators of the five most important algebraic modeling languages were considered. Also the popularity of a language based on NEOS Server [NEO18], a free internet-based service for solving numerical optimization problem, input statistics was taken into consideration. Finally an emerging open-source alternative `Pyomo` was added to the list, since it might be attractive for situations where budgets are tight or where the greatest degree of flexibility is required – such as when new or customized algorithmic ideas are being investigated [Fou17a].

3 Review criteria

In the following review we investigate how the the requirements for a modern algebraic modeling language defined in the previous section are met within each of the chosen languages. Modeling language websites and vendor documentation are used for theoretical comparison. Support of the identified features and capabilities are validated against the documentation algebraic modeling language creators provide. An in depth survey concluded by Robert Fourer in Linear Programming Software Survey [Fou17b] is also used as a reference.

For the practical comparison of the chosen algebraic modeling languages a classical Dantzig Transport Problem was chosen [Dan63].

In this problem, we are given the supplies at several plants and the demands at several markets for a single commodity, and we are given the unit costs of shipping the commodity from plants to markets. The economic question is: how much shipment should there be between each plant and each market so as to minimize total transport cost [Ros18].

The problem formulated as a model in all the reviewed languages is compared based on the following criteria:

- model size in bytes
- model size in number of code lines
- model size in number of language primitives used
- basic model instance creation time i.e. time needed to export concrete model instance to MPS format
- extended (larger data set based) model instance creation time

Sample Transportation Problem model implementations for discussed languages provided by the following sources AIMMS Wikipedia [Wik18], GNU Linear Programming Kit [LL14], GAMS Model Library [GAM18], Pyomo Gallery [Pyo18] are used.

It should be noted that the textual representation of an `AIMMS` model presents the model as a tree of attributed identifier nodes. It reflects the way in which the model is presented to the modeler in the AIMMS IDE, and is typically generated by the AIMMS IDE.

Also it is worth to note that for the sake of simplicity problem model samples are concrete models i.e. data of the model instance is described alongside with model structure.

4 Summary of findings

All of the reviewed modeling languages belong to the algebraic modeling language family so problems are represented in a declarative way. Furthermore since all of them are part of overall modeling system a clear separation between problem definition and the solution process in the context of the modeling system exists.

Separation between the problem structure and it's data is supported in all of the reviewed languages. It should be noted that `GAMS` and `Pyomo` also allows to initiate data structures during their declaration while `AIMMS` and `AMPL` only supports it as a separate step in model instance building process. However, while it might be convenient for building a simple model we do not consider lack of direct data structure initiation as an advantage since in real world cases it's rarely needed.

So we can conclude that all of the reviewed languages fulfill basic characteristics of a modern algebraic modeling language as defined in the previous section.

Comparison of the characteristics for the sample Transportation Problem model implemented in all the reviewed algebraic modeling languages can be seen in the Table 1. To have a more concise view the simplification of model implementations provided in the literature sources is made. All the optional comments, documentation and explanatory texts are removed. All empty lines are excluded. Parts of the code responsible for calling the solver and displaying results are omitted.

While counting language primitives generic functions (*sum*, *for*), data loading directives (*data*), arithmetical and logical operators are excluded.

Table 1: Comparison of Transport Problem models

	AIMMS	**AMPL**	**GAMS**	**Pyomo**
size in bytes	2229	683	652	1207
lines of code	68	24	31	28
primitives used	9	5	8	6

As we see from the table above models implemented in both `AMPL` and `GAMS` are the most compact ones, while model written in `AIMMS` is much more verbose and `Pyomo` is somewhere in the middle.

The reason for `AIMMS` model being much more verbose is in the nature of `AIMMS` modeling system, which propagates model creation using graphical user interface (GUI) while keeping source code of the model hidden from a modeler. So it is natural that there is not that much of the focus on how the model is actually stored. We can argue that while GUI based approach might be convenient to some of the modelers it enforces greater vendor lock-in and makes extensibility and maintainability of the model harder.

`Pyomo` model being larger in file size is mostly caused by the usage of a generic purpose programing language `Python` as building base of `Pyomo` modeling system. Basic `Python` language constructs are not optimized to a specific modeling domain as is in case of `AMPL` and `GAMS` languages.

While comparing number of language primitives required to create a model `AMPL` and `Pyomo` showed best results which allows us to predict that these modeling languages might have a more gentle learning curve.

So we can conclude that in the context of reviewed algebraic modeling languages `AMPL` allows to formulate an optimization problem in the shortest and potentially easiest to learn way.

For performance benchmark, creation of an instance of the transportation problem model defined in each of the algebraic modeling languages was measured. The process was done in the following steps: 1) loading model instance from a problem definition written in the native language of the modeling system; 2) exporting model instance to MPS format; 3) measuring total execution time 4) investigating characteristics of an instance model. Multiple iterations of this process were executed to measure the impact of possible data caching and other improvements being used by the modeling system. Since creators of `AIMMS` system did not respond to the request for an academic license we were not able to include `AIMMS` in to the benchmark. Benchmark was executed under Windows 10 operating system and free licensed versions of the modeling systems.

Characteristics of the created model instances can be seen in the Table 2. We can conclude that all of the modeling languages have created a model instance using same amount of variables and constraints, however the definition of non zero elements is dif-

ferent between GAMS and other modeling systems.

Table 2: Transport Problem Model Characteristics

	Pyomo	**GAMS**	**AMPL**
Constraints	6	6	6
Non zeros	13	19	13
Variables	7	7	7

In the Table 3 model instance creation time measured in milliseconds is provided. We can exhibit that AMPL showed significantly better results compared to others. This allows to conclude that AMPL is the mostly optimized performance wise. On the other hand, Pyomo coming in last is not a big surprise as it is build on top of interpreted programming language Python which in itself adds significant overhead to model loading process. We can also conclude that there is very little or none caching of previous runs being done, since in none of the systems multiple consequent model instance creations lead to a observable performance improvement.

Table 3: Model Instance Generation Time

	1	**10**	**100**
Pyomo	720	7280	79600
GAMS	170	1730	16490
AMPL	30	220	2130

5 Conclusions

From the current research we can conclude that AMPL allows to formulate an optimization problem in the shortest and potentially easiest way while also being the best performing in simple model instance loading times. GAMS is a strong runner up and AIMMS can be considered as being of it's own class of modeling languages as it has taken a pure graphical user interface based approach. Open source alternative Pyomo does not run that far behind while limitations imposed by an interpreted general purpose programming languages it is built on top (Python) leads to the significant performance downsides.

In order to give a more detailed comparison and thoroughly examine characteristics of algebraic modeling languages a more extensive benchmark involving much larger optimization problem models is needed. It would also be of a great benefit to analyze how each of the modeling languages performs within an area of the specific type of optimization problems (linear, non linear, global, mixed integer, quadratic, etc.). So a large and extensive library of sample optimization problems for the analyzed algebraic modeling languages is needed.

We have chosen GAMS Model Library [GAM18] as a reference for creating such a sample optimization problem suite against which future research will be done. To build

such a library automated scripts were created which convert each of the sample problems provided in the GAMS modeling library to a scalar model in the `AMPL`, `GAMS` and `Pyomo` formats. Characteristics of a sample problem models (number of equations, variables, discrete variables, non zero elements, non zero non linear elements) are automatically extracted and noted. Sample problems are also grouped into optimization problem types. In the future work a benchmark using created optimization problem library will be performed to investigate how each of the modeling languages handles optimization problems of different sizes and types. This should give us a solid background to draw final conclusions about each of the algebraic modeling languages and possible future improvements to be included in the building of an open source optimization system.

References

[Dan63] George B Dantzig. The Classical Transportation Problem. In *Linear Programming and Extensions*, pages 299–315. Princeton University Press, 1963.

[FG02] Emmanuel Fragniere and Jacek Gondzio. Optimization modeling languages. *Handbook of Applied Optimization*, pages 993–1007, 2002.

[Fou13] Robert Fourer. Algebraic Modeling Languages for Optimization. In *Encyclopedia of Operations Research and Management Science*, pages 43–51. Springer, 2013.

[Fou17a] Robert Fourer. Linear Programming: Software Survey. *OR/MS Today*, 44(3), June 2017.

[Fou17b] Robert Fourer. Linear Programming Software Survey. https://www.informs.org/ORMS-Today/OR-MS-Today-Software-Surveys/Linear-Programming-Software-Survey, 2017. (Online; accessed 2018-08-24).

[GAM18] GAMS Development Corporation. GAMS Model Library. https://www.gams.com/latest/gamslib_ml/libhtml/index.html, 2018. (Online; accessed 2018-08-07).

[H99] Tony Hürlimann. *Mathematical Modeling and Optimization*, volume 31 of *Applied Optimization*. Springer US, Boston, MA, 1999.

[HWW11] William E. Hart, Jean-Paul Watson, and David L. Woodruff. Pyomo: Modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3(3):219–260, September 2011.

[INF12] INFORMS. INFORMS Impact Prize 2012. https://www.informs.org/About-INFORMS/News-Room/Press-Releases/INFORMS-Impact-Prize-2012, 2012. (Online; accessed 2018-08-24).

[KPH04] Josef Kallrath, Panos M. Pardalos, and Donald W. Hearn, editors. *Modeling Languages in Mathematical Optimization*, volume 88 of *Applied Optimization*. Springer US, Boston, MA, 2004.

[LL14] Lopaka Lee and Louis Luangkesorn. GNU Linear Programming Kit. https://github.com/cran/glpk/blob/master/inst/doc/transport.mod, 2014. (Online; accessed 2018-08-07).

[MIP11] MIPLIB Developers. MPS input format. http://miplib.zib.de/miplib3/mps_format.txt, 2011. (Online; accessed 2018-08-07).

[NEO18] NEOS Server. Neos Solver Access Statistics. https://neos-server.org/neos/report.html, 2018. (Online; accessed 2018-03-25).

[Pyo18] Pyomo. Pyomo Gallery. https://github.com/Pyomo/PyomoGallery, 2018. (Online; accessed 2018-08-07).

[Ros18] Richard E. Rosenthal. A GAMS Tutorial. https://www.gams.com/latest/docs/UG_Tutorial.html, 2018. (Online; accessed 2018-08-07).

[Wik18] Wikipedia contributors. AIMMS — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=AIMMS&oldid=836119826, 2018. (Online; accessed 2018-08-08).