UCL CHEMICAL ENGINEERING

PROFESSOR

l papageorgiou, dipl. eng., phd, dic, eur.ing

Prof. [Marian Gheorghe](https://peerj.com/user/15943/)
Academic Editor, PeerJ Computer Science

 London, July 11th 2018

Dear Prof. Gheorghe,

Thank you for handling our manuscript “An integrated platform for intuitive mathematical programming modeling using LATEX” by C.P. Triantafyllidis and L.G. Papageorgiou (#28593) which was submitted for possible publication in *PeerJ Computer Science*. We would also like to thank the reviewers for reading our manuscript in detail and making valuable comments. We believe that we have addressed all the comments raised and a detailed response together with a list of changes is provided below (reviewers comments in boldface and our response in normal font and any additions directly made into the original manuscript in italics). To make it more clear, we have highlighted all the additions in the manuscript and provided a separate version as well without highlighting.

**Responses to Reviewer #1:**

1. **This paper is well-written and easy to follow. While I think it is suited for publication, I did make a few notes in the main PDF that need to be addressed first. I do appreciate the authors' effort to offer a solution that connects the model description in LaTeX to the optimization model to be solved (in this case, via Pyomo). However, this could potentially generate a "philosophical discussion" among researchers, and I would be interested in knowing the authors' views on the following arguments.\* Some people would argue that coding the model in a programming language is better because of flexibility (i.e., the language allows one to do more than just specifying objective function, constraints, and variables). Others would say that the additional flexibility is unnecessary as the sole purpose is to write a model in as much high-level and human-readable as possible, and then just solve it. I tend to prefer having additional flexibility, so using the proposed approach, which takes another step back (i.e., it's at a higher abstraction layer), would go against that. What do the authors think? Please see the next point.\* On the other hand, I would argue that using the proposed framework could have the advantage of more easily detecting (logic) errors in the model implementation. In other words, it may be easier to detect typos in the mathematical formulation than in a Pyomo, JuMP, GAMS, AIMMS, etc. model. I do think this an important pro of the proposed approach, which does not seem to be mentioned in the text.\* Another potential disadvantage of the proposed framework (maybe for some people more than others) is the fact that it currently has some stringent rules about the mathematical model description. For example, it is customary to number equations/constraints, whether in a monolithic model block or interspersed with text explaining the meaning of each of them and, sometimes, the symbols in them. In other words, I am afraid this may result in duplicated work, i.e., the researcher would type the paper in LaTeX with all the equations and explanatory text, and would also have to create a separate .tex file with the model description that can be understood by the proposed framework. Perhaps these issues will be addressed in the future, in which case, I think it would be appropriate to say a few words about "future work" in the text.Lastly, I think it would be appropriate to include timing information for each major step in the supplemental material, or at least add mentions in the main text. For example, how much time does it take to parse the .tex file, and then to generate the model in Pyomo, and then to solve it (this one is shown in the solver log)? Does it take much longer to parse the .tex file (which is the focus of this work) than all the other steps?**

In order to avoid any misleading conclusions concerning whether this is the first workable scheme for using LaTeX in mathematical programming modeling, we have now adjusted two sentences in the main text as follows:

(lines 18-19, page 1):

*Overall, this is a presentation of a complete workable scheme on using LaTeX for mathematical programming modeling ….*

 (lines 100-101, page 5)

*This is a complete prototype workable scheme to address how LaTeX could be used as an input language to perform mathematical programming modeling,…*

Flexibility in the model generation is indeed a major advantage of the first approach in building an optimization model, that is, in programming it directly in a given programming language. However, one must admit that the proposed framework has its trade-offs as everything in Computer Science. For a significantly large portion of optimization models from the literature, we were able to generate and solve them without this “extra flexibility”. To counter this potential disadvantage, one also may argue as indicated that among peers, exchanging optimization models written in LATEX notation and not in an algebraic modelling language (or even worse, as raw code in a programming language) can accelerate typo and logic error detection (which was added in the advantages of the platform mentioned in the abstract and in the discussion section as:

“*iv) easier typo and logic error detection in the description of the models*” (line 16, page 1).

Concerning the descriptive information for the constraints (or other symbols), the user can include in the same input .tex file comments for each constraint which would be easily transferred as comments in the target AML (Pyomo in this case). Comments are currently supported but not transferred to the resulting formal AML representation (Pyomo), and we have added mention of this as part of the future work in the conclusions as:

“*the ability to embed explanatory comments in the input model file which would transfer to the target AML*” (line 554, page 24). Indeed, this would polish the framework significantly.

Concerning cpu time, we have added a table (Table1) at the end of the Supplementary Information file, with cpu-times for each of the three processing steps of the platform (parser, Pyomo model generation, solver level) for the given examples. We have also added a small paragraph in the Discussion section as follows (line 521, page 22):

*The cpu time required for each step in the modeling process of the … of the modeling process.*

1. **What do authors mean here? Is "application sector" an area of expertise, e.g., power systems optimization? I am not sure I follow it.**

To avoid any misleading conclusions, we replaced this bullet point with the advantage of the platform to enable easier detection of errors in the description of the models as noted in previous point (line 16, page 1)..

1. **What do the authors mean by "easier for the parser" in this context? Does the code generation take longer if the user opts for the \frac command instead of using the forward slash? While I agree that it may be easier to read with the \frac command, I think it is irrelevant for the use which method to choose, assuming the generated code will be correct and equivalent. Unless, there are side effects of using an approach versus another. it seems that the next paragraph addresses some of these concerns, so I am unsure if saying that one approach is easier to handle than the other adds value in text, unless the side effects, if any, are specified.**

The phrasing here attempts to denote that while a \frac environment is compact and easy to read for the eye, in the background the parser post processes this information in the analytical form to avoid overcomplicating the parser. The equivalency of these two representations is maintained no matter the choice of the user input preference. No cpu-time overheads are associated with this. We changed the phrasing to simplify the imparted meaning. We have added the following text to reflect this (line 223, page 10):

*When handling fractions, the user can employ the frac environment to generate them; the parser in the background always though processes the analytic form (the same applies with the distributive form of multiplications), no matter if the initial input was done using the frac environment.*

1. **Are variable bounds supported? For example, a relaxed binary variable is a real variable with lower and upper bounds of 0 and 1, respectively. Or a more general case in which an integer variable has lower and upper bounds of 0 and the value based on parameter, i.e., it is parameterized? Or even if it's not parameterized, but neither bounds are +/- infinity.**

For the prototype version of the platform the user can include bounds on the variables as additional constraints. This has been added as part of future work in the conclusion section as follows (line 555-556, page 24) as follows:

*extending the functionality concerning bounds on the variables.*

1. **Does the parser understand \left( and \right)? Or other commonly used brackets, such as \left[ and \left\{? These may be used when there are fractions inside the brackets, and again to make the mathematical expression easier to read. From the text, it only seems that ordinary brackets, ( and ), are supported.**

The parser indeed only understands the ordinary brackets. We agree that further incorporation of LATEX inherent commands will greatly improve the representation of the models and is also part of the future work which we added in the manuscript in the conclusion’s section (line 556-557, page 24)*:*

*as well as adding further support to built-in LaTeX commands (such as \backslash left[ \;) which would capture more complex mathematical relationships.*

**Responses to Reviewer #2:**

**Basic reporting**

1. **The paper is well-organized, and the use of the English language is generally good.**

**In line 84, the authors state that “this is the first prototype workable scheme to address how LaTeX could be used as an input language to perform mathematical programming modeling.” This statement is not necessarily true. I encourage the authors to take a closer look at MOSAICmodeling (www.mosaic-modeling.de), which is a modeling and optimization framework based on a LaTeX-style syntax for inputting algebraic and differential equations. A comparison between MOSAICmodeling and the proposed framework would be helpful in terms of evaluating the advantages of this contribution.**

We have added the necessary citation and included a paragraph (line 58, page 3) which addresses the comparison between the two as follows:

*A systems approach…… whereas the ability to export in many other formats is given in the MOSAIC environment.*

1. **In lines 92-97, 22 references are listed with no further details, which is not very useful for the reader. Also, this seems to be a fairly biased list, with contributions mostly from the process systems engineering (PSE) community and many papers co-authored by the authors of this work. I recommend selecting a more representative set of papers from a wider range of fields, considering that the topic addressed here is of interest to more than just the PSE community.**

We have now redefined the list of references as by removing a number of PSE works and adding some new ones. Below we list the references included which are categorised per field of application:

Data mining:

Yang, L., Liu, S., Tsoka, S., Papageorgiou, L. G., 2017. A regression tree approach using mathematical programming. Expert Systems with Applications 78, 347 – 357.

Machine Learning/Artificial Intelligence:

Tanveer, M., 2015. Robust and sparse linear programming twin support vector machines. Cognitive Computation 7 (1), 137–149.

Physics:

Silva, J. C., Bennett, L., Papageorgiou, L. G., Tsoka, S., 2016. A mathematical programming approach for sequential clustering of dynamic networks. The European Physical Journal B 89 (2), 39.

Management:

Sitek, P., Wikarek, J., 2015. A hybrid framework for the modelling and optimisation of decision problems in sustainable supply chain management. International Journal of Production Research 53 (21).

S. Liu and L.G. Papageorgiou, “Fair Profit Distribution in Multi-echelon Global Supply Chains via Transfer Prices”, Omega, 80, 77-94 (2018).

Environmental Modelling:

Triantafyllidis, C. P., Koppelaar, R. H., Wang, X., van Dam, K. H., Shah, N., 2018. An integrated optimisation platform for sustainable resource and infrastructure planning. Environmental Modelling Software 642, 101, 146 – 168.

Operations Research:

Cohen, M. C., Leung, N.-H. Z., Panchamgam, K., Perakis, G., Smith, A., 2017. The impact of linear optimization on promotion planning. Operations Research 65 (2), 446–468.

Romeijn, H. E., Ahuja, R. K., Dempsey, J. F., Kumar, A., 2006. A new linear programming approach to radiation therapy treatment planning problems. Operations Research 54 (2), 201–216.

Systems Biology:

Mitsos, A., Melas, I. N., Siminelakis, P., Chairakaki, A. D., Saez-Rodriguez, J., Alexopoulos, L. G., 2009. Identifying drug effects via pathway alterations using an integer linear programming optimization formulation on phosphoproteomic data. PLOS Computational Biology 5 (12), 1–11.

Melas, I. N., Samaga, R., Alexopoulos, L. G., Klamt, S., 2013. Detecting and removing inconsistencies between experimental data and signaling network topologies using integer linear programming on interaction graphs. PLOS Computational Biology 9 (9), 1–19.

Mathematical Modelling:

Kratica, J., Dugoija, D., Savi, A., 2014. A new mixed integer linear programming model for the multi level uncapacitated facility location problem. Applied Mathematical Modelling 38 (7), 2118 – 2129.

Security:

Mouha, N., Wang, Q., Gu, D., Preneel, B., 2012. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 57–76.

**Experimental design**

**1) The proposed approach is sound and it makes natural sense to divide the work process of the parser into three parts: objective function, constraints, and variables. However, I have some questions that I hope the authors could answer in their response or address in the revised manuscript:**

1. **The authors state that the parser can handle Greek letters. How about variables that consist of a Greek and a Latin letter, e.g. \Delta t?**

This is currently not a supported feature. We cover all of the supported Greek letter representations built-in LATEX distributions but not combinations of them with normal Latin Characters. Extending this is straightforward in terms of implementation for our future work.

1. **How does the parser handle superscripts? Does the parser differentiate between superscripts and exponents?**

We currently do not support superscripts. The caret symbol (“^”) is to be used in our future work to support nonlinearity as a power on variable(s).

1. **In the literature, summations are often defined over sets instead of explicit conditions, e.g. \sum\limits\_{i \in K} with K being a subset of the full-cardinality set I. Can the parser handle this case?**

Of course, dynamic (sparse) sets are supported. For instance, in 2.2 (Supplementary information) we demonstrate the use of this functionality:

\sum\limits\_{n,e: (n,e) \in LK }^{}(X\_{n,e,m}) (summation)

Or

 ∀ m,(n,e) ∈ LK (constraint)

The sets can be activated either within summation expressions or for the entire constraint generation.

1. **Looking at the example in line 347, it seems to be possible to use subsets for constraint generation. How are these subsets defined as these definitions usually do not explicitly appear in the LaTeX formulation of an optimization model? Is this information possibly directly extracted from the .dat file?**

A set found to be dynamic in the description of the LaTeX input file is recognized on the fly and expected to be found correctly defined (with the associated indexes which make up the set) in the AMPL data file. An error is generated otherwise.

**Validity of the findings**

**The validity of the proposed approach is nicely demonstrated in an illustrative example. However, some information on the platform’s ability to detect errors in the LaTeX formulation would be useful. For example, in the illustrative parsing example, what happens if in one of the constraints, the user forgot to write one of the two indices of the variable x?**

Sanity checking is a large portion of our future work. Issues such as defining for instance the summation symbol in a different manner or missing closing brackets/parentheses normally generate warning/exiting messages. Additionally, logical errors which might not be captured by the parser, will eventually be captured by the immediate Pyomo model compilation. In this specific case the error would be caught if for instance variable X was defined with two indices, and then appearing with only one of those.

**Moreover, I personally would welcome a discussion on the challenges of extending this platform to the nonlinear case.**

Constitutes our future work and the challenges have been briefly mentioned in discussions section (line 539, page 23) as follows:

*Additionally, another challenge constitutes the extension of the platform to support nonlinear terms, where each term itself can be a combination of various operators and mathematical functions.*

Also in the conclusion section (lines 552-553, page 23) as follows:

*Future work could possibly involve expansion to support nonlinear terms…*

-----------------------------------------

Minor note: Affiliationsfor CPT have been updated.

We believe that by addressing the reviewers comments the quality of the manuscript has greatly improved. Looking forward to your final decision.

Yours sincerely,

Prof. Lazaros Papageorgiou