Generation of MINLP Problems for Process Synthesis Using Phenomena-based Building Blocks

Erik Eschea\*, David Kronea, Jens-Uwe Repkea

aProcess Dynamics and Operations Group, Technische Universität Berlin, Str. des 17. Juni 135, D-10623 Berlin, Germany

erik.esche@tu-berlin.de

Abstract

Formulation and solution of superstructure optimization problems for process synthesis is still challenging. This concerns both the formulation of suitable mathematical models and optimization problems as well as their solution. Here, the focus lies on the formulation and code generation to facilitate subsequent solution. As a novelty, superstructures are generated in MathML / XML form, which are then used to automatically formulated MINLP problems with interfaces to accurate thermodynamic functions. For these large-scale problems, code consistency needs to be addressed early on. MOSAICmodeling’s capabilities for processing MathML / XML models are exploited to this end using language specificators for automatic code generation towards target languages and respective frameworks for solution of the MINLP problems. The feasibility of the novel approach is highlighted with an example of a superstructure with phenomena-based building blocks.

**Keywords**: superstructure, process synthesis, automatic code generation

* 1. Challenges in Optimal Process Synthesis

Optimal process synthesis requires the formulation and solution of complex optimization problems. An example is the generic network of phenomena-based building blocks as demonstrated by Kuhlmann and Skiborowksi (2017). These problems are typically large and frequently require accurate thermodynamic property information, e.g., to predict vapor liquid equilibria (Krone et al., 2022). This can either be achieved by incorporating the thermodynamic models into the process synthesis problem or by interfacing with external tools. Both cases are challenging during the formulation and implementation of synthesis problems.

For process synthesis problems as in (Kuhlmann and Skiborowski, 2017) and (Krone et al., 2022) there is usually a split between the software used for formulation and solution of the MINLP problem and the tool employed for dedicated solution of thermodynamic equations and properties. The former are typically tools such as GAMS, AMPL, or PYOMO, while the latter are frequently specialized thermodynamic property packages such as Aspen Properties, KBC’s MultiFlash, or AmsterChem’s TEA, which perform tasks such as determination of the number of phases, computing phase equilibria, etc. Interfacing between these heterogeneous tools is an error-prone process with disastrous effects for the reliability of results.

MOSAICmodeling (Esche et al., 2017) is a collaborative modeling platform, which follows the paradigm of modelling at the documentation level, i.e., an identity of model formulation and documentation. Models are formulated in MathML / XML, i.e., standardized descriptive markup language, which is flexible and independent of the final implementation in a target language. The backend of MOSAICmodeling instantiates systems according to user specifications regarding index sets and the translation engine converts these instances into programming code (C, C++, Python, Fortran, etc.) or modeling language (GAMS, Pyomo, AMPL, Matlab, etc.). This strategy allows for error-free implementation of models in the form of code but was in the past limited to single language settings.

In the present contribution, we elaborate an approach to exploit the capabilities of the aforementioned thermodynamic property packages for rigorous process synthesis tasks in an error-free fashion. This entails the formulation of synthesis problems as MINLP in MOSAICmodeling with interface definitions for all state variables computed externally. Afterwards, an automatic model decomposition is carried out at the MathML / XML level, which separates the MINLP into two parts: (1) the actual MINLP problem, and (2) the list of required external function calls and their derivatives. These parts are then automatically exported for desired target languages.

Novel decomposition techniques regarding model hierarchy and classifications of variables have been implemented on the MathML / XML level for superstructure problems, which ensures rigor regarding model consistency across heterogeneous target languages.

* 1. Methodology and Implementation
     1. Setting

Of interest in this setting are formulations for superstructure optimization problems for general process synthesis which consist of model equations describing at least mass and energy balances for units or phenomena-based building blocks (PBBs), algebraic constraints governing which parts of the superstructure should be activated or deactivated given their connection to others, and an objective function minimizing, e.g., total annualized costs. In general, these types of problems will amount to large-scale MINLP formulations. A specialty in the work of Kuhlmann and Skiborowski (2017), later adapted by Krone et al. (2022) is the inclusion of accurate thermodynamics in these models. Naturally, this leads to a further increase in the computational complexity of the MINLP formulation. To alleviate the situation, Kuhlmann and Skiborowski (2017) suggested outsourcing thermodynamics to an external engine, e.g., Aspen Properties, AmsterChem’s TEA, KBC’s multiflash, or similar. These external engines ensure solution of the thermodynamic relationships. Nevertheless, setting up the MINLP with external function calls puts an extra burden on the user formulating and solving the overall MINLP.

* + 1. Modeling in MathML / XML

In the scope of the here presented work, we exploit the capabilities of the MathML / XML-based model formulation within MOSAICmodeling and extend it towards superstructure problems. During the model formulation of the above described MINLP, all equations and inequality constraints are entered in LaTeX and then automatically translated to MathML. Regarding the thermodynamic function calls, an alternative strategy is pursued. Instead of directly implementing functions or equations, the users may specify for each that this is to be implemented externally, e.g., in compliance with the CAPE-OPEN standard (COLaN, 2023). For example, in case a liquid phase enthalpy is required, the user specifies as part of an equation system, that this enthalpy is a function of the respective temperature, pressure, and liquid-phase composition. The information is stored in an interface element in XML linking the variables in the equation system to an – as of yet unknown - external engine. In general, this can be done for all thermodynamic properties, which are needed within the model formulation, i.e., enthalpies, entropies, temperatures or pressures of boiling point or dew point, equilibrium coefficients, etc. This way, the model formulation in MathML / XML stays efficient, remains lean, and is still easily understandable to the person formulating the model.

* + 1. Model Decomposition and Code Generation

In examples of Kuhlmann and Skiborowski (2017) and Krone et al. (2022), a hierarchical structure is present in the MINLP: A distribution network connects feed nodes and splitters to product nodes and mixers. An example hereof will be given in section 3. Mixer nodes link to individual PBBs and the outlets of PBBs connect to the splitter nodes of the distribution network. The structure is also present in the MathML / XML formulation as model hierarchy. Naturally, the same types of interfaces for thermodynamics reappear everywhere. On all levels of this hierarchy, there may be interfaces to thermodynamic function calls. Also, nonlinear constraints, integer, and continuous variables can appear throughout.

The modeling engine of MOSAICmodeling simultaneously holds a flat version, e.g., all variables numbered globally, as well as a fully hierarchical version, e.g., model equations and variable namings retain hierarchical structure, of the instantiated MathML / XML model while processing it. Instantiated means that the numbers of PBBs, number of components, product nodes, etc., are already confirmed. This dual representation is exploited for code generation and can be of help during the subsequent solution of the MINLP.

First, thermodynamic calls are captured by the modeling engine across all hierarchy levels and two maps are built. The first links the variable namings of the flat to the hierarchical version. The second lists each function call type (boiling point temperature, liquid enthalpy, etc.) and for each instance (i.e., function call) clearly marks all required variables with their type (temperature, pressure, enthalpy, entropy, liquid/vapor composition, boiling point temperature, ), their engineering unit (K, Pa, kJ/kmol, etc.), their direction from the perspective of the function call (input or output), and their dimensionality (scalar or vectorial).

Second, the MathML / XML model is decomposed in such a fashion that function values and derivatives can be computed externally and linked back to the occurrences within the hierarchy. This entails a complete separation of the function calls from the rest of the model. The link between both separated parts is based on the hierarchical variables in the MathML / XML structure. Based on the structure of the target code, maps are built for the derivatives either based on the flat (numbering scheme) or the hierarchical variable names. To generate code for the solution of the MINLP, two aspects need to be considered: (1) the environment for the solution and (2) the framework for the external function calls. The translation from MathML / XML form to a target language is governed by language specificators (LS) (Tolksdorf et al., 2019). These can be defined and adjusted by the users and are specified in XML form themselves. In general, an individual LS is required per target language, i.e., AMPL, GAMS, Pyomo, C++, FORTRAN, etc.

For the generation of MINLP problems with external function calls, this is augmented to exploit the hierarchical structure and the aggregation of thermodynamic calls as detailed above. In practice, this means that the code generation is split between MINLP problem on the one hand side and calls to external functions on the other. The aforementioned maps are used to link these different sides governed by the properties of the target code. For each an LS is set-up to generate the specific, executable code. Further details hereon will be given below.

* 1. Case Study
     1. Example Superstructure and Component System

The methodology outlined in section 2 is here presented for a process synthesis problem involving phenomena-based building blocks (Krone et al., 2022; Krone et al., 2023). This is applied on a feed stream of n-pentane, n-hexane, and n-heptane, which is to be separated. The optimization problem consists of 1,303 (in-)equality constraints, 62 binary, and 1,123 continuous variables. The superstructure is depicted in Fig. 1.



Figure 1. Generic superstructure with vapour-liquid equilibria blocks (VL-U). The left-hand side shows the distribution network with the connected VL-U PBB and the right-hand side a close-up of the connection of splitter nodes to mixer nodes. Red dashed lines signal vapor phase, blue solid lines liquid phase.

The left-hand side shows the distribution network (blue box) with all possible connections between feed, product nodes, splitters, and mixers, as well as the attached PBBs, which here are countercurrent vapor-liquid contactors, depicted as VL-Us (Krone et al., 2023). These could present sections of, e.g., a distillation column. The whole superstructure is implemented in MOSAICmodeling.

* + 1. Implementation and Solution of the MINLP

For the solution of the MINLP, GAMS is selected, while the external function calls are implemented using CAPE-OPEN’s binary interop architecture (COBIA) (COLaN, 2023). GAMS and COBIA are connected by a dynamic-link library (DLL), which is programmed in C++. On the GAMS side, a solution strategy involving branch & bound algorithms, DICOPT, CONOPT, and IPOPT, is implemented, which is detailed in (Krone et al., 2023). Here, the focus lies on the code generation and connection of GAMS and COBIA. On the GAMS side, our translation engine inside of MOSAICmodeling translates the interface definitions for the thermodynamic function calls into “external equations”, see Fig. 2. For this, the mapping between system variables and interfaces for function calls is used and translated into consistent code. In this case, GAMS operates on a numbering scheme regarding external communication, while internally using the hierarchical variable namings. In Fig. 2, the numbers before the multiplication sign are the IDs of each variable for external communication with the DLL, while the strings after the sign are the hierarchical names of the model as translated from MathML into GAMS-friendly form. For instance, e0s5\_p is the pressure of stream 5 and e0e0e2\_x\_VL\_i1\_j2 is the liquid mole fraction of component 1 on the second tray of the VL-U 2.

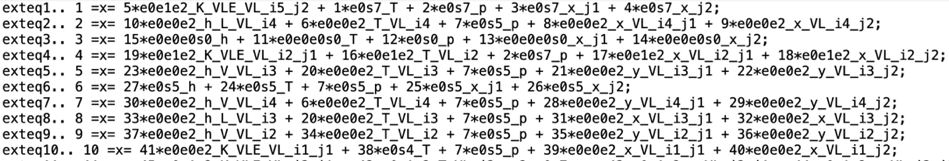


Figure 2. Example of automatically generated external equations in GAMS.

On the C++ / COBIA side, a case differentiation needs to be performed. For those cases, where there is a direct match between XML interfaces and CAPE-OPEN specifications little needs to be done: on the C++ side the components need to match the physical components in the CAPE-OPEN specification and the derivatives need to be structured in the correct way to suit GAMS. This is now automatically done by MOSAICmodeling’s translation engine. For all other cases, reformulations are required to match CAPE-OPEN-compliant function calls. This implies deriving functional dependencies and derivatives via the implicit function theorem. These need to be derived once and implemented directly in the LS in MOSAICmodeling, which then generates consistent code for each call. Fig. 3 depicts an example of an external function implementation on the C++ side. mylocalThermoObj is a COBIA object. x is the vector of all variables at the interface between GAMS and DLL, f the return vector of residuals for all external function calls, and d the derivative vector.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Figure 3. Example of automatically generated C++ and COBIA code for external function calls.

The proposed methodology and described implementation have been tested with a varying number of components (2-3) and number of PBBs (2-6). Consistent, error-free code was generated by the translation engine in MOSAICmodeling for both GAMS and C++. A number of these example MINLP have been successfully solved. At this point we will limit ourselves to showing one cost optimal superstructure which was obtained after 172 h CPU time. Therein, AmsterChem’s TEA supplies the thermodynamic properties via COBIA. Fig. 4 shows the result for four vapor-liquid-type PBBs with a fixed number of five equilibrium stages, which separate a ternary mixture of n-pentane, n-hexane, and n-heptane. The validity of this optimum has been confirmed by comparison with a commercial process simulator. Further details on the optimal solution are published in (Krone et al., 2023).

* 1. Conclusions

Setting up and solving optimization problems for superstructure-based process synthesis is a challenging task. This is particularly the case when multiple software tools are required to reap the benefits of a state-of-the-art environment for solving MINLPs and efficient implementations for complex thermodynamics. With the here presented solution implemented in the translation and code generation engine within MOSAICmodeling a methodology is available to generate highly complex, large MINLP problems. The proposed procedure can be guaranteed to be error-free regarding code implementation and interfacing of different tools. The results validate the reliability of the decomposition and code generation provided by the novel approach and promise even larger process synthesis applications as a next step. In future work we will extend this approach beyond the GAMS / C++ pair to other frameworks such as Pyomo to interface solvers for generalized disjunctive programming.

Figure 4. Configuration of four vapor-liquid-type PBB as the optimal design for the separation of an n-pentane, n-hexane, and n-heptane mixture determined via MINLP optimization (published by Krone et al., 2023).

Acknowledgements

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 523327609.

References

COLaN, 2023, Thermodynamics and Physical Properties interface specification v1.1, https://www.colan.org/specifications/

E. Esche, C. Hoffmann, M. Illner, D. Müller, S. Fillinger, G. Tolksdorf, H. Bonart, G. Wozny,J.-U. Repke, 2017, MOSAIC – Enabling Large-Scale Equation-Based Flow Sheet Optimization, Chem. Ing. Tech. 89, 620–635

D. Krone, E. Esche, N. Asprion, M. Skiborowski, J.-U. Repke, 2022, Enabling optimization of complex distillation configurations in GAMS with CAPE-OPEN thermodynamic models, Comp. Chem. Eng. 157, 107626.

D. Krone, E. Esche, M. Skiborowski, J.-U. Repke, 2023, Optimization-based Process Synthesis by Phenomena-Based Building Blocks and an MINLP-Framework Featuring Structural Screening, *submitted to Comp. Chem. Eng.*

H. Kuhlmann, M. Skiborowski, 2017, Optimization-Based Approach To Process Synthesis for Process Intensification: General Approach and Application to Ethanol Dehydration, Ind. Eng. Chem. Res. 56, 13461–13481.

G. Tolksdorf, E. Esche, G. Wozny, J.-U. Repke, 2019, Customized code generation based on user specifications for simulation and optimization, Comp. & Chem. Eng. 121, 670–684