Cameo: A Python Library for Computer Aided Metabolic Engineering and Optimization of Cell Factories

Cardoso, Joao G.R.; Jensen, Kristian; Lieven, Christian; Hansen, Anne Sofie Lærke; Galkina, Svetlana; Beber, Moritz; Özdemir, Emre; Herrgård, Markus J.; Redestig, Henning; Sonnenschein, Nikolaus

Published in:
A C S Synthetic Biology

Link to article, DOI:
10.1021/acssynbio.7b00423

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Cameo: A Python Library for Computer Aided Metabolic Engineering and Optimization of Cell Factories

João G. R. Cardoso, Kristian Jensen, Christian Lieven, Anne Sofie Lærke Hansen, Svetlana Galkina, Moritz Beber, Emre Özdemir, Markus J. Herrgård, Henning Redestig, and Nikolaus Sonnenschein* 

The Novo Nordisk Foundation Center for Biosustainability, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

ABSTRACT: Computational systems biology methods enable rational design of cell factories on a genome-scale and thus accelerate the engineering of cells for the production of valuable chemicals and proteins. Unfortunately, the majority of these methods’ implementations are either not published, rely on proprietary software, or do not provide documented interfaces, which has precluded their mainstream adoption in the field. In this work we present cameo, a platform-independent software that enables in silico design of cell factories and targets both experienced modelers as well as users new to the field. It is written in Python and implements state-of-the-art methods for enumerating and prioritizing knockout, knock-in, overexpression, and down-regulation strategies and combinations thereof. Cameo is an open source software project and is freely available under the Apache License 2.0. A dedicated Web site including documentation, examples, and installation instructions can be found at http://cameo.bio. Users can also give cameo a try at http://try.cameo.bio.

KEYWORDS: metabolic engineering, genome-scale metabolic models, heterologous pathway predictions, computer-aided design, software, Python

The engineering of cells for the production of chemicals and proteins affects all areas of our modern lives. Beer, yogurt, flavoring, detergents, and insulin represent just a few products that are unimaginable without biotechnology. Engineered cells may further provide solutions to many of mankind’s greatest challenges like global climate, multiple drug resistance, and overpopulation, by producing fuels, novel antibiotics, and food from renewable feedstocks. Manipulating cells to perform tasks that they did not evolve for, however, is challenging and requires significant investments and personnel in order to reach economically viable production of target molecules.1

A central task in developing biotechnological production processes is to reroute metabolic fluxes toward desired products in cells. This task is particularly prone to failure due to our limited understanding of the underlying biology and the complexity of the metabolic networks in even the simplest of organisms. In line with other recent technological advancements, like high-fidelity genome editing through CRISPR/Cas9 and DNA synthesis costs dropping,2 modeling methods are increasingly used to accelerate cell factory engineering, helping to reduce development time and cost.3

Genome-scale models of metabolism (GEMs)4 are of particular interest in this context as they predict phenotypic consequences of genetic and environmental perturbations affecting cellular metabolism.5 These models have been developed throughout the past 15 years for the majority of potential cell factory host organisms ranging from bacteria to mammalian cells. A large repertoire of algorithms has been published that utilize GEMs to compute cell factory engineer-
api.design(product='vanillin')

This function call will run the workflow depicted in Figure 1. It is also possible to call the same functionality from the command line. First, it enumerates native and heterologous production pathways for a series of commonly used host organisms and carbon sources. Then it runs a whole suite of design algorithms available in cameo to generate a list of metabolic engineering strategies, which can then be ranked by different criteria (maximum theoretical yield, number of genetic modifications, etc.).

More advanced users can easily customize this workflow by providing models for other host organisms, changing parameters and algorithms, and of course by including their own methods.

In order to become a community project and attract further developers, cameo has been developed as a modular Python package that has been extensively documented and tested using modern software engineering practices like test-driven development and continuous integration/deployment on travis-ci.org (Figure 2 shows an overview of the package organization).

To avoid duplication of effort, cameo is based on the constraint-based modeling tool cobrapy13 thus providing its users with already familiar objects and methods (see also Figure 2a). Furthermore, cameo takes advantage of other popular tools of the scientific Python stack, like for example Jupyter notebooks for providing an interactive modeling environment14 and pandas for the representation, querying, and visualization of results.15

Accessing published GEMs can be a challenging task as they are often made available in formats that are not supported by existing modeling software.11 Cameo provides programmatic access to collections of models (Figure 2b) hosted by BiGG16 and the University of Minho darwin.di.uminho.pt/models. Furthermore, by relying on the common namespace for reaction and metabolite identifiers provided by the MetaNetX.org project that covers commonly used pathway databases like KEGG,18 RHEA19 and BREnda20 a universal reaction database can be used to predict heterologous pathways (see Supplementary Notebook 7 [v0.11.4, current]).

Most design algorithms rely on solving optimization problems. In order to speed up simulations and ease the formulation of optimization problems, cameo initially replaced the solver interfaces utilized in cobrapy with optlang,12 a Python interface to commonly used optimization solvers and symbolic modeling language. By now, the new optlang-based solver interface has been moved into cobrapy where it replaced the previous solver interfaces. Cameo and cobrapy thus always maintain a one-to-one correspondence of the GEM and its underlying optimization problem, greatly facilitating debugging and efficient solving by enabling warm starts from previously found solutions.21 Furthermore, being based on sympy,22 optlang enables the formulation of complicated optimization problems using symbolic math expressions, making the implementation of published design methods straightforward.

Runtimes of design methods are usually on the order of seconds to minutes. Nevertheless, scanning large numbers of potential products, host organisms, and feedstocks, can quickly make computations challenging. As described above, cameo makes unit operations as fast as possible by implementing an efficient interface to the underlying optimization software. In addition, a number of methods in cameo can be parallelized, and can thus take advantage of multicore CPUs and HPC infrastructure if available (see documentation).

With this broad overview of capabilities, we would like to emphasize the role of cameo as a useful resource to the modeling community and wish to support its development as a
community effort in the long run. The majority of published strain design algorithms have not been experimentally validated and we believe that their inaccessibility to users is a major factor for the lack of validation. With cameo we hope to counteract this problem by making these methods accessible to the entire metabolic engineering community and also providing a platform for modelers to implement and publish novel methods.

**CONCLUSIONS**

With cameo version 0.11.4 we release a tool that is ready to be used in metabolic engineering projects. It is under active development, and future work will include interfacing cameo with genome-editing tools to streamline the translation of computed strain designs into laboratory protocols, modeling of fermentation processes to get estimates on titers and productivities, and include pathway predictions based on retrobiosynthesis including hypothetical biochemical conversions.

**AUTHOR INFORMATION**

**Corresponding Author**
*E-mail: niso@biosustain.dtu.dk.*

**ORCID**
Nikolaus Sonnenschein: 0000-0002-7581-4936

**Notes**
The authors declare no competing financial interest.

**ACKNOWLEDGMENTS**

We would like to thank Kai Zhuang, Miguel Campodonico and Sumesh Sukumura for providing valuable feedback and bug reports as early users of cameo. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 686070. Furthermore, we acknowledge financial support from the Novo Nordisk Foundation.

**REFERENCES**


