SOFTWARE

moped - A Python package for metabolic modelling and topological analysis

Nima P. Saadat*, Marvin van Aalst and Oliver Ebenhöh

*Correspondence:
nima.saadat@hhu.de
Institute for Quantitative and Theoretical Biology, Heinrich Heine University, Universitätsstrasse 1, Düsseldorf, Germany
Full list of author information is available at the end of the article

Abstract

Background: Mathematical modeling of metabolic networks is a powerful approach to investigate the underlying principles of metabolism and growth. Such approaches include, amongst others, differential equations based modeling of metabolic systems, constraint based modeling and topological analysis of metabolic networks. Most of these methods are well established and are implemented in numerous software packages, but these are scattered between different programming languages, packages and syntaxes. This complicates establishing straight forward pipelines integrating model construction and simulation.

Results: We present the Python package `moped` which serves as an integrative hub for constructing, modifying and analysing metabolic models. `moped` supports the de novo construction of models directly from genome sequences and pathway/genome databases, providing a completely reproducible model construction and curation process. Alternatively, existing models published in SBML format can be easily imported. Models are represented as Python objects, for which a wide spectrum of easy-to-use modification and analysis methods exist. The model structure can be manually altered by adding, removing or modifying reactions, and gaps can be filled automatically. This greatly supports the development of curated models. Moreover, moped provides several analysis methods, in particular including the calculation of biosynthetic capacities using metabolic network expansion. The integration with other Python based tools is facilitated through various model export options. For example, a model can be directly converted into a cobrapy object for constraint-based analyses. Likewise, conversion into a modelbase object supports dynamic simulations using ordinary differential equations.

Conclusion: `moped` is a fully documented and expandable Python package. We demonstrate the capability to serve as a hub for integrating model construction, database import, topological analysis and export for constraint-based and kinetic analyses.

Keywords: Metabolic networks; Modelling; Topology; Metabolic network expansion; Network reconstruction

Background

Theoretical analysis of metabolic pathways has a long standing tradition. The early approaches to study glycolosis, for example, have considerably increased our understanding of fundamental regulatory principles in metabolism [1]. In recent approaches, metabolic modelling was employed to study metabolic interdependen-

cies in microbial communities, and to identify putative drug targets for microbial pathogens [2, 3].

Several theoretical techniques to study metabolism have been established. The most classic technique is the analysis of metabolic networks by representing them in systems of ordinary differential equations (ODEs). These systems heavily depend on detailed knowledge of stoichiometries, parameters of enzyme kinetics and regulatory mechanisms of reactions [4]. This approach is extremely useful for investigating relatively small systems. The upsurge of novel high-throughput experimental 'omics' techniques led to the collection of immense amounts of data, resulting in an ever increasing number of fully sequenced genomes. The improved quality of annotated genes resulted in a tremendous increase in information of enzymes and the respective metabolic reactions. This information has been collected in biochemical databases like MetaCyc, BioCyc, KEGG or BiGG [5, 6, 7, 8]. Such databases provide information for large-scale metabolic networks of many different organisms. However, analysing such large-scale metabolic networks using systems of ordinary differential equations is difficult. This is, to a large extent, due to missing information on kinetic parameters of the involved enzymatic reactions [9]. One convenient alternative is constraint-based modelling, and its mathematical method flux balance analysis (FBA) [10]. This commonly used approach uses the stoichiometric matrix of a reaction network and finds a vector of fluxes for all reactions in steady state that maximizes or minimizes an objective function that linearly depends on the reactions rates. Other structural analysis techniques focus on the topology of metabolic networks [11]. One such technique is metabolic network expansion. The metabolic scope describes the set of metabolites, which are topologically producible by a given network from an initial set of compounds [12]. Thus, metabolic network expansion allows to functionally characterize metabolic networks with respect to their biosynthetic capacities [13].

Topological techniques are extremely useful in the process of curating models, in particular to identify and add missing reactions [14]. This process, called gap filling, allows, for example, to complement draft metabolic networks in order to guarantee that observed compounds can be produced from the growth medium [15].

Many of the techniques described above have been implemented as Python packages. However, most of these software packages are not directly compatible with each other due to differences in data structure.

In this work, we present `moped`, a compact but very useful Python package that serves as a hub, offering tools for analysis, development and extension or modification of metabolic models. The integration of BLAST and pathway/genome databases such as MetaCyc and BioCyc into `moped` furthermore allows reconstructing metabolic network models directly from genome sequences [16], and ensuring that the reconstruction process is fully transparent and reproducible. In addition to the de novo construction of models, `moped` provides an interface to import existing metabolic network models in SBML format.

To facilitate curation of metabolic models, `moped` provides an interface to Meneco, a topological gap-filling tool based on answer set programming [17]. All models created with `moped` can easily be exported as CobraPy or modelbase objects, thus directly integrating constraint-based and kinetic modelling with model construction

and modification [18, 19]. The Python package `moped` presented here is the first mathematical modelling hub, which allows constructing metabolic models de novo, integrating existing models in SMBL format, curating models by gap-filling, and performing topological, constraint-based or kinetic analyses.

Implementation

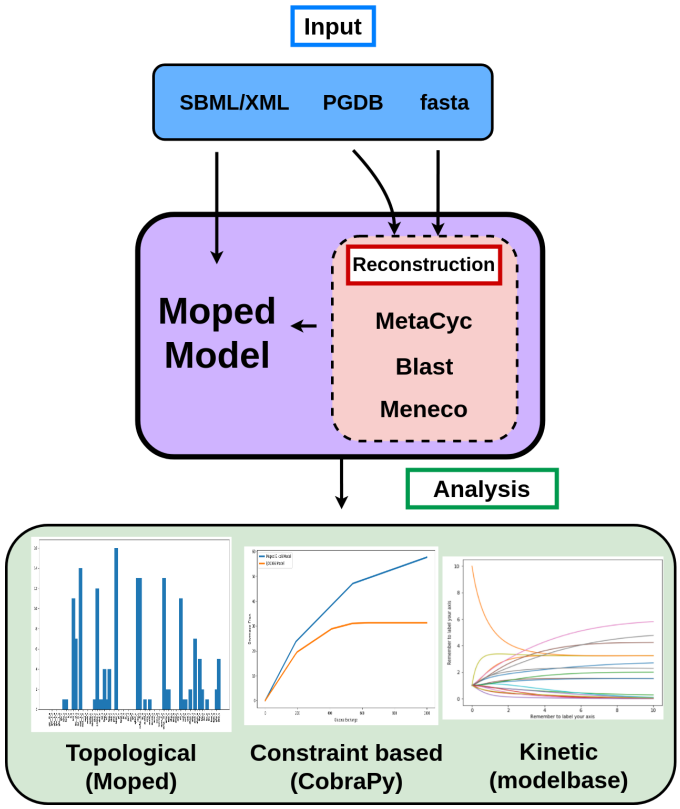


Figure 1: **The modelling hub `moped`.** `moped` accepts SBML, FASTA files or MetaCyc and BioCyc PGDBs as inputs. PGDBs and SBML files are directly converted into a `moped` object. By BLASTing genome-sequences against MetaCyc, `moped` models can be constructed. Further reconstruction can be achieved using Meneco for gap-filling. Topological model analysis is implemented in `moped`. For contraint based and kinetic analysis, `moped` offers export as CobraPy and `modelbase` objects, respectively [18, 19]

General Implementation and structure of Python object/classes

`moped` is a package fully integrated in the object-oriented programming language Python. The core is the `moped.model` class, which instantiates a metabolic network from scratch or from input files like SBML or pathway/genome database (PGDB) flat files. This class includes the subclasses `reactions` and `compounds`, which contain all extracted information for the respective attributes of the metabolic network, which is mostly stored in dictionary structures. The `moped.databases` class constructs the `moped.model` object by parsing PGDBs.

The `moped.analysis` class includes the subclass `moped.analysis.blast`, which constructs a `moped.model` from all reactions in the MetaCyc database that can be found within a genome sequence using BLAST. For this, `moped` requires FASTA files as input, and parameters for BLAST can be specified.

The subclass `moped.analysis.gap-filling` allows gap-filling via Meneco using one `moped.model` object as the draft network and another one as the repair network. The last subclass, `model.analysis.scope` provides topological analysis of `moped.model` objects using metabolic network expansion. The core classes are displayed in an unified modelling language graph in Figure 2.

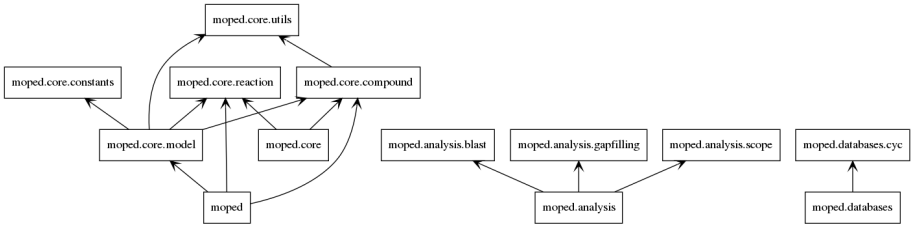


Figure 2: **Core classes in `moped`.** Details of object attributes and methods can be found in the documentation.

Model import, extension and modification

`moped` uses SBML files or PGDB flat files as input for constructing a metabolic network model. PGDBs are organism specific pathway/genome databases containing predicted reactions and compounds of the metabolism of the organism [6]. These databases further include detailed information about reactions and compounds, such as sum formulas, charges, references to other database entries or subcellular localisation. This information is of great importance for a consistent analysis of metabolic networks. SBML files represent metabolic networks in an XML-based format and can be considered as a standard for the exchange of reconstructed and curated metabolic models between tools and platforms [20]. Such files can be, amongst others, obtained from databases like BiGG, which provides SBML files of curated metabolic models together with information about the corresponding publications [7].

Because of the wide range of import methods (FASTA, PGDBs and SBML), one particular strength of `moped` is the integration of several analysis tools. Furthermore, `moped` provides a very accessible environment to extend or modify constructed or imported models. Therefore, adding alternative or additional metabolic pathways to pre-existing models, as well as modifying compound and reaction identifiers, is simple and straight forward. Naturally, all `moped` objects can be exported as SBML.

Metabolic network expansion and tools for topological analysis

A useful and valuable feature of moped is the fully implemented network expansion algorithm to perform topological analyses on `moped` objects. Metabolic network expansion can be used to investigate structural properties of metabolic networks, such as biosynthetic capacities and their robustness against structural perturbations [12]. The core concept of metabolic network expansion is the metabolic scope,

which contains all compounds that are producible by a network from a given initial set of compounds, termed the seed (see Figure 3). In the expansion process, the seed is used to find all reactions that can proceed in their annotated direction. The respective products are then added to the seed, forming the new seed for the next expansion step. This process continues until no new compounds are added to the seed. Thus, scopes characterise biosynthetic capacities of metabolic networks, based exclusively on their topology.

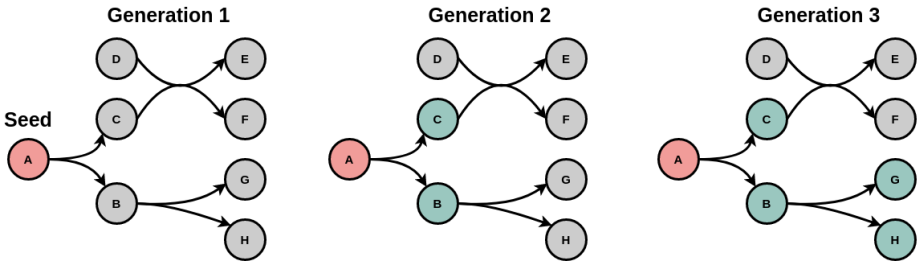


Figure 3: **Metabolic Network Expansion** Beginning with an initial set of compounds, the seed (here **A**), the expansion process detects all producible compounds in a network and adds them to the seed for the next generation until no additional producible compounds are found.

Any topological analysis based on network expansion depends on a precise definition of reaction reversibilities and involved cofactors. Network expansion uses the stoichiometry of reactions to identify producible compounds. However, stoichiometric coefficients of reactions are annotated for one particular direction. To include the opposite direction (for reversible reactions) into the topological analysis, `moped` provides a method for reversibility duplication. As illustrated in Figure 4 for Triose-phosphoisomerase as an example, this method finds all reversible reactions in a `moped` object and adds the reversed reaction to the network. The new reaction identifier is identical to the identifier of the original reaction concatenated with the suffix '__rev__'. This model modification can be reverted, if no longer needed.

Many reactions depend on specific cofactors. Cofactors usually appear in pairs. One of the most prominent examples is the cofactor pair ATP and ADP. In the majority of reactions with ATP as substrate, ATP serves as a donor of a phosphate group, thus producing ADP. Only a few reactions actually modify the adenosine moiety (for example in nucleotide de novo synthesis). In network expansion, therefore, no reaction utilising ATP or ADP as cofactor could proceed, unless these compounds are either included in the seed or can be produced from metabolites within the seed. If the purpose of network expansion is to realistically calculate a set of producible compounds, this behaviour is not desired, because it leads to a drastic underestimation of the scope. The most naive approach to directly include cofactors in the seed yields misleading results, because in such a case all compounds that can be generated from digesting, e.g., ATP would be included in the scope.

A pragmatic approach to solve this problem is the duplication of cofactors as proposed in [12]. Here, reactions with cofactor pairs are duplicated, where the copied reactions contain "mock cofactors". In contrast to the real cofactors, the mock

cofactors only occur in reactions, in which they act in their role as cofactors. For ATP, this is the transfer of a phosphate group, for NADH or NADPH the transfer of protons and electrons, and for acetyl coenzyme-A, the transfer of the acetyl group. The cofactor duplication allows the use of mock cofactors inside the initial seed. Reactions depending on cofactors might now be able to occur in the expansion process. However, reactions using the cofactors as proper substrates can only occur if the real cofactor can be produced from the seed.

`moped` provides a convenient method for finding and duplicating all reactions using cofactor pairs. The cofactor pairs can either be automatically determined by `moped` for networks imported from BiGG or MetaCyc, or they can be declared individually by the user. The identifiers of the duplicated cofactors are replaced by mock identifiers, which contain the suffix '__cof__'. The same modification is applied to the respective reaction identifiers. This model modification can be reverted if no longer needed.

The implemented methods for cofactor and reversibility duplication are commonly used to obtain biologically meaningful results for metabolic network expansion. However, they are also highly useful for topological gap-filling using Meneco, during model reconstruction. This is further explained in the next section.

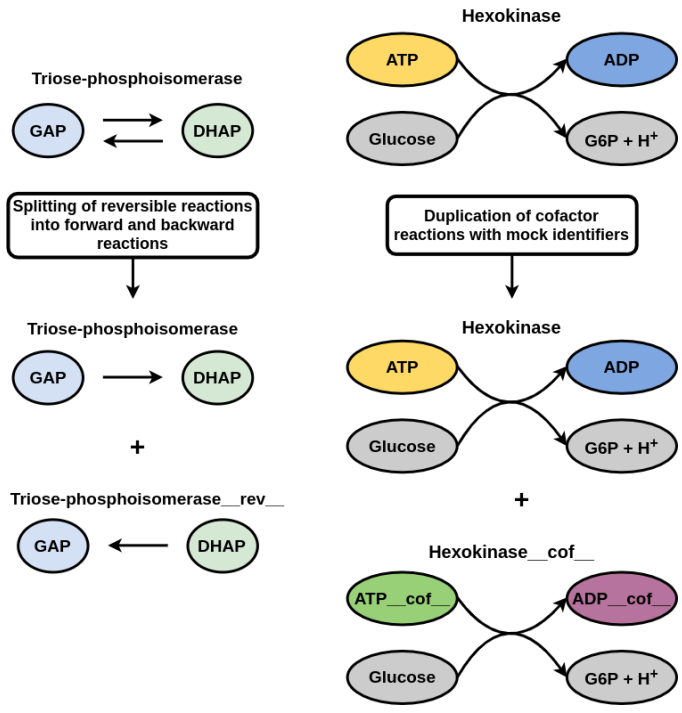


Figure 4: **Topological network modifications** `moped` offers functions for splitting reversible reactions into forward and backward reactions in a network. Adding a copy of each cofactor dependent reaction and replacing cofactors (here ATP and ADP) with mock identifiers allows unblocking cofactor dependent reactions while avoiding degradation products of cofactors contained in the seed. Such modifications enable biologically feasible topological analysis.

Reconstruction of draft network models

Construction of metabolic networks highly depends on reliable databases. In order to enable user-friendly metabolic network reconstruction, `moped` includes methods for importing data from the MetaCyc and BioCyc databases, identifying homologous sets of genes with BLAST and gap-filling.

MetaCyc is a universal, highly curated reference database of metabolic pathways and biochemical reactions from all domains of life. BioCyc is a database of organism specific PGDBs containing metabolic network information based on predictions by the PathoLogic component of the Pathway Tools software [21, 22]. The MetaCyc and BioCyc databases provide many advantages. Both databases are freely available for academic and nonprofit users. All PGDBs are available in useful flat file formats.

In order to use BioCyc and MetaCyc for metabolic network construction and analysis, `moped` offers a parser for PGDBs, allowing direct construction of `moped` objects from MetaCyc or BioCyc flat files. `moped` objects can directly be used for network analyses including network expansion and constraint-based modelling. Especially for the latter, it is extremely important that all reactions are mass- and charge-balanced to ensure that all solutions obey mass conversation. Therefore, only reactions which are mass- and charge-balanced are parsed in `moped`. This pipeline provided by the database import and parsing of `moped` makes it straight forward to construct prokaryotic network models. For eukaryotic metabolic networks, however, intensive and careful curation is required due to missing compartment information. More detailed information about the parsing of PGDBs using `moped` can be found in the documentation.

There exist several pipelines to automatically extract a set of metabolic reactions from a genome sequence. One popular pipeline is the above mentioned PathoLogic software. `moped` integrates such a pipeline into the Python programming language, directly converting a genome sequence into a `moped` object which can be immediately used for modelling applications. This functionality is provided by an implemented wrapper for BLAST to find enzyme reactions in genome sequences by similarity search against enzyme reference sequences from the MetaCyc database. This methods constructs a new `moped` object representing a metabolic network of all reactions that are found in a genome sequence. This process can be controlled by user defined thresholds. This integrated pipeline makes the model reconstruction perfectly reproducible, and illustrates the functionality of `moped` as a modelling hub.

The next curation step after the initial automatic network construction is usually gap-filling. This describes a process in which reactions are added to the network in order to ensure that the reconstructed model reflects experimentally observed behaviour, such as the production of experimentally measured compounds from the growth medium [23]. There are many available gap-filling methods like GapFill or MIRAGE [24, 25].

Most of these methods are based on constraint-based approaches. A common problem is that these approaches predict gap-filling solutions which use thermodynamically infeasible cycles. In this sense, these approaches are sensitive to self-producing or energy generating cycles. Meneco, in contrast, is a topological gap-filling tool based on the network expansion algorithm. Meneco calculates a minimal set of reactions that need to be added to a draft network such that a specified list of target

compounds can be produced from a given set of seed compounds. This gap-filling approach offers the advantage that it is inherently impossible for gap-filling solutions to depend on infeasible cycles. Meneco gap-filling can be directly applied as a method to `moped` objects. One `moped` object represents the draft network and a second the repair network, from which the added reactions are chosen.

The topological network modifications, i.e. reversibility and cofactor duplication, harmonize ideally with the application of Meneco, resulting in networks with biologically realistic behaviour. This again illustrates the integrative nature of the modelling hub `moped`.

By integrating database import, topological modification and gap-filling, `moped` represents a unifying, easy-to-use package for the initial reconstruction of draft genome scale metabolic networks of high quality. Naturally, all resulting draft models always require additional manual curation.

Results

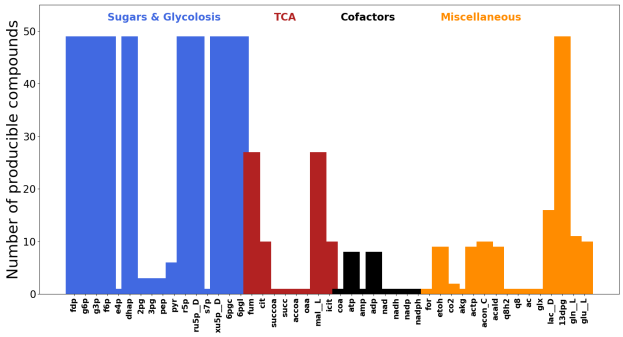
Applying metabolic network expansion to a model of *E. coli* core metabolism

We illustrate `moped`'s topological analysis functions by applying the metabolic network expansion algorithm to a compact network of *E. coli* core metabolism, which is freely available in SBML format from the BiGG database [27]. After importing the SBML file into moped, we applied cofactor and reversibility duplications as described above.

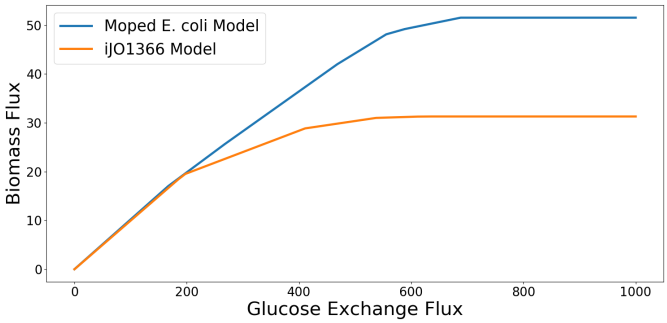
For each metabolite in the network, we calculate the scope size, i.e. how many new compounds are producible if only this metabolite, water and a set of mock cofactors are available. The results of that analysis are displayed in Figure 5a. In this relatively small metabolic network (72 metabolites and 95 reactions), eleven key compounds, which are mostly part of central metabolism, exhibit a largest observed scope size of 47. Such detailed topological analysis is useful to provide insight about central metabolites, as well as structural and functional characteristics of metabolic networks [13]. Whereas we here only display the scope size, the methods implemented in `moped` allow a far wider spectrum of analysis methods, including determination the set of producible metabolites, as well as following each step of the expansion process. The code used to produce the results and Figure 5a can be found on gitlab.

Integration of `moped` with CobraPy for Flux Balance Analysis

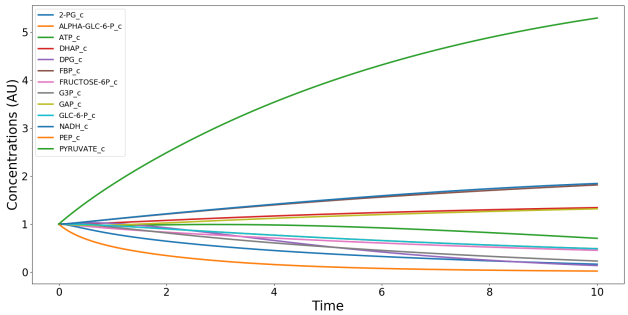
We demonstrate how moped provides a complete and easy-to-use pipeline to construct genome scale models from genome sequences and directly apply these models for constraint-based analyses. For this, we download the freely available MetaCyc PGDB and a FASTA file of the complete genome sequence of *Escherichia coli* str. K-12 substr. MG1655 [28]. We imported the MetaCyc PGDB to construct a `moped` object of the MetaCyc database as a reference network. Applying the BLAST wrapper, which was described above, to the FASTA file and the reference network, we obtained a `moped` object with 4001 reactions and 1446 compounds. Then we applied gap-filling to ensure that the reconstructed model can produce all basic biomass compounds (all nucleic acids, amino acids and lipid precursors) from a minimal



(a) Metabolic scopes of all compounds in the *E. coli* core metabolic model calculated using `moped`. The Y-axis indicates the total amount of compounds producible from every compound, water and a set of acceptor mock cofactors.



(b) Biomass production fluxes over different glucose exchange rates of the curated model iJO1366 and a draft metabolic model constructed with `moped`. The FBA simulations have been performed using CobraPy.



(c) Metabolite concentrations over time (both in arbitrary units) of a draft kinetic model of glycolosis using `moped`. The initial concentration of α-glucose is 10, of all other metabolites 1.

Figure 5: **Examples illustrating the usage of `moped` as a modelling hub** The shown examples display the utility of `moped` for different theoretical approaches to model metabolic networks. Here a model of *E. coli* was constructed from genome sequences and databases [26, 6, 5]. This model was used to perform topological analysis, constraint based optimization and kinetic analysis.

glucose medium, represented by a seed containing glucose, ammonia, water, protons, oxygen, phosphate and sulfate, as well as a set of acceptor mock cofactors including ADP, NAD^+ and $NADP^+$. This gap-filling step resulted in a set of 11 reactions that needed to be added to the draft network. To this resulting network we added exchange reactions for all seed compounds. This `moped` object was then exported as a CobraPy object. With CobraPy, we performed a loopless FBA with the biomass reactions as the objective function to maximize. As expected, CobraPy returns a feasible solutions.

In order to test the quality of our draft model, we compared it with the established model iJO1366, a curated genome scale metabolic model of *Escherichia coli* str. K-12 substr. MG1655 [26]. For this, we performed loopless FBA on both models, systematically changing the bounds of glucose import. Figure 5b displays the calculated flux of biomass production for different glucose import fluxes of both models. Remarkably, both models behave rather similarly, at least from a qualitative perspective. With every point where the slope decreases, a new constraint becomes limiting in both models. The quantitative differences can be explained by various factors. First of all, the moped derived model is an automatically constructed draft network and iJO1366 is a manually curated published model widely used by the scientific community. In contrast to iJO1366, the moped derived model only contains default values for lower and upper bounds of fluxes. Moreover, the biomass is defined slightly differently in both models. Still, the general agreement of the model results demonstrates that it is easily possible to construct draft genome scale metabolic models using `moped` and the MetaCyc database, and directly use these models for constraint-based analyses in CobraPy. Naturally, draft metabolic models constructed via `moped`, like any other automatically derived model, need to be further curated. Thus, `moped` offers a console based and object oriented alternative, fully integrated into the Python programming language, to other reconstruction software packages [29, 30, 31, 32]. The code used to produce the results and Figure 5b can be found here.

Export of reactions in `moped` for kinetic modelling with `modelbase`

The most detailed method for investigating the dynamics of metabolic pathways is by kinetic models based on differential equations. Because models based on ODEs provide insight on the dynamics and regulation, moped provides integrated methods for the construction of ODE-based models from stoichiometric models. The construction of such models, however, requires detailed knowledge of all enzymatic parameters, such as Michaelis constants and catalytic rates. Therefore, an automatic construction of kinetic models from a stoichiometric model is extremely challenging. Nevertheless, `moped` provides methods for constructing backbone models for kinetic analysis. For this, a set of reactions from a `moped` network can be exported into a `modelbase` model. `modelbase` is a Python package specifically designed for construction of kinetic models [19]. The unique feature of modelbase is that model construction and analysis are fully integrated into the Python programming language. Moreover, modelbase is particularly designed to make model modification a straight forward exercise. The direct export of a `moped` object into modelbase can, of course, not be expected to result in a model that displays realistic behaviour.

Rather, it presents a scaffold of a model with the correct stoichiometry where all parameters and all rate laws are represented by default place-holder values and functions. These values and functions can then be modified with expert knowledge to result in a more realistic model. For this task, the modelbase package is perfectly suited because it provides methods for exporting the source code required to generate the model, which can subsequently be modified.

To illustrate this functionality, we again imported the MetaCyc database into a `moped` object. We then constructed a new `moped` object containing only those reactions, which are annotated as part of the glycolytic pathway. This model was directly exported into a kinetic modelbase model. The place-holder rate equations for all reactions are simple reversible mass-action kinetics with all kinetic parameters set to 1 (arbitrary units, AU). Although these parameters are certainly unrealistic, we performed a dynamic simulation to demonstrate the principle functionality of the model. For this, we set the initial concentration of the substrate Glucose-6-Phosphate to 5 (AU) and the initial concentration of all other metabolites to 1 (AU). The results are displayed in Figure 5c. Because the automatically constructed model is a closed system without in- or effluxes, the system obviously relaxes to an equilibrium state. As stressed above, the kinetic model is not intended for realistic simulations, but as a basis for developing a more realistic model. The code used to produce the results and figure can be found here.

Further improvements of the integration of `moped` with kinetic modelling are already planned. For example, the MetaCyc database contains kinetic parameters (k_{cat} and K_m) for some enzymatic reactions. It is envisaged that these parameters are parsed and automatically integrated into the kinetic models. Similarly, equilibrium constants can in principle be computationally determined by the group contribution method, and by integrating the Equilibrator tool, these constants can be automatically included in the constructed kinetic model [33, 34].

Conclusion

Here, we present `moped`, a Python package representing a hub connecting the construction, modification and curation of genome scale metabolic networks with various analysis methods, which support studies of metabolic networks. `moped` supports the de novo construction of metabolic networks by importing databases, providing homology searches and integrating an established gap-filling routine. Existing models from several sources can be imported using the standardized format SBML. Metabolic network models are represented as `moped` objects, which can be modified by easy-to-use and intuitive methods. `moped` models can be exported into various formats, thus integrating a diverse set of established analysis tools. Topological analysis, constraint-based optimization or dynamic analysis can be easily performed for any model represented as a `moped` object. The modular architecture of the open source package `moped` is particularly designed for allowing further extensions to enhance its functionality, such as the integration of additional software tools. We provide an extensive documentation for `moped`, as well as troubleshooting guides, unit-tests for all provided methods and example notebooks illustrating the usage of `moped` at `https://gitlab.com/marvin.vanaalst/moped`.

Availability and requirements
Project Name: Moped
Project home page: https://gitlab.com/marvin.vanaalst/moped
Operating systems: Linux, OS X
Programming language: Python
License: GPLv3
Any restrictions to use by non-academics: For non-profit use only

Abbreviations
PGDB - Pathway/Genome Database
FBA - Flux Balance Analysis
ODE - Ordinary Differential Equations

Funding
Funded by the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy EXC 2048/1, Project ID: 390686111

Availability of data and materials
All source code including scripts to produce all manuscript figures can be found at https://gitlab.com/marvin.vanaalst/moped

Competing interests
The authors declare that they have no competing interests.

Author's contributions
conceptualization, N.S.; implementation, N.S., M.v.A.; writing–original draft preparation, N.S.; writing–review and editing, N.S., M.v.A., and O.E.

Acknowledgements
We thank Anna Matuszyńska and Ovidiu Popa for critically reading our manuscript draft, Clémence Frioux for providing support for Meneco integration, Dipali Singh for giving valuable advice on MetaCyc and PDGB issues.

References
1. Rapoport, T.A., Heinrich, R., Jacobasch, G., Rapoport, S.: A linear steady-state treatment of enzymatic chains. a mathematical model of glycolysis of human erythrocytes. Eur J Biochem **42**(1), 107–120 (1974)
2. Zomorrodi, A.R., Segrè, D.: Genome-driven evolutionary game theory helps understand the rise of metabolic interdependencies in microbial communities. Nature communications **8**(1), 1–12 (2017)
3. Hartman, H.B., Fell, D.A., Rossell, S., Jensen, P.R., Woodward, M.J., Thorndahl, L., Jelsbak, L., Olsen, J.E., Raghunathan, A., Daefler, S., *et al.*: Identification of potential drug targets in salmonella enterica sv. typhimurium using metabolic modelling and experimental validation. Microbiology **160**(6), 1252–1266 (2014)
4. Heinrich, R., Schuster, S.: The Regulation of Cellular Systems. Chapman and Hall, New York (1996)
5. Caspi, R., Billington, R., Keseler, I.M., Kothari, A., Krummenacker, M., Midford, P.E., Ong, W.K., Paley, S., Subhraveti, P., Karp, P.D.: The metacyc database of metabolic pathways and enzymes-a 2019 update. Nucleic acids research **48**(D1), 445–453 (2020)
6. Karp, P.D., Billington, R., Caspi, R., Fulcher, C.A., Latendresse, M., Kothari, A., Keseler, I.M., Krummenacker, M., Midford, P.E., Ong, Q., *et al.*: The biocyc collection of microbial genomes and metabolic pathways. Briefings in bioinformatics **20**(4), 1085–1093 (2019)
7. King, Z.A., Lu, J., Dräger, A., Miller, P., Federowicz, S., Lerman, J.A., Ebrahim, A., Palsson, B.O., Lewis, N.E.: Bigg models: A platform for integrating, standardizing and sharing genome-scale models. Nucleic acids research **44**(D1), 515–522 (2016)
8. Kanehisa, M., Goto, S.: Kegg: kyoto encyclopedia of genes and genomes. Nucleic acids research **28**(1), 27–30 (2000)
9. Raman, K., Chandra, N.: Flux balance analysis of biological systems: applications and challenges. Briefings in bioinformatics **10**(4), 435–449 (2009)
10. Orth, J.D., Thiele, I., Palsson, B.Ø.: What is flux balance analysis? Nature biotechnology **28**(3), 245–248 (2010)
11. Wunderlich, Z., Mirny, L.A.: Using the topology of metabolic networks to predict viability of mutant strains. Biophysical journal **91**(6), 2304–2311 (2006)
12. Handorf, T., Ebenhöh, O., Heinrich, R.: Expanding metabolic networks: scopes of compounds, robustness, and evolution. Journal of molecular evolution **61**(4), 498–512 (2005)
13. Ebenhöh, O., Handorf, T.: Functional classification of genome-scale metabolic networks. EURASIP J Bioinform Syst Biol, 570456 (2009). doi:10.1155/2009/570456
14. Christian, N., May, P., Kempa, S., Handorf, T., Ebenhöh, O.: An integrative approach towards completing genome-scale metabolic networks. Mol Biosyst **5**(12), 1889–1903 (2009). doi:10.1039/B915913b
15. Orth, J.D., Palsson, B.Ø.: Systematizing the generation of missing metabolic knowledge. Biotechnology and bioengineering **107**(3), 403–412 (2010)
16. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. Journal of molecular biology **215**(3), 403–410 (1990)
17. Prigent, S., Frioux, C., Dittami, S.M., Thiele, S., Larhlimi, A., Collet, G., Gutknecht, F., Got, J., Eveillard, D., Bourdon, J., *et al.*: Meneco, a topology-based gap-filling tool applicable to degraded genome-wide metabolic networks. PLoS computational biology **13**(1), 1005276 (2017)
18. Ebrahim, A., Lerman, J.A., Palsson, B.O., Hyduke, D.R.: Cobrapy: Constraints-based reconstruction and analysis for python. BMC systems biology **7**(1), 74 (2013)

19. Ebenhöh, O., van Aalst, M., Saadat, N.P., Nies, T., Matuszyńska, A.: Building mathematical models of biological systems with modelbase. Journal of Open Research Software **6**(1) (2018)
20. Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., Bray, D., Cornish-Bowden, A., *et al.*: The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. Bioinformatics **19**(4), 524–531 (2003)
21. Karp, P.D., Paley, S., Romero, P.: The pathway tools software. Bioinformatics **18**(suppl_1), 225–232 (2002)
22. Karpe, P.D., Latendresse, M., Caspi, R.: The pathway tools pathway prediction algorithm. Standards in genomic sciences **5**(3), 424–429 (2011)
23. Thiele, I., Palsson, B.Ø.: A protocol for generating a high-quality genome-scale metabolic reconstruction. Nature protocols **5**(1), 93 (2010)
24. Kumar, V.S., Dasika, M.S., Maranas, C.D.: Optimization based automated curation of metabolic reconstructions. BMC bioinformatics **8**(1), 212 (2007)
25. Vitkin, E., Shlomi, T.: Mirage: a functional genomics-based approach for metabolic network model reconstruction and its application to cyanobacteria networks. Genome biology **13**(11), 111 (2012)
26. Orth, J.D., Conrad, T.M., Na, J., Lerman, J.A., Nam, H., Feist, A.M., Palsson, B.Ø.: A comprehensive genome-scale reconstruction of escherichia coli metabolism—2011. Molecular systems biology **7**(1) (2011)
27. Orth, J.D., Fleming, R.M., Palsson, B.O.: Reconstruction and use of microbial metabolic networks: the core escherichia coli metabolic model as an educational guide. EcoSal plus (2010)
28. Blattner, F.R., Plunkett, G., Bloch, C.A., Perna, N.T., Burland, V., Riley, M., Collado-Vides, J., Glasner, J.D., Rode, C.K., Mayhew, G.F., *et al.*: The complete genome sequence of escherichia coli k-12. science **277**(5331), 1453–1462 (1997)
29. Poolman, M.G.: Scrumpy: metabolic modelling with python. IEE Proceedings-Systems Biology **153**(5), 375–378 (2006)
30. Agren, R., Liu, L., Shoaie, S., Vongsangnak, W., Nookaew, I., Nielsen, J.: The raven toolbox and its use for generating a genome-scale metabolic model for penicillium chrysogenum. PLoS computational biology **9**(3) (2013)
31. Arkin, A.P., Cottingham, R.W., Henry, C.S., Harris, N.L., Stevens, R.L., Maslov, S., Dehal, P., Ware, D., Perez, F., Canon, S., *et al.*: Kbase: the united states department of energy systems biology knowledgebase. Nature biotechnology **36**(7), 566 (2018)
32. Latendresse, M., Krummenacker, M., Trupp, M., Karp, P.D.: Construction and completion of flux balance models from pathway databases. Bioinformatics **28**(3), 388–396 (2012)
33. Mavrovouniotis, M.L.: Estimation of standard gibbs energy changes of biotransformations. Journal of Biological Chemistry **266**(22), 14440–14445 (1991)
34. Flamholz, A., Noor, E., Bar-Even, A., Milo, R.: equilibrator—the biochemical thermodynamics calculator. Nucleic acids research **40**(D1), 770–775 (2012)