Synthetic and Systems Biotechnology 9 (2024) 494–502

Contents lists available at ScienceDirect

Synthetic and Systems Biotechnology

journal homepage: www.keaipublishing.com/en/journals/synthetic-and-systems-biotechnology

Original Research Article

ECMpy 2.0: A Python package for automated construction and analysis of enzyme-constrained models

Zhitao Mao [a,b,*], Jinhui Niu [a,b], Jianxiao Zhao [a,b,c], Yuanyuan Huang [a,b,d], Ke Wu [e], Liyuan Yun [f], Jirun Guan [a,b], Qianqian Yuan [a,b], Xiaoping Liao [a,b,g], Zhiwen Wang [c], Hongwu Ma [a,b,**]

[a] Biodesign Center, Key Laboratory of Engineering Biology for Low-carbon Manufacturing, Tianjin Institute of Industrial Biotechnology, Chinese Academy of Sciences, Tianjin, 300308, China
[b] National Center of Technology Innovation for Synthetic Biology, Tianjin, 300308, China
[c] Frontier Science Center for Synthetic Biology and Key Laboratory of Systems Bioengineering (Ministry of Education), Tianjin University, Tianjin, 300072, China
[d] College of Biotechnology, Tianjin University of Science and Technology, Tianjin, 300457, China
[e] Institute of Biopharmaceutical and Health Engineering, Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[f] Tianjin Agricultural College, Tianjin, 300384, China
[g] Haihe Laboratory of Synthetic Biology, Tianjin, 300308, China

A B S T R A C T

Genome-scale metabolic models (GEMs) have been widely employed to predict microorganism behaviors. However, GEMs only consider stoichiometric constraints, leading to a linear increase in simulated growth and product yields as substrate uptake rates rise. This divergence from experimental measurements prompted the creation of enzyme-constrained models (ecModels) for various species, successfully enhancing chemical production. Building upon studies that allocate macromolecule resources, we developed a Python-based workflow (ECMpy) that constructs an enzyme-constrained model. This involves directly imposing an enzyme amount constraint in GEM and accounting for protein subunit composition in reactions. However, this procedure demands manual collection of enzyme kinetic parameter information and subunit composition details, making it rather user-unfriendly. In this work, we've enhanced the ECMpy toolbox to version 2.0, broadening its scope to automatically generate ecGEMs for a wider array of organisms. ECMpy 2.0 automates the retrieval of enzyme kinetic parameters and employs machine learning for predicting these parameters, which significantly enhances parameter coverage. Additionally, ECMpy 2.0 introduces common analytical and visualization features for ecModels, rendering computational results more user accessible. Furthermore, ECMpy 2.0 seamlessly integrates three published algorithms that exploit ecModels to uncover potential targets for metabolic engineering. ECMpy 2.0 is available at https://github.com/tibbdc/ECMpy or as a pip package (https://pypi.org/project/ECMpy/).

1. Introduction

Genome-scale metabolic models (GEMs), a class of mathematical constructs, elucidate the intricate interplay among cellular genes, proteins, and reactions, effectively guiding and enhancing industrial chemical and biofuel production [1]. Current approaches for predicting microbial phenotypes and yields using GEMs primarily rely on conventional techniques like constraining carbon source uptake rates and adjusting the toggling of metabolic reactions. Unfortunately, these methods overlook the significant impact of enzyme concentrations, kinetic parameters, and pathway thermodynamics on reaction fluxes, resulting in substantial discrepancies between predictions and experimental results [2]. In an effort to overcome these limitations, the concept of enzymatic constraints on metabolic reactions has been skillfully integrated into various constraint-based methodologies. Notably, this integration is evident in frameworks such as GECKO (If no version is specified, the default is GECKO 3.0) [3] and AutoPACMEN [4]. These sophisticated models have significantly expanded the scope of classical flux balance analysis (FBA), providing insights into overflow metabolism and cellular growth across diverse environments for a range of organisms, including *Escherichia coli* [5,6], *Saccharomyces cerevisiae* [7], *Yarrowia lipolytica* [8], *Aspergillus niger* [9], *Corynebacterium*

Peer review under responsibility of KeAi Communications Co., Ltd.
* Corresponding author. Biodesign Center, Key Laboratory of Engineering Biology for Low-carbon Manufacturing, Tianjin Institute of Industrial Biotechnology, Chinese Academy of Sciences, Tianjin 300308, China.
** Corresponding author. Biodesign Center, Key Laboratory of Engineering Biology for Low-carbon Manufacturing, Tianjin Institute of Industrial Biotechnology, Chinese Academy of Sciences, Tianjin 300308, China.
E-mail addresses: mao_zt@tib.cas.cn (Z. Mao), ma_hw@tib.cas.cn (H. Ma).

https://doi.org/10.1016/j.synbio.2024.04.005
Received 22 January 2024; Received in revised form 13 March 2024; Accepted 7 April 2024
Available online 10 April 2024

glutamicum [10], and *Bacillus subtilis* [11].

ECMpy, a simplified Python-based workflow developed in 2021 [6], has been applied to construct ecGEMs for *E. coli* [6], *C. glutamicum* [10], and *B. subtilis* [11]. In contrast to the methodologies of GECKO and AutoPACMEN, ECMpy introduces constraints on the total enzyme amount without adding pseudo metabolites or reactions to the stoichiometric matrix (S-matrix). It also reveals and emphasizes the influence of protein subunit composition within reactions on model simulation outcomes, along with presenting an automated calibration approach for enzyme kinetic parameters. As ECMpy avoids modifying the foundational S-matrix, it seamlessly integrates multiple algorithms already incorporated into COBRApy [12], including flux variability analysis (FVA), FASTCC [13], and minimization of metabolic adjustment (MOMA). This integration is achieved without the need to rewrite algorithms based on new S-matrix features, unlike GECKO and AutoPACMEN. However, it's important to note that within ECMpy, the enzymatic kinetic parameters and protein subunit composition data required for model construction must be acquired and curated manually, resulting in a user experience that is notably user-unfriendly.

In this study, we have advanced the ECMpy toolbox to its enhanced version, ECMpy 2.0, thereby expanding its capabilities to automatically construct ecGEMs for a wider range of organisms. Specifically, the integration of AutoPACMEN and DLKcat [14] enhances ECMpy 2.0's capacity to provide proficient support for both model and non-model organisms, ensuring a comprehensive incorporation of kinetic constraints, even for organisms that have received limited attention. Additionally, an automated procedure has been established within the realm of protein molecular mass calculation to parse the UniProt database [15], providing precise protein subunit composition information and reinforcing the accuracy of corresponding reaction-associated protein molecular masses. To broaden the functionality of ECMpy 2.0, a comprehensive set of simulation utility functions has been seamlessly integrated, covering model parameter analysis [5,7–11], phenotypic phase plane (PhPP) analysis [9–11], metabolic overflow simulation [6,7,10,11], and trade-off phenomena simulation [6,10,11]. Finally, ECMpy 2.0 integrates three prominent algorithms, each notably effective in predicting target for metabolic engineering. These include an enzyme cost-based sorting method [5,11], the flux scanning based on enforced objective flux (FSEOF) method [16,17], and an approach based on protein cost discrepancies across reactions in distinct scenarios: high growth low product generation (HGLP) and low growth high product generation (LGHP) [10]. The enzyme cost-based sorting method, known for its simplicity and widespread application, has the capacity to reveal pivotal enzymes within a pathway. It has demonstrated success in various contexts, such as enhancing lysine production in *E. coli* [5] and synthesizing riboflavin and menaquinone 7 in *B. subtilis* [11]. However, this method excels primarily in identifying overexpressed targets, leading to the development of an approach rooted in the disparities of protein costs within reactions (HGLP and LGHP). This innovative methodology distinguishes overexpression and attenuation targets within a pathway and has been effectively applied in scenarios such as enhancing lysine production in *C. glutamicum* [10]. Furthermore, ECMpy 2.0 incorporates the FSEOF algorithm, an innovation introduced in 2010 and primarily designed for identifying targets within GEMs [17]. Recently, Ishchuk et al. refined and applied this algorithm to the yeast ecModel, ecYeast8. By predicting combinations of targets, they managed to amplify heme production within cells by an impressive 70-fold [16]. To the best of our knowledge, ECMpy 2.0 stands as a pioneering automated toolkit seamlessly integrating data acquisition, model construction, model refinement, model analysis and visualization, along with the prediction of metabolic engineering targets.

2. Materials and methods

2.1. Model parsing

ECMpy requires the retrieval of protein molecular weight information from UniProt using UniProt IDs, in conjunction with the allocation of enzyme kinetic data through substrate (BiGG ID) and EC matching. Recognizing the existing variability in the quality of GEMs, we have introduced a preliminary step in ecModel construction that involves evaluating gene coverage (UniProt ID coverage), reactions coverage (EC number coverage excluding exchange reactions), and metabolites coverage (BiGG ID coverage). During this assessment, we have established that models with coverage below 33 % are not recommended for direct ecModel construction. Users are required to augment these annotation details within the model to proceed with subsequent processes.

2.2. Parameter acquisition

The total enzyme amount constraint ($\sum_i^n \frac{v_i * MWi}{\sigma_i * k_{cat,i}} \leq P_{total} * f$) is primarily composed of the following elements: molecular weight (MW), saturation coefficient (σ), enzyme kinetic parameters (k_{cat}), the total protein fraction (P_{total}), and the mass fraction of enzymes (f). Notably, the saturation coefficient typically adopts a uniform average of 0.5 across all enzymes [7,10,11], while P_{total} of 0.56 g total cellular protein/g DCW represents the average protein content in most microbial cells [18]. The f value can be derived from protein composition data, such as for *E. coli* with 0.406 g enzyme/g total cellular protein [6], *C. glutamicum* with 0.46 g enzyme/g total cellular protein [10], *B. subtilis* with 0.588 g enzyme/g total cellular protein [11], and *S. cerevisiae* with 0.446 g enzyme/g total cellular protein [7]. In ECMpy 2.0, we furnish a function to compute the f value (see 02.get_ecModel_using_ECMpy.ipynb), whereby users need only provide GEM files and protein abundance data to automate the acquisition of f values.

ECMpy 2.0 offers two approaches for accessing enzyme kinetic parameter information, ensuring comprehensive coverage for both model organisms and non-model organisms. The first method utilizes the AutoPACMEN process to extract enzyme kinetic parameter information from the BRENDA [19] and SABIO-RK [20] databases. Using substrate and EC number, the enzyme kinetic parameter information associated with the model is parsed in 9 steps (detailed in 01.get_reaction_kcat_using_AutoPACMEN.ipynb). If a perfect match for organism, substrate, and EC number is not found, AutoPACMEN facilitates k_{cat} matches by querying the BRENDA and SABIO-RK databases using a set of hierarchical matching criteria as summarized in the literature [4]. Furthermore, to augment the number of experimental sources for k_{cat} data, we extended the AutoPACMEN workflow in ECMpy 2.0, introducing a transformation for enzyme specific activity data (Eq. (1)). This method is suitable for cases with high annotation quality in the species' GEM and a database rich in enzyme kinetic parameter data. For non-model organisms, there is a scarcity of enzyme kinetic parameter data in the BRENDA and SABIO-RK databases, which is insufficient for constructing ecModels. The second method integrates the DLKcat [14] approach, which predicts enzyme kinetic parameters based on the sequence information of enzymes catalyzing reactions and substrate information. This method involves seven steps outlined in 01.get_reactiion_kcat_using_DLKcat.ipynb. While this approach can provide kinetic parameter information for nearly all enzymes in the model, there is a discrepancy between predicted values and experimental data. It is particularly useful for cases with poor GEM annotation quality and limited database coverage of kinetic parameters for that species.

$$k_{cat}\left[s^{-1}\right] = \frac{SA\left[umol * mg^{-1} * min^{-1}\right] * MW\left[mg * umol^{-1}\right]}{60\left[s * min^{-1}\right]} \quad (1)$$

Molecular weight (MW) is a critical factor influencing the predictive accuracy of ecModels. Two key factors impact the final MW assigned to a specific reaction's enzyme: whether the protein is constituted by subunits ('and' relationship in the GPR) and the quantity of each subunit. Previous ecModel construction workflows, such as GECKO and AutoPACMEN, have overlooked subunit composition information (typically set to 1 or provided by the user). In ECMpy 2.0, we have introduced an automated process for acquiring protein subunit composition, streamlining the process for users who only need to provide species information to retrieve quantitative subunit information extracted from the 'Interaction' section in UniProt. Initially, we retrieve all UniProt IDs and 'Interaction' section for the specified species based on species name or ID. Subsequently, we developed a word list to analyze the descriptive information within the 'Interaction' section (e.g., Homodimer; Heterotrimer; Tetramer of two alpha and two beta chains) and converted it into corresponding subunit numbers [10]. For instance, P0A796 is described in UniProt as 'Homotetramer', indicating a subunit number of 4.

2.3. ecModel construction

The process of constructing an ecModel can be outlined through the following steps (as detailed in 02.get_ecModel_using_ECMpy.ipynb). Initially, reversible reactions in the GEM were divided into pairs of irreversible reactions. Additionally, reactions governed by multiple isoenzymes were segmented into distinct reactions, appending a numerical identifier to the reaction ID, such as GLCpts_num1. This approach ensures that each reaction is uniquely associated with a corresponding enzyme entity. Following this clarification, we calculated the MW for each individual enzyme entity. For reactions catalyzed by enzyme complexes, we aggregated the cumulative sum of proteins constituting the complex (as in Eq. (6)). Furthermore, we proceeded to determine the kinetic parameters for the enzymes, a task accomplished through the AutoPACMEN or DLKcat methodologies. Concurrently, we quantified the mass fraction of total cellular enzymes (Eq. (7)), a process facilitated by proteomic or RNA-Seq data. Ultimately, the ECMpy process was used to construct the ecGEM, and the mathematical representation of this method is as follows:

$$Z = \max\left\{C^T * v\right\} \quad (2)$$

$$S * v = 0 \quad (3)$$

$$lb \le v \le ub \quad (4)$$

$$\sum_{i=1}^{n} \frac{v_i * MW_i}{\sigma_i * k_{cat,i}} \le P_{total} * f \quad (5)$$

$$\mathrm{MW} = \sum_{j=1}^{m} N_j * MW_j \quad (6)$$

$$f = \sum_{i=1}^{n(gene_{model})} A_i MW_i \Big/ \sum_{j=1}^{n(gene_{total})} A_j MW_j \quad (7)$$

Where C^T is the transposed vector of the integer coefficient of each flux in the objective function Z; S is the stoichiometric matrix; lb and ub are the lower and upper bounds of the reaction fluxes in the system, respectively; $k_{cat,i}$ is the turnover number of enzymes that catalyze reaction i; MW_i denotes the molecular weight of enzyme i; m is the number of different subunits in the enzyme complex; N_j is the number of jth subunits in the complex; σ_i is the saturation coefficient for enzyme i; P_{total} is the average protein content in most microbial cells ; f is the total mass fraction of all cellular enzymes in our ecGEM.

2.4. Parameter calibration

The initial ecGEM exhibited discrepancies in predicting the experimental phenotype, necessitating a correction of the k_{cat} values. In the context of ECMpy 2.0, we have implemented an automated parameter calibration process for the ecModel, by identifying the reaction with the highest enzyme cost in the pathway and substituting its k_{cat} values with the highest value (as detailed in 03.ecModel_calibration.ipynb). The calibration procedure aligns with the coherent approaches of ECMpy and GECKO, with a comprehensive blueprint elaborated in the ecBSU1 [11]. Initially, our effort involved quantifying the enzyme cost attributed to each specific reaction in the pathway, focusing on biomass maximization as the primary objective. Subsequently, reactions exhibiting the most substantial enzyme costs were identified as candidates for recalibration. Following this identification, adjustments were made to the reaction's k_{cat} value, aligning it with the highest corresponding k_{cat} value extracted from the BRENDA and SABIO-RK databases. In determining the highest corresponding k_{cat} value, the current process does not distinguish between different experimental conditions or species. Instead, it relies solely on identifying the maximum value associated with the same EC number in the two databases. This iterative refinement process continued until convergence with experimental data was achieved or the specified threshold of iterations was reached.

2.5. ecModel analysis and visualization

ECMpy 2.0 introduces a suite of analytical functionalities, complemented by visual representations, to comprehensively dissect ecModels. For instance, we have incorporated cumulative distribution plots to showcase the complexity of ecModel parameters, spanning dimensions such as k_{cat} and MW. Additionally, we utilize Phenotypic Phase Plane (PhPP) analysis to reveal trends in the model solution space. Various analyses related to metabolic overflow are integrated, enabling the effortless capture of overflow metabolites and presenting the reasons behind overflow metabolism occurrences. Lastly, simulations are provided to illustrate the trade-off between enzyme usage efficiency and biomass yield.

2.6. Metabolic engineering target prediction

Within the context of ECMpy 2.0, we offer three distinct methodologies for predicting metabolic engineering targets using ecModels. These methodologies include the enzyme cost-based sorting method [5,11], the FSEOF algorithm-based method [16], and the HGLP/LGHP method [10]. The enzyme cost-based sorting method, utilized for product computation, focuses on the product, setting the lower threshold of the biomass reaction at 10 % of the maximal growth rate. By analyzing flux values, it examines the enzyme cost of each reaction, pinpointing kinetic bottleneck reactions characterized by the most substantial enzyme costs (Eq. (8)). The HGLP/LGHP method calculates the cost of each reaction within two pathways, namely HGLP and LGHP. It subsequently scrutinizes fold changes in enzyme costs, selecting those with a fold change exceeding 1.5 as potential targets for metabolic engineering (Eqs. (9) and (10)). For the FSEOF method, we have successfully reproduced and translated this methodology from MATLAB code to Python. During each simulation, the core objective was to maximize

product production, a pursuit undertaken across ten distinct biomass-yield conditions (0.5* wild-type biomass - 0.9* wild-type biomass). Scores were generated for each reaction in the network, discerning fluxes that consistently increased or decreased as biomass requirements diminished. Thus, the flux scores meticulously highlighted reactions that exhibited consistent upregulation (score >1), down-regulation (0 < score <0.95), or stability (0.95 < score <1).

$$Enzyme\ cost_i = \frac{v_i * MW_i}{\sigma_i * k_{cat,i}} \tag{8}$$

$$Enhance\ target = \left\{ Enzyme \middle| \frac{Enzyme\ cost_{LGHP}}{Enzyme\ cost_{HGLP}} \geq 1.5 \right\} \tag{9}$$

$$Weaken\ target = \left\{ Enzyme \middle| \frac{Enzyme\ cost_{HGLP}}{Enzyme\ cost_{LGHP}} \geq 1.5 \right\} \tag{10}$$

2.7. Configuration of operating environment

ECMpy 2.0 is implemented in Python 3.7 and utilizes Python's standardized library along with additional modules such as Biopython, COBRApy (version = 0.21.0) [12], openpyxl, requests, xlsxwriter, scikit-learn [21], RDKit, pubchempy, plotly, among others, as detailed on the project's GitHub page (https://github.com/tibbdc/ECMpy). Users can access the ECMpy 2.0 codebase and documentation via two methods:

(1) Installing ECMpy 2.0 using pip:

pip install ECMpy.

(2) Acquiring ECMpy 2.0 via git clone:

git clone https://github.com/tibbdc/ECMpy.git.

Furthermore, detailed documentation on the functionalities of all ECMpy 2.0 functions is available at https://ecmpy.readthedocs.io/en/latest/.

3. Results and discussion

3.1. An overview of ECMpy 2.0

ECMpy 2.0 comprises four major modules: model parsing, parameter acquisition, model construction, and model analysis (Fig. 1). The quality of GEMs significantly impacts the construction of ecModels, especially regarding the acquisition of kinetic parameters. In the initial phase of ecModel construction, we implement model parsing rules, which involve a straightforward evaluation of metabolite BiGG IDs, gene UniProt IDs, and reaction EC number annotations coverage within the model to determine the feasibility of ecModel construction. We propose that future GEMs should adhere to the standard-GEM guidelines [22], explicitly incorporating support for diverse database IDs within the annotations of genes, metabolites, and reactions. This enhancement facilitates the construction of multi-constraint models (mcModels), encompassing enzyme constraints, thermodynamic constraints, and others.

The construction of ecModels involves critical parameters such as enzyme kinetic parameters (k_{cat}), enzyme molecular weight (MW), and the mass fraction of enzymes (f). To acquire enzyme kinetic parameters, we currently support two methods: one based on AutoPACMEN for parsing the BRENDA and SABIO-RK databases, and another based on DLKcat for direct prediction. We have expanded the capabilities of AutoPACMEN to utilize specific activity [5] (Eq. (1)) data from BRENDA, thereby broadening the coverage of enzyme kinetic parameters. However, AutoPACMEN is more suitable for organisms with

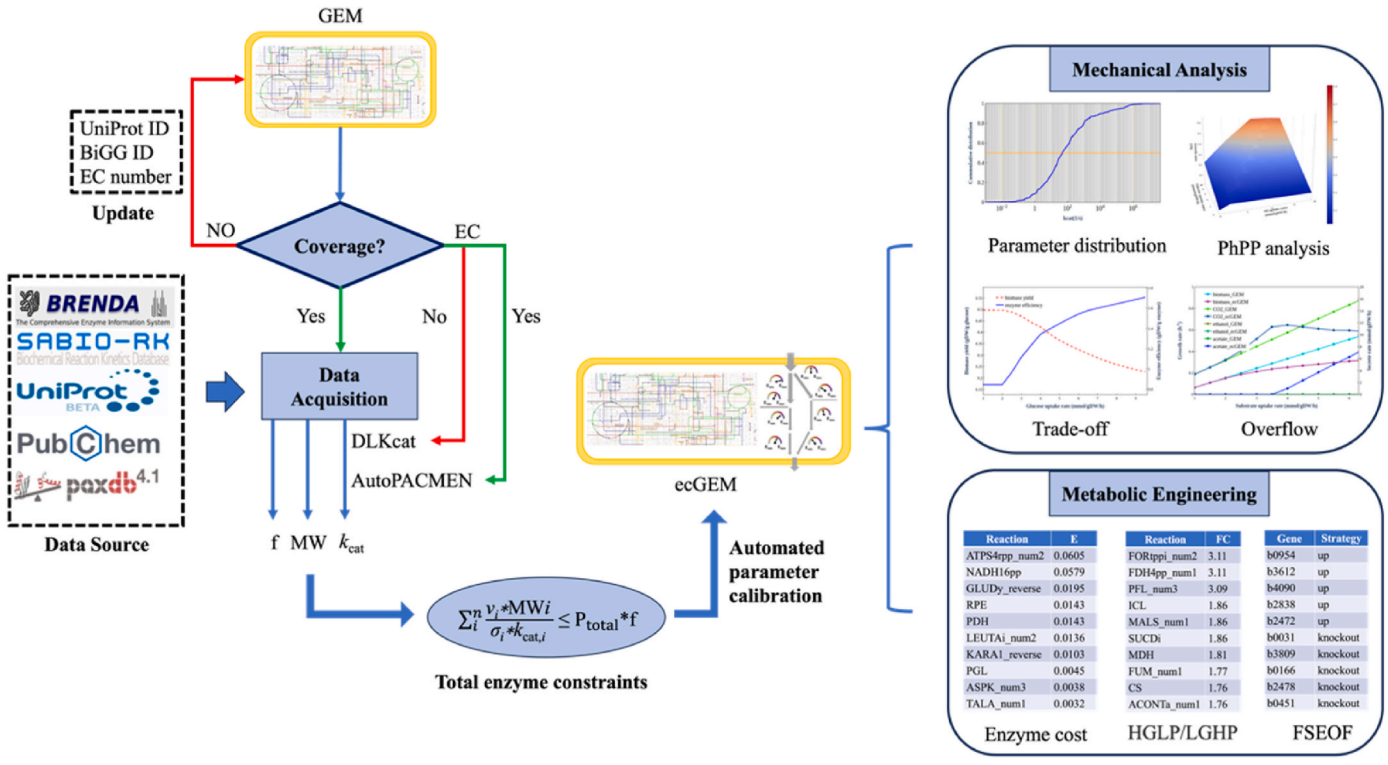

Fig. 1. The framework of ECMpy 2.0. The framework comprises four main modules: firstly, the model parsing module parses the model to assess the coverage of UniProt ID, BiGG ID, and EC numbers; secondly, the parameter acquisition module retrieves the required f, MW, and k_{cat} data from various data sources for modeling; thirdly, the model construction module adds enzyme constraints directly by incorporating total enzyme equations and performs parameter calibration; finally, the model analysis module allows users to conduct mechanistic analysis and predict metabolic engineering targets.

Table 1
Recent workflow for constructing and analyzing ecModels.

Method	AutoPACMEN	GECKO	ECMpy
k_{cat} source	BRENDA and SABIO-RK	BRENDA; Deep learning (DLKcat)	BRENDA and SABIO-RK; Deep learning (DLKcat)
Protein Subunit composition source	–	–	Automatic parsing of UniProt
Method of adding enzyme constraints	To each enzyme catalyzed reaction, add enzyme usage as substrate to the S-matrix, and one total protein exchange reaction.	For each enzyme-catalyzed reaction, add enzyme usage as a substrate to the S-matrix, and include an equal number of protein exchange reactions in the S-matrix.	no metabolites or reactions added, only a total protein constraint outside the S-matrix.
Model complexity	Medium	High	Low
Simulation utilities	–	Flux variability analysis; Model parameter analysis	The S-matrix remains unchanged, ensuring compatibility with any GEM analysis method. Additionally, ECMpy 2.0 integrates model parameter analysis, phenotype phase plane Analysis, metabolic overflow simulation, and trade-off phenomenon simulation.
Prediction of metabolic engineering targets	–	FSEOF	Three methods: enzyme cost, the enzyme cost differences in different conditions, and FSEOF
Year	2020	2017, 2022, 2024	2021
Reference	[4]	[3,7,8]	[6]
Platform	Python	MATLAB/Python	Python

abundant enzyme kinetic parameters in databases, such as model organisms (e.g., *E. coli*, *C. glutamicum*, and *S. cerevisiae*). For most non-model organisms, this approach often yields unreliable k_{cat}. Hence, in ECMpy 2.0, we have integrated the DLKcat method, providing enhanced enzyme kinetic parameters, particularly for non-model organisms. In terms of obtaining enzyme molecular weight data, inspired by GPRuler [23], we have developed a method that automatically parses UniProt annotations to retrieve protein subunit composition information, facilitating the subsequent computation of molecular weights for corresponding reactions [10,11]. For proteins not record on UniProt, ECMpy 2.0 defaults to considering it as a monomer, similar to the approach taken by AutoPACMEN and GECKO. To calculate the f value, we provide a computational function that takes the model and protein abundance data (accessible from PAXdb [24] or determined through experimentation) as inputs, enabling direct computation of the f value.

The model construction module in ECMpy employs a streamlined process, introducing enzyme constraints by directly incorporating the total enzyme amount constraint and enabling automated calibration of enzyme kinetic parameters based on enzyme utilization. Regarding the model analysis module, commonly used analysis methods for ecModels have been integrated, including model parameter analysis, PhPP analysis, metabolic overflow simulation, and trade-off phenomenon simulation. A notable achievement is the seamless integration of three distinct metabolic engineering target prediction methodologies: an enzyme cost-based sorting approach, the FSEOF-based method, and the HGLP/LGHP method.

3.2. Comparison of enzyme-constrained model construction and analysis tools

Currently, three primary workflows exist for constructing ecModels: AutoPACMEN, GECKO, and ECMpy. These methods, originating from FBAwMC (flux balance analysis with molecular crowding) [25], share a common principle focusing on enzyme selection and quantification for synthesis, emphasizing pathways with enzymes combining low molecular weight and high catalytic capacity. The key distinction lies in the approach to introducing total enzyme amount constraint. AutoPACMEN and GECKO utilize pseudo-metabolites and pseudo-exchange reactions, while ECMpy imposes a total protein constraint outside the S-matrix, resulting in varying model complexities.

For acquiring enzyme kinetic parameters, all three processes extract data from authoritative enzyme databases, BRENDA and/or SABIO-RK (Table 1). ECMpy 2.0 and GECKO enhances coverage by integrating the DLKcat method, a machine learning approach for predicting enzyme kinetic parameters, extending applicability to non-model organisms. Recognizing the influence of molecular weight on reaction fluxes, GECKO, AutoPACMEN, and ECMpy require users to manually organize protein subunit composition data; otherwise, it defaults to monomers (with a subunit count of 1). ECMpy 2.0 introduces automated parsing of UniProt annotations for this purpose, enhancing the accuracy of molecular weight calculations for enzyme-constrained models.

In terms of model analysis and metabolic engineering target prediction, GECKO incorporates FVA and FSEOF algorithms. ECMpy 2.0, by not modifying the S-matrix, accommodates diverse analytical methods within the COBRApy toolkit. In contrast, GECKO and AutoPACMEN necessitate the rewriting of analysis functions within COBRApy to align with the new S-matrix structure. It introduces ecModel analysis and visualization tools, including model parameter analysis, PhPP analysis, metabolic overflow simulation, and trade-off phenomenon simulation. Additionally, three published ecModel methods for metabolic engineering target prediction—enzyme cost, HGLP/LGHP, and FSEOF—are integrated. ECMpy2.0 is subject to limitations and currently lacks integration with proteomic data, unlike GECKO.

AutoPACMEN and ECMpy 2.0 are solely written in Python, ensuring universality and extensibility. GECKO supports both MATLAB and Python versions through geckopy [26] but lags in development, not fully incorporating all GECKO analysis methods (e.g., FSEOF).

3.3. ecModel construction and analysis

ECMpy 2.0 provides two methodologies for ecModel construction. The first involves a step-by-step process through specific notebook files

(00.Model_preview.ipynb, 01.get_reaction_kcat_using_DLKcat.ipynb/, 01.get_reaction_kcat_using_AutoPACMEN.ipynb, 02.get_ecModel_using_ECMpy.ipynb, and 03.ecModel_calibration.ipynb). Alternatively, a one-click model construction is possible via the command line (06.One-click_modeling.ipynb). This notebook offers four one-click modeling options that can be combined in various ways:

(1) Direct utilization of user-provided k_{cat}, MW, and f.
(2) Utilization of user-provided k_{cat} and MW, with f values calculated from protein abundance data.
(3) Calculation of k_{cat}, MW, and f values, with k_{cat} data parsed using AutoPACMEN.
(4) Calculation of k_{cat}, MW, and f values, with k_{cat} data predicted using DLKcat.

As of now, the one-click modeling process for Method 4 takes approximately 3–4 h to complete. The primary bottleneck is the automated acquisition of k_{cat} and MW data, where tasks like parsing protein subunit composition data take around 1–1.5 h, and metabolite ID conversion takes around 1–2 h. The data retrieval speed depends on internet speed, suggesting potential improvements like caching such data in future ECMpy. For instance, pre-parsing protein subunit composition data for all species to create a dedicated matching file and pre-arranging a metabolite ID conversion file. Addressing these data matching and ID conversion challenges is anticipated to reduce ecModel construction time to within 0.5 h.

To validate the feasibility of ECMpy 2.0, we constructed ecGEMs using 108 models stored in the BiGG database [27], with the majority being *Escherichia coli* (58 models). During the modeling process, we excluded five models whose genes could not be mapped to UniProt, as well as five mammalian models (of limited utility for ecGEMs), including four *Homo sapiens* models and one *Mus musculus* model. Additionally, for cases with more than two models per species, only one model was retained. Consequently, a total of 20 models were ultimately selected for the construction of ecGEMs, encompassing diverse species such as bacteria (including 16 species like *E. coli*, *Pseudomonas putida*, *Bacillus subtilis*, etc.), fungi (*Saccharomyces cerevisiae*), protozoa (*Plasmodium falciparum* and *Trypanosoma cruzi*), and algae (*Phaeodactylum tricornutum*) (detailed in 07.BiGG_to_ecGEM.ipynb). The ecGEMs mentioned above can be directly accessed through GitHub at the following link: https://github.com/tibbdc/ECMpy/tree/master/model/BiGG/. However, it is worth noting that these models are still in their preliminary stages. To be used effectively, they require further kinetic parameter corrections based on experimental results.

To demonstrate how ECMpy 2.0 constructs ecModels, we use *E. coli* as an example and employ the first method described above to build an ecModel for *E. coli*. The initial model utilized iML1515^{R} [6] and protein abundance data from PAXdb. After the initial model construction, automated calibration was performed based on the experimentally determined *E. coli* growth rate (0.66 h^{-1} [18]). Following 49 rounds of calibration (adjustable threshold; set here to fewer than 50 iterations of calibration), the simulated growth rate was adjusted to 0.401 h^{-1}. It's important to note that the growth rate has not been iterated here to match the experimental growth rate. Additionally, it is advisable to perform manual corrections using C13 flux data [6] or protein abundance data, as the flux distribution results from the automated calibration process may differ from experimental values.

Afterwards, we utilized the manually corrected ecModel of *E. coli* (eciML1515 [6]) to showcase the analysis capabilities of ECMpy2.0. Initially, ECMpy 2.0's model parameter analysis module was employed to clearly illustrate distinct features of k_{cat} and MW (Fig. 2 A and B). Further analysis using PhPP visualization provided a global perspective on how changes in two environmental variables, such as carbon and oxygen uptake rates, impact optimal growth rates (Fig. 2C and D). The introduction of enzyme constraints significantly reduced the solution space (Fig. 2 D). Utilizing ECMpy 2.0's overflow simulation module, an analysis of *E. coli* metabolic overflow was conducted, and a visualization graph illustrated the stages of overflow occurrence and overflow byproducts (Fig. 2E). Moreover, the integration of the trade-off phenomenon simulation module revealed a distinct trade-off between yield and enzyme usage efficiency in *E. coli*'s metabolic process (Fig. 2F). Additionally, the metabolic processes were categorized into the substrate-limited stage (less than 6 mmol/gDCW/h), overflow switching stage (between 6 and 6.5 mmol/gDCW/h), and overflow stage (greater than 6.5 mmol/gDCW/h) (Fig. 2 E and F). Finally, the phenotype simulation module allows for the simulation and comparison of phenotype outcomes under different experimental conditions. It's important to note that the comparative functionality of phenotype results may not be applicable to all models and requires supporting phenotype data. Here, we demonstrate the phenotype simulation and comparison capabilities of ECMpy2.0 using the example of the maximum growth rates of *E. coli* on 24 different carbon sources [18]. The predicted results showed good agreement with previously reported experimental data [18], as illustrated in Fig. 2G), and outperformed non-enzyme-constrained models (Fig. 2H). This indicates that the introduction of enzyme constraint conditions enables the model to simulate phenotypes more accurately. Moreover, we conducted a comparison of computational time for calculating the maximum growth rates of ecGEMs constructed using three methods (ECMpy 2.0, GECKO, and AutoPACMEN) under these 24 carbon source conditions. Our analysis revealed that models built with the GECKO method demonstrated slightly slower computational times (all exceeding 0.8 s) compared to the other two models, primarily attributed to the larger number of metabolites and reactions in the GECKO model (Supplementary Table S1). Nevertheless, for users, the discrepancy in computational speed may not be substantial.

3.4. Applications in metabolic engineering

GEMs provide fundamental insights for predicting targets in metabolic engineering. In contrast to GEMs, ecModels possess the capability to compute enzyme costs alongside reaction fluxes, allowing for the identification of pivotal enzymes within pathways. ECMpy 2.0 introduces three algorithms for pinpointing metabolic engineering targets, demonstrated through a case study focused on tryptophan production in *E. coli* (utilizing eciML1515 [6]), showcasing the effectiveness of each approach. Firstly, we evaluated the enzyme cost of each reaction to identify kinetic bottleneck reactions, categorized as reactions with the highest enzyme costs. In this analysis, glucose served as the substrate, tryptophan as the objective, and the lower bound of the biomass reaction was set at 10 % of the maximal growth rate [11]. Subsequently, reactions with the greatest enzyme costs (exceeding 1 % of total enzyme capacity) were identified as potential targets for metabolic engineering (Supplementary Table S2). Following that, we computed the cost of reactions across two pathways: HGLP (growth rate set at 0.6 h^{-1}) and LGHP (growth rate set at 0.1 h^{-1}). Reactions exhibiting fold changes in enzyme cost greater than 1.5 were selected as candidates for metabolic engineering [10]. This method not only identifies overexpression targets but also reveals potential targets for attenuation (Supplementary Table S2). Lastly, the FSEOF method was employed to scan relevant

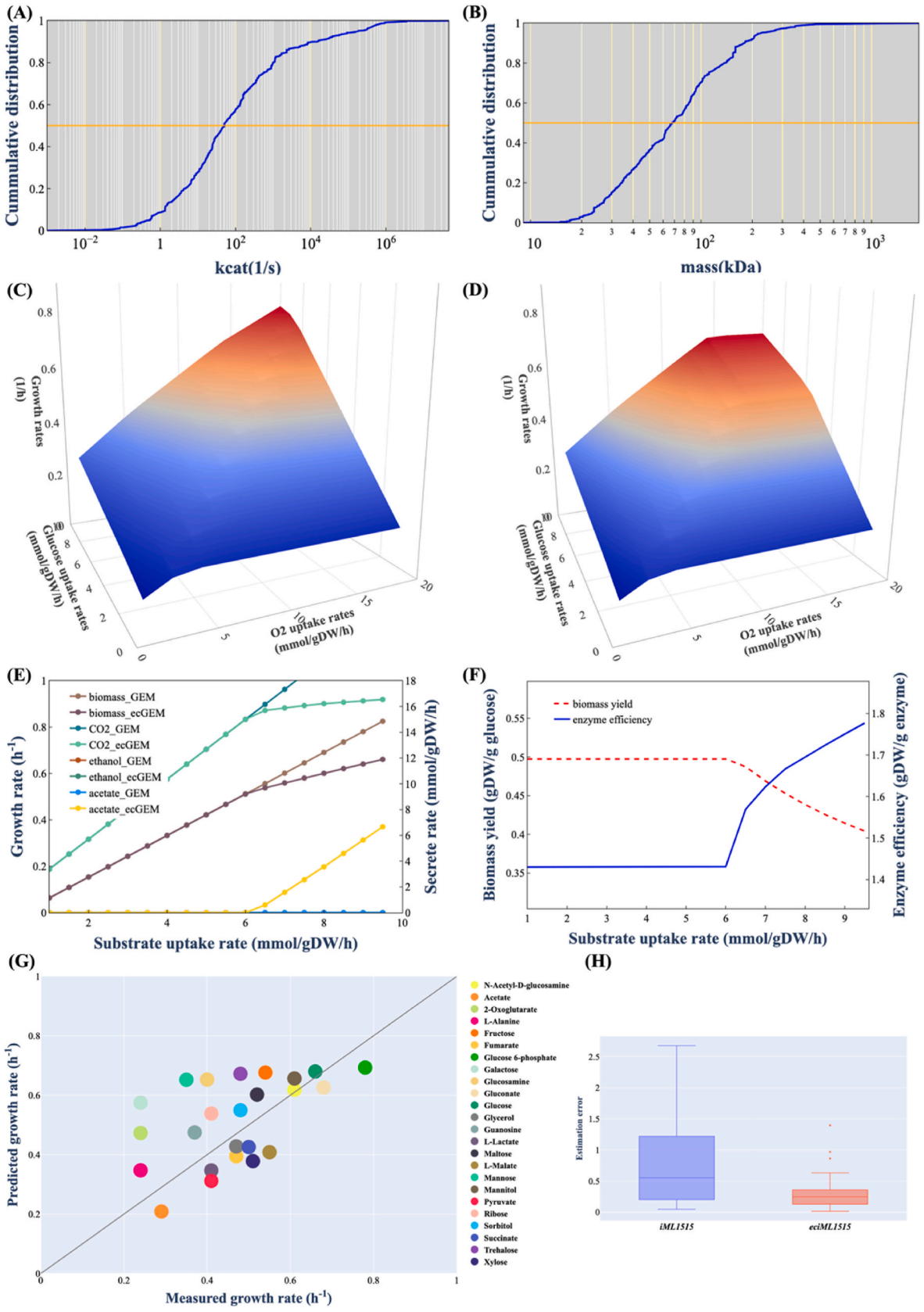

Fig. 2. Construction and Analysis of ecModel. (A) Cumulative distribution of k_{cat} values. (B) Cumulative distribution of molecular weights. Changes in the maximal growth rate with the increase of glucose and oxygen uptake rates in iML1515 (C) and eciML1515 (D). Comparison of *in silico* overflow between iML1515 and eciML1515. (F) Trade-off phenomenon simulated by eciML1515. (G) Predicted *E. coli* growth rates on different carbon sources using eciML1515. (H) Distribution of prediction errors of internal fluxes from different models.

targets within the tryptophan synthesis pathway. In each simulation, the objective function aimed to maximize tryptophan production while determining an optimum solution for each biomass-yield condition.

Upon comparing the target predictions generated by these three methods, we identified eight reactions (ANPRT, ANS, CHORS, IGPS, PRAIi, PRPPS, SHK3Dr, and TRPS3) predicted simultaneously by all three algorithms (Fig. 3 and Supplementary Table S2). Notably, five of these reactions were corroborated by existing literature [28]. Furthermore, comparing HGLP/LGHP and FSEOF revealed 13 consistent overexpression targets. Almost half of these targets were already documented in the existing literature [28] (refer to Table 2). Both of these methods also identified a significant number of attenuation targets (refer to Supplementary Table S2).

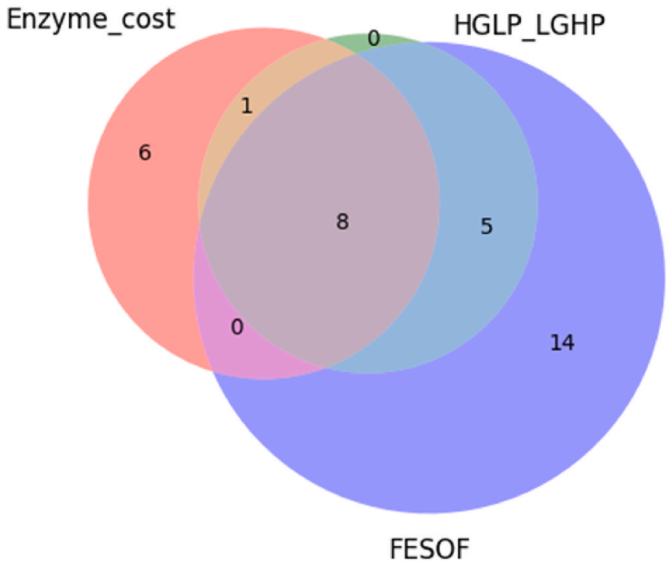

Fig. 3. Comparative Analysis of Predicted Overexpression Targets. Dark blue: targets predicted only by the FSEOF method. Light red: targets predicted only by the Enzyme cost method. Light green: targets predicted only by the HGLP/LGHP method. Lavender: targets predicted by all three methods. Light blue: targets predicted by both the HGLP/LGHP and FSEOF methods. Light brown: targets predicted by both the HGLP/LGHP and Enzyme cost methods. Mauve: targets predicted by both the FSEOF and Enzyme cost methods.

Table 2
Metabolic engineering target prediction.

	Enzyme cost	HGLP/LGHP	FSEOF
Overexpression targets	ANS [28], TRPAS2_reverse, ANPRT [28], TRPS3 [28], PRAIi [28], IGPS [28], GLCDpp, PRPPS, GNK, PDH, CHORS, PGK_reverse, PGM_reverse, SHK3Dr, KARA1_reverse	SHK3Dr, DHQTi, IGPS [28], TRPS3 [28], PRAIi [28], PRPPS, SHKK, CHORS, DHQS, PSCVT, TRPAS2_reverse, ANS [28], ANPRT [28], DDPA [28]	H2Otpp_reverse, H2Otex_reverse, CO2tex_reverse, Htex_reverse, ACtex_reverse, TRPtex_reverse, TRPt2rpp_reverse, ANS [28], TRPS3 [28], ANPRT [28] IGPS [28], PRAIi [28], SHKK, DHQTi, CHORS, DHQS, DDPA [28], PSCVT, SHK3Dr, NH4tpp, NH4tex, PGI, PRPPS, RPI_reverse, GLNS, PPA, ADK1
Attenuation targets	–	78 reactions	340 reactions
Knockout targets	–	–	12 reactions

4. Conclusions

In summary, we have introduced ECMpy 2.0, the first comprehensive automated toolkit that integrates data acquisition, model construction, model refinement, model analysis and visualization, and predictive targeting for metabolic engineering. This advanced toolkit not only streamlines the extraction and acquisition of enzyme kinetic parameters and protein subunit composition data but also introduces a machine learning approach for predicting enzyme kinetic parameters. This innovation enables the construction of ecModels even for species with limited kinetic annotations.

ECMpy 2.0 represents a groundbreaking integration of model analysis and visualization capabilities within the domain of ecModel construction tools. This enhancement not only enables users to interact with the model-generated results but also facilitates in-depth exploration. Moreover, ECMpy 2.0 seamlessly incorporates three established algorithms utilizing ecModels to identify targets for metabolic engineering, thereby lowering barriers to ecModel application.

The integration of enzyme constraints significantly improves the predictive accuracy of GEMs, aligning model predictions more closely with experimental measurements. However, given the inherent complexity of biological systems, relying solely on enzymatic constraints proves insufficient for a comprehensive description. Hence, there is a necessity to incorporate additional biological data into novel composite constraints, such as thermodynamics [29] and regulatory networks [30], or to construct a whole-cell GEM [31]. Furthermore, in terms of visualization of results, there is a lack of pathway visualization functionality similar to what CAVE [32] provides. This feature is crucial for users to assess pathways, make model adjustments, and showcase results. This functionality will be updated in future versions of ECMpy.

5. Availability of data and materials

ECMpy 2.0 is available at https://github.com/tibbdc/ECMpy or as a pip package (https://pypi.org/project/ECMpy). The repository provides a detailed notebook that thoroughly documents data acquisition, model construction, model analysis and visualization, as well as how to apply it to metabolic engineering. The documentation for the code can be found at https://ecmpy.readthedocs.io/en/latest/.

CRediT authorship contribution statement

Zhitao Mao: Software, development, Software, testing, Manuscript drafting, Manuscript review and editing. **Jinhui Niu:** Software, development. **Jianxiao Zhao:** Software, development. **Yuanyuan Huang:** Data acquisition, Software, testing. **Ke Wu:** Software, development. **Liyuan Yun:** Software, testing. **Jirun Guan:** Software, testing. **Qianqian Yuan:** Software, testing. **Xiaoping Liao:** Supervision. **Zhiwen Wang:** Supervision. **Hongwu Ma:** Supervision, Manuscript review and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was funded by the National Key Research and Development Program of China (2021YFC2100700), National Natural Science Foundation of China (32300529, 32201242, 12326611), Tianjin Synthetic Biotechnology Innovation Capacity Improvement Projects (TSBICIP-PTJS-001, TSBICIP-PTJS-002, TSBICIP-PTJJ-007), and Major Program of Haihe Laboratory of Synthetic Biology (22HHSWSS00021), and Strategic Priority Research Program of the Chinese Academy of

Sciences (XDB0480000).

Here we are deeply grateful to klamt-lab for releasing the code for AutoPACMEN (https://github.com/klamt-lab/autopacmen) and to SysBioChalmers for sharing the code for DLKcat (https://github.com/SysBioChalmers/DLKcat), which enables ECMpy 2.0 to rapidly obtain enzyme kinetics parameter information for the corresponding models. We extend our heartfelt thanks to qLSLab for making the code for GPRuler available (https://github.com/qLSLab/GPRuler), as it has inspired ideas for ECMpy 2.0 to automatically acquire the subunit composition of proteins.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.synbio.2024.04.005.

References

[1] Maia P, Rocha M, Rocha I. *In silico* constraint-based strain optimization methods: the quest for optimal cell factories. Microbiol Mol Biol Rev 2016;80:45–67.
[2] Massaiu I, Pasotti L, Sonnenschein N, Rama E, Cavaletti M, Magni P, Calvio C, Herrgård MJ. Integration of enzymatic data in *Bacillus subtilis* genome-scale metabolic model improves phenotype predictions and enables in silico design of poly-γ-glutamic acid production strains. Microb Cell Factories 2019;18:3.
[3] Chen Y, Gustafsson J, Tafur Rangel A, Anton M, Domenzain I, Kittikunapong C, Li F, Yuan L, Nielsen J, Kerkhoven EJ. Reconstruction, simulation and analysis of enzyme-constrained metabolic models using GECKO toolbox 3.0. Nat Protoc 2024;19:629–67.
[4] Bekiaris PS, Klamt S. Automatic construction of metabolic models with enzyme constraints. BMC Bioinf 2020;21:19.
[5] Ye C, Luo Q, Guo L, Gao C, Xu N, Zhang L, Liu L, Chen X. Improving lysine production through construction of an *Escherichia coli* enzyme-constrained model. Biotechnol Bioeng 2020;117:3533–44.
[6] Mao Z, Zhao X, Yang X, Zhang P, Du J, Yuan Q, Ma H. ECMpy, a simplified workflow for constructing enzymatic constrained metabolic network model. Biomolecules 2022;12:65.
[7] Sánchez BJ, Zhang C, Nilsson A, Lahtvee P-J, Kerkhoven EJ, Nielsen J. Improving the phenotype predictions of a yeast genome-scale metabolic model by incorporating enzymatic constraints. Mol Syst Biol 2017;13:935.
[8] Domenzain I, Sánchez B, Anton M, Kerkhoven EJ, Millán-Oropeza A, Henry C, Siewers V, Morrissey JP, Sonnenschein N, Nielsen J. Reconstruction of a catalogue of genome-scale metabolic models with enzymatic constraints using GECKO 2.0. Nat Commun 2022;13:3766.
[9] Zhou J, Zhuang Y, Xia J. Integration of enzyme constraints in a genome-scale metabolic model of *Aspergillus niger* improves phenotype predictions. Microb Cell Factories 2021;20:125.
[10] Niu J, Mao Z, Mao Y, Wu K, Shi Z, Yuan Q, Cai J, Ma H. Construction and analysis of an enzyme-constrained metabolic model of. Corynebacterium Glutamicum Biomol. 2022;12:1499.
[11] Wu K, Mao Z, Mao Y, Niu J, Cai J, Yuan Q, Yun L, Liao X, Wang Z, Ma H. ecBSU1: a genome-scale enzyme-constrained model of *Bacillus subtilis* based on the ECMpy workflow. Microorganisms 2023;11:178.
[12] Ebrahim A, Lerman JA, Palsson BO, Hyduke DR. COBRApy: COnstraints-based reconstruction and analysis for Python. BMC Syst Biol 2013;7:74.
[13] Vlassis N, Pacheco MP, Sauter T. Fast reconstruction of compact context-specific metabolic network models. PLoS Comput Biol 2014;10:e1003424.
[14] Li F, Yuan L, Lu H, Li G, Chen Y, Engqvist MKM, Kerkhoven EJ, Nielsen J. Deep learning-based kcat prediction enables improved enzyme-constrained model reconstruction. Nat Catal 2022;5:662–72.
[15] Consortium TU. UniProt: the universal protein knowledgebase in 2023. Nucleic Acids Res 2022;51:D523–31.
[16] Ishchuk OP, Domenzain I, Sánchez BJ, Muñiz-Paredes F, Martínez JL, Nielsen J, Petranovic D. Genome-scale modeling drives 70-fold improvement of intracellular heme production in. Saccharomyces cerevisiae Proc Natl Acad Sci 2022;119:e2108245119.
[17] Choi HS, Lee SY, Kim TY, Woo HM. *In silico* identification of gene amplification targets for improvement of lycopene production. Appl Environ Microbiol 2010;76:3097–105.
[18] Adadi R, Volkmer B, Milo R, Heinemann M, Shlomi T. Prediction of microbial growth rate versus biomass yield by a metabolic network with kinetic parameters. PLoS Comput Biol 2012;8:e1002575.
[19] Chang A, Jeske L, Ulbrich S, Hofmann J, Koblitz J, Schomburg I, Neumann-Schaal M, Jahn D, Schomburg D. BRENDA, the ELIXIR core data resource in 2021: new developments and updates. Nucleic Acids Res 2020;49:D498–508.
[20] Wittig U, Rey M, Weidemann A, Kania R, Müller W. SABIO-RK: an updated resource for manually curated biochemical reaction kinetics. Nucleic Acids Res 2017;46:D656–60.
[21] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V. Scikit-learn: machine learning in Python. J Mach Learn Res 2011;12:2825–30.
[22] Anton M, Almaas E, Benfeitas R, Benito-Vaquerizo S, Blank LM, Dräger A, Hancock JM, Kittikunapong C, König M, Li F, et al. standard-GEM: standardization of open-source genome-scale metabolic models. bioRxiv 2023;2023.2003.2021.512712.
[23] Di Filippo M, Damiani C, Pescini D. GPRuler: metabolic gene-protein-reaction rules automatic reconstruction. PLoS Comput Biol 2021;17:e1009550.
[24] Wang M, Weiss M, Simonovic M, Haertinger G, Schrimpf SP, Hengartner MO, von Mering C. PaxDb, a database of protein abundance averages across all three domains of life*. Mol Cell Proteomics 2012;11:492–500.
[25] Beg QK, Vazquez A, Ernst J, de Menezes MA, Bar-Joseph Z, Barabási A-L, Oltvai ZN. Intracellular crowding defines the mode and sequence of substrate uptake by *Escherichia coli* and constrains its metabolic activity. Proc Natl Acad Sci USA 2007;104:12663–8.
[26] Muriel JC, Long C, Sonnenschein N. Simultaneous application of enzyme and thermodynamic constraints to metabolic models using an updated Python implementation of GECKO. Microbiol Spectr 2023;11:e01705. 01723.
[27] King ZA, Lu J, Dräger A, Miller P, Federowicz S, Lerman JA, Ebrahim A, Palsson BO, Lewis NE. BiGG Models: a platform for integrating, standardizing and sharing genome-scale models. Nucleic Acids Res 2015;44:D515–22.
[28] Guo L, Ding S, Liu Y, Gao C, Hu G, Song W, Liu J, Chen X, Liu L. Enhancing tryptophan production by balancing precursors in. Escherichia coli Biotechnol Bioeng 2022;119:983–93.
[29] Yang X, Mao Z, Zhao X, Wang R, Zhang P, Cai J, Xue C, Ma H. Integrating thermodynamic and enzymatic constraints into genome-scale metabolic models. MetaEscherichia coli Biotechnol Bioengb Eng 2021;67:133–44.
[30] Shen F, Sun R, Yao J, Li J, Liu Q, Price ND, Liu C, Wang Z. OptRAM: in-silico strain design via integrative regulatory-metabolic network modeling. PLoS Comput Biol 2019;15:e1006835.
[31] Karr Jonathan R, Sanghvi Jayodita C, Macklin Derek N, Gutschow Miriam V, Jacobs Jared M, Bolival B, Assad-Garcia N, Glass John I. Covert Markus W: a whole-cell computational model predicts phenotype from genotype. Cell 2012;150:389–401.
[32] Mao Z, Yuan Q, Li H, Zhang Y, Huang Y, Yang C, Wang R, Yang Y, Wu Y, Yang S, et al. CAVE: a cloud-based platform for analysis and visualization of metabolic pathways. Nucleic Acids Res 2023;51:W70–7.