Implementation of a new method to ease the reconstruction of metabolic models in *merlin*

João Ribeiro[1], Fernando Cruz[1], Davide Lagoa[1], and Oscar Dias[1]

Centre of Biological Engineering, University of Minho, 4710-057 Braga, Portugal

Abstract. The study of biological systems as a whole is dubbed as systems biology. This scientific field often resorts to the use of models for achieving such aim. Genome-Scale Metabolic (GSM) models are a successful case in the field, providing important insights of an organism metabolism. Since exhaustive tasks are often associated with the reconstruction of GSM models, the generation of new tools to ease and hasten this process is an issue of relevance. In this work, two computational tools, *TranSyT* and *BioCoISO*, that ease the reconstruction of GSM models, were integrated in *merlin*, an open-source software for the assistance of the reconstruction process.

Keywords: Genome-Scale Metabolic Models · *merlin* · *TranSyT* · *BioCoISO*

1 Introduction

1.1 Context and Motivation

Systems biology is a field of study focused on understanding biological entities as a whole by accounting the interactions between the system's components [20]. Thus, the use of models that can capture these interactions while describing the biological system as a whole is a current practice in this field. For instance, metabolic models have been widely used over the years to describe the metabolic behavior of a given organism in a wide range of conditions. Due to previous advances in high-throughput sequencing techniques, the amount of organisms with its genome sequence available has been growing [16]. This has facilitated the development of these metabolic models at the genome-scale.

Genome-Scale Metabolic (GSM) models are tools in systems biology that have been built towards deeper and faster comprehension of an organism metabolism using genome-scale information. Nevertheless, other applications turned out to rise such as the guidance of metabolic engineering by contextualizing engineering efforts with whole-cell metabolism. This allowed several metabolic engineering aims such as strain optimization, production of value-added compounds and discovery of drug targets [17].

Concerning the importance and broad applications of GSM models, several efforts were made towards the development of tools that aid the reconstruction process [8,12,5,10,15,19,6]. *merlin* [5] is a user-friendly Java^{TM} application

in continuous development. This computational tool assists the reconstruction of GSM models for both eukaryotes and prokaryotes whose genomes have already been sequenced. This software makes available a framework for the reconstruction of metabolic models at genome-scale, retrieving enzymatic, transporter proteins, and cellular compartment information based on the organism's genome sequence. Furthermore, *merlin* provides a Graphical User Interface (GUI) that eases the manual curation of the genome functional annotation and network reconstruction [5].

Nevertheless, the reconstruction of high-quality models is still a complex process that requires exhaustive tasks. Thus, developing new tools for automatizing the reconstruction process is a topic of extreme relevance.

Transport Systems Tracker (*TranSyT*) is a standalone software that provides functional annotations of transmembrane transporter systems encoded in the genome as well as the respective compounds transported by these, automatically generating the transport reactions relative to each system while providing Gene-Protein-Reaction (GPR) information [13].

Biomass Constraint-based In Silico Optimization (*BioCoISO*) is a computational python tool that assists in the initial steps of network evaluation and debugging [3]. This tool can detect errors in the biomass formulation, which is one of the most important steps in the reconstruction process [21].

As relevant as these tools can be, they are not yet available in *merlin*, and thus unavailable to the user by a user-friendly interface.

1.2 Goals

This project is aimed at developing two plugins for integrating both *TranSyT* [13] and *BioCoISO* [3] tools in *merlin*. These plugins would improve the reconstruction of a GSM model in different stages. Such stages would be the retrieval of results from the annotation of transporter systems and the evaluation of the biomass formulation.

2 State of the Art

2.1 Genome-Scale Metabolic Models

GSM models are valuable computational tools that join genomics data of a given organism and biochemical information, allowing the *in silico* prediction of a given organism phenotype in response to either environmental or genetic conditions [18].

These models can be obtained from metabolic networks that comprise chemical reactions taking place inside of an organism. As for these reactions, they are primarily inferred from the functional annotation of enzymes encoded in the organism's genome sequence. The applications of GSM models are wide, markedly metabolic engineering, contextualization of high-throughput data, interrogation of multi-species relationships, network property discovery [17], among others.

The reconstruction of a GSM model is an iterative process involving strenuous efforts. The reconstruction milestones include the genome (re-)annotation, the assembly of the draft metabolic network based on the genome annotation and the inclusion of information retrieved from several biochemical databases. The further stages encompass the biomass and energy requirements formulation and the metabolic network debug. Afterwards, additional constraints are set and the conversion of the reconstructed network into a stoichiometric model is conducted. Finally, comparisons between the predictions made by the model and experimental data are performed in the model validation step [21].

2.2 Computational tools

merlin

merlin is an open-source application completely developed in Java™. *merlin* is aimed at assisting the reconstruction of GSM models. This computational tool allows one to perform genome functional annotations using homology tools such as Basic Local Alignment Search Tool (BLAST) [1] and Hidden Markov Models (HMMER) [7]. Then, *merlin* can automatically retrieve the enzymatic functions found in the genome and combine these with biochemical data from the Kyoto Encyclopedia of Genes and Genomes (KEGG) database, generating thus a reaction set that stands for the draft metabolic network. Additionally, it can provide the annotation of transporter proteins also encoded in the genome. The assembly of a compartmentalized draft GSM model is also a straightforward process. Moreover, GPR associations can be automatically inferred. Besides the several automatic tools provided by *merlin*, this computational tool also offers an user-friendly interface that allows the user to perform the manual curation of all results obtained with the automatic tools at any stage of the reconstruction [5].

This software was built on top of the Artificial Inteligent workBENCH (AIBench) [9], which is a Java™ development framework that helps the development of research applications based on Input-Processing-Output (IPO) model. Moreover, it allows one to converge all the efforts in dividing and structuring the problem into Operations, Datatypes and Views [9].

TranSyT

In the previous version, *merlin* could perform the annotation of transporter proteins using Transport Proteins Annotation and Reactions Generation (*TRIAGE*) [4]. However, this tool had some limitations related with strict rules in the pipeline for Transport Candidate Gene (TCG) identification. Also, *TRIAGE* was only usable while embedded in *merlin* [4], limiting its availability. To overcome these limitations, a standalone software named *TranSyT* was developed [13].

TranSyT can be divided into four main modules. The first module is the Transporter Classification Database (TCDB) scraper, whose functions are retrieving information related to each transport system, and generating the family-specific transport reactions. The second module is intended to identify metabolites using Biosynth [14]. The identification of hierarchical ontologies and descendants is also performed at this stage. The third module consists on the transport reactions generator. The information retrieved from the first two modules is combined together for reconstructing the final substrate specific transport reactions, thereby assigning them to each transport systems and respective ontological descendants. The last stage is based on the identification of transporter proteins based on the organism's proteome. A homology search is performed using BLAST against TCDB's complete record of genes encoding transport systems. The homology search is aimed at discovering homologous sequences in the organism's proteome associated with transport systems in TCDB. Furthermore, Transporter Classification (TC) family annotation regarding the BLAST results is also performed at this stage. The attribution of GPR reactions is proceeded using the homology results and protein complexes information retrieved from TCDB [13].

TranSyT is capable of performing the same tasks as *TRIAGE* does, but with less strict approaches. Unlike *TRIAGE*, *TranSyT*'s internal database is automatically updatable, allowing it to follow the growth pace of TCDB's database [13]. Moreover, the fact that this tool is a standalone application allows it to be integrated in other frameworks and applications, as it is to be integrated into Kbase [2]. It is worth noting that only *TRIAGE* and *TranSyT* are able to make the classification of transporter systems while generating the respective transport reactions [13].

TranSyT is already implemented as a web service using Docker (`https://www.docker.com`). Docker software is a container-orchestration tool that allows virtualization at the operating-system-level, isolating the applications from the host operating system and avoiding compatibility and stability issues, due to the requirement of several dependencies and environmental variables. Docker applications are wrapped up inside "containers" that are based on a given "Docker image", containing the required code, system tools, runtime, libraries and configuration files. To implement *TranSyT* using a software-as-a-service strategy, two types of Docker containers have been created: the manager and the workers. Both containers carry python scripts that use microframework *Flask* in order to manage Hypertext Transfer Protocol (HTTP) communications. The manager is important to coordinate the workers activity and the user requests, as well as sending warning messages, errors and the results back to *merlin*, whereas the workers contain the tool and required libraries for running *TranSyT*. The worker container can be replicated in multiple instances of the same image in order to handle multiple users' requests. A simple representation of this architecture can be found in Figure 1.

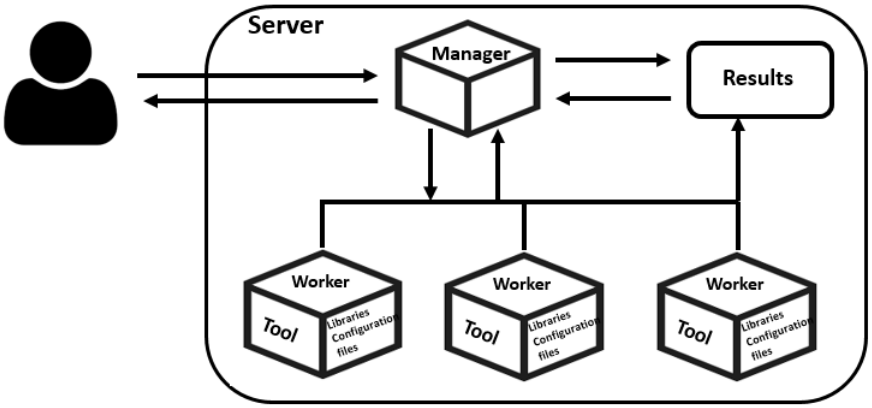

Fig. 1. Architecture of the interaction between users and *TranSyT* available into a hosting server.

To execute *TranSyT*, one must provide the following inputs: the organism's proteome, taxonomy identifier, and compounds present in the model. The output is a Systems Biology Markup Language (SBML) [11] format file containing the transport reactions generated by *TranSyT* and the associated genes boolean rules (GPR associations).

BioCoISO

BioCoISO is a python script developed to inspect the biomass formulation of a given constraint-based metabolic model [3]. This computational tool is based on the python package named COBRApy [6]. This package relies on the widely used Constraints-Based Reconstruction and Analysis (COBRA) methods [19]. These methods implement several physicochemical and biological constraints to predict feasible phenotypic states in a given condition with a given reconstructed biological network [19]. Besides COBRApy, other similar computational tools have been used over the years such as *framed* (`https://github.com/cdanielmachado/framed`).

BioCoISO can receive as input a constraint-based metabolic model SBML [11] format file and return either potential errors in the biomass formulation or gaps in the metabolic network [3]. This tool can extract complex information related with the biomass formulation such as reactants in the biomass reaction, namely the biomass single units, as well as precursors of metabolites directly associated to the biomass formulation. Besides identifying the biomass single units and other precursors, this tool can also track errors in the metabolic network that impair the synthesis of these metabolites. For that, the GSM model is iteratively simulated for the production of each biomass single unit and precursor. Thus, this tool highlights which reactions cannot carry flux towards the

production of biomass. The process can be iterated until the biomass reaction flux turn out positive [3].

At the moment, the tool can create an excel spreadsheet file compiling the flux of each reaction specifically created for accumulating the biomass single unit or precursor. This allows the user to make further analysis of errors in the GSM model reconstruction [3].

3 Methods and Problem Solution

Following the strategy adopted by *TranSyT*, *BioCoISO* will be provided as a web service implemented in Docker, so that compatibility and stability problems can be avoided while running this tool on different operating systems. Then, in order to make these tools available to *merlin*, each one must have a custom-made plugin within *merlin* to manage the interaction with the server and handle the tool's inputs and outputs. The plugins were developed through the implementation of new AIBench operations, datatypes and views, using Eclipse IDE 2018-12 (`https://www.eclipse.org/`).

The implementation of *TranSyT*'s plugin was firstly performed, so that whenever one uses it in *merlin*, the respective plugin makes an HTTP POST request to the server where the tool is located, initiating a succession of interactions between the server and *merlin* already described in Figure 1. This POST request will carry the necessary input files for *TranSyT* to classify transporters. When the request is received by the manager, it will assign the task to one of the available workers. If no workers are available to complete the tasks due to being occupied processing other tasks, the submission is refused, and the respective error response code is returned to *merlin*. Otherwise, the process is assigned to a worker which will be able to run the application. While the process is running, the plugin will constantly check in the server, through GET requests, whether the process is finished or not. For that, the plugin verifies the HTTP response code returned by the server. Likewise, before returning a response, the manager also checks whether the worker finished the process or not. When the worker is over, the results will become available in a directory created beforehand by the manager. The manager is then responsible for sending the results back to the plugin within *merlin*. These results are automatically integrated in the model, using operations already available. When the results arrive to the user machine, the process in the web service will be finished, and the worker will be available for a new submission.

A plugin very similar to the previously developed for *TranSyT* was also created in *merlin* to handle *BioCoISO*'s requests and retrievals. Besides, it was also implemented an AIBench view to render the *BioCoISO* results.

As regards to the *BioCoISO* web service implementation, although the interaction between the manager and the workers is sufficient for *TranSyT* to work, it turned out to be a limitation for *BioCoISO*. Since only some workers are available, when all of them are occupied with instructions to run, no other submissions can be made until one of them is available. This handicap

forces users to try executing the plugin until a given worker is available. Besides, given that *BioCoISO* will be run more often by a single user than *TranSyT*, the aforementioned system is not perfectly suitable for this tool. Thus, in the case of *BioCoISO*, some changes were implemented in the submission system to overcome this limitation.

This new strategy is based on a queue system encompassing three entities: the Workers Manager (WM), the Submissions Manager (SM) and the workers. It is worth noting that although WM and SM are wrapped up inside the same Docker container, they represent two different entities having different functions. Furthermore, three types of folders were created: the submissions, processing and results folders. The function of SM is to add the submission made by *BioCoISO* plugin within *merlin* to a queue as well as sending warning messages, submission status information, errors and results back to *merlin*, whereas WM controls the workers and checks whether there are any submissions in the queue or not. The workers will run the application every time this is requested by the WM. The interaction between these entities can be followed up in Figure 2.

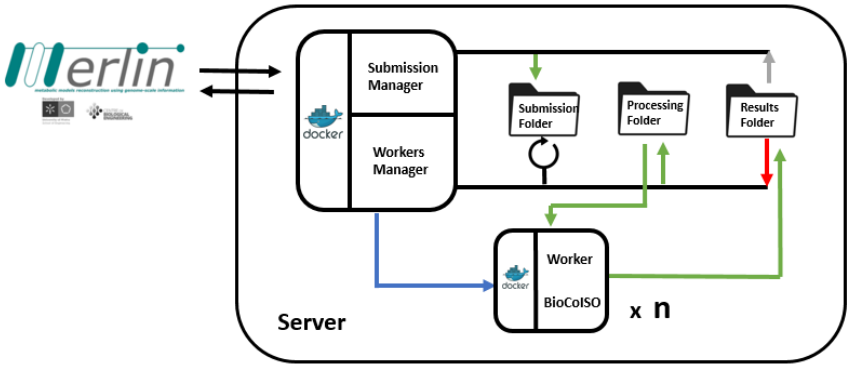

Fig. 2. Architecture of the interaction between *merlin* and the tool available in the host server. The black arrow that binds *merlin* and SM illustrates the request made by users for *BioCoISO* to run. The green arrow illustrates the files transfer into the target entity. The round arrow in black represent the iteration made by WM when searching for any submission. The blue arrow pictures the request for a worker to run whereas the red one is the flag indicating whether the results are in the results folder or not. The gray one is representative of the results download made by the SM. Lastly, the black arrow binding SM and *merlin* pictures the transfer of the results into *merlin* and the retrieval of warning messages sent by the server.

Whenever the user requests *BioCoISO* to run within *merlin*, the plugin will make a request to the server. Then, the SM adds the submission to the end of the queue, creating a folder whose name will be the submission identifier (ID). This

folder will contain the input files of the application. Simultaneously, the WM checks iteratively the submissions queue. If there are any submissions, the WM will search for an available worker. If not, the submission waits in the queue. Otherwise, a worker is assigned to process the submission and the previously created folder will be moved into the processing folder. Then, the required files for the tool to run are tracked and processed in the worker. When the process is complete, the folder whose name is the submission ID will be deleted and the worker will upload the results into the results folder. Finally, the SM will send the results back to *merlin* through the plugin. In order to free the workers for another submission, the WM is waiting for a flag in the results folder indicating that the process has already finished and the worker previously assigned to work can be freed.

Using this strategy, submissions are accumulated in the queue and processed sequentially, allowing users to make requests, even if no worker is available at that precise time. Nevertheless, like *TranSyT*'s workers, these can be also be replicated in multiple instances of the same image in order to process several submissions at the same time.

4 Results and Discussion

The present work was aimed at developing two plugins that integrate both *TranSyT* and *BioCoISO* tools in *merlin*. Correspondingly, this goal has been accomplished successfully following the methodologies previously mentioned.

As for *TranSyT*, the plugin encompasses two Java™ classes, in which are contained the AIBench operation and the algorithm for the management of submission and retrieval. The plugin is able to generate all required files for the transporters annotation, namely a text file with all metabolite IDs present in the current model, a text file with the taxonomy ID of the organism being studied and a fasta file with the proteome. When required, the plugin renders a GUI containing a field with all workspaces loaded (Figure 3A), so that the users can choose which model they want to annotate. Once the files are submitted to the server, a progress bar is displayed informing the operation status until the process is finished (Figure 3B). Lastly, the results are integrated into the model automatically (Figure 3C and Figure 3D).

Similarly, a plugin was created in order to set up *BioCoISO* in *merlin*. This plugin is composed of three Java™ classes: one to create the GUI, another to generate the AIBench operation, and a last one to manage the submissions as well as both messages and results sent by the server.

As for the operation, it is able to create the required files for the submission and control the class responsible by handling requests and retrievals. Furthermore, the operation establishes the interaction with the user by requesting an input in three different fields, as depicted in Figure 4A. The first is a combo box providing all workspaces into *merlin*. The second is an option that enables users to choose whether the goal is to verify the biomass reaction formulation or not. The third is also a combo box with all reactions in the model, so that users

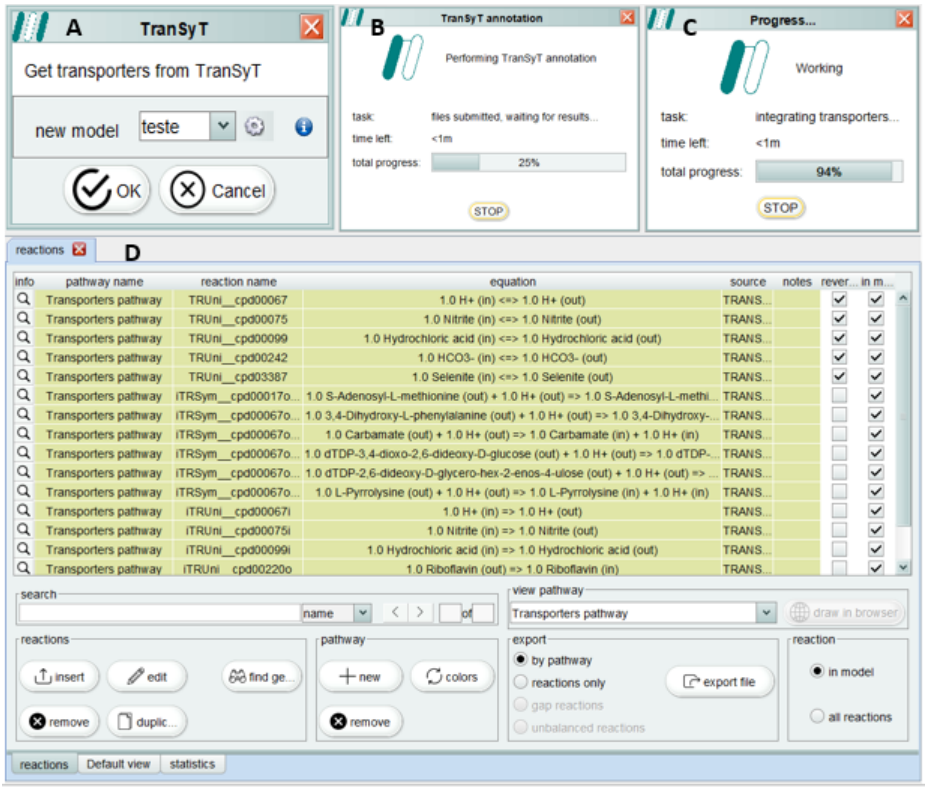

Fig. 3. *TranSyT*'s GUI inside *merlin* and the reactions view containing the transport reactions integrated.

can choose a reaction formulation to be inspected. It is worth noting that each model has a different set of reactions. Hence, the combo box with reactions has to be updated every time the workspace is changed. Therefore, when it happens, the new set of reactions has to be retrieved from the database and the content present in the combo box has to be updated. When all the fields are complete and confirmed by the user, the model for the correct workspace is exported by the plugin and the submission is initiated (Figure 4B). The last field will only be relevant in the next version of *BioCoISO*, since, by now, this tool can only take into consideration the biomass reaction instead of all.

When the required files are submitted, the plugin provides a progress bar (Figure 4B), informing the submission state. Afterwards, the result is displayed in a section created beforehand for this end.

To provide a better approach for visualizing the results, the creation of a view was performed and two more Java™ classes were developed for that purpose. Namely, one Java™ class was created to assemble a results table based on an AIBench datatype. The other Java™ class creates a section in the *merlin*

clipboard, so that the results can be displayed into a AIBench view within a subsection of *merlin*. This is represented in Figure 5.

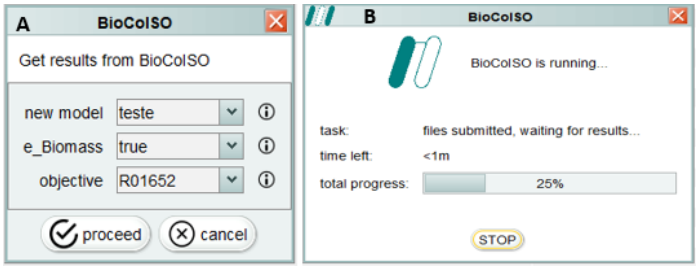

Fig. 4. *BioCoISO*'s GUI inside *merlin* and the view rendering the results.

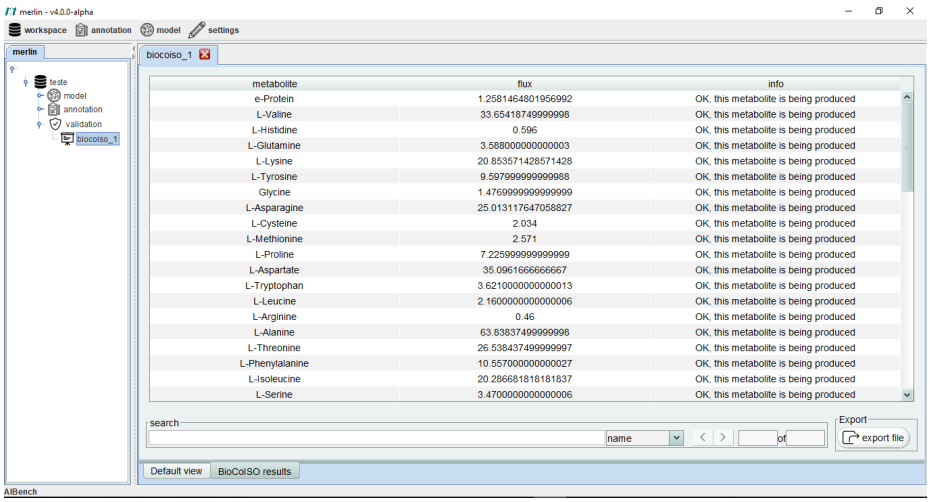

Fig. 5. *BioCoISO*'s view rendering the results.

Finally, Figure 6 highlights the interaction between the server, the plugin and the user. It is possible to notice that several warning messages appear when something goes wrong in the server (server overloaded, for instance) or while running the tool.

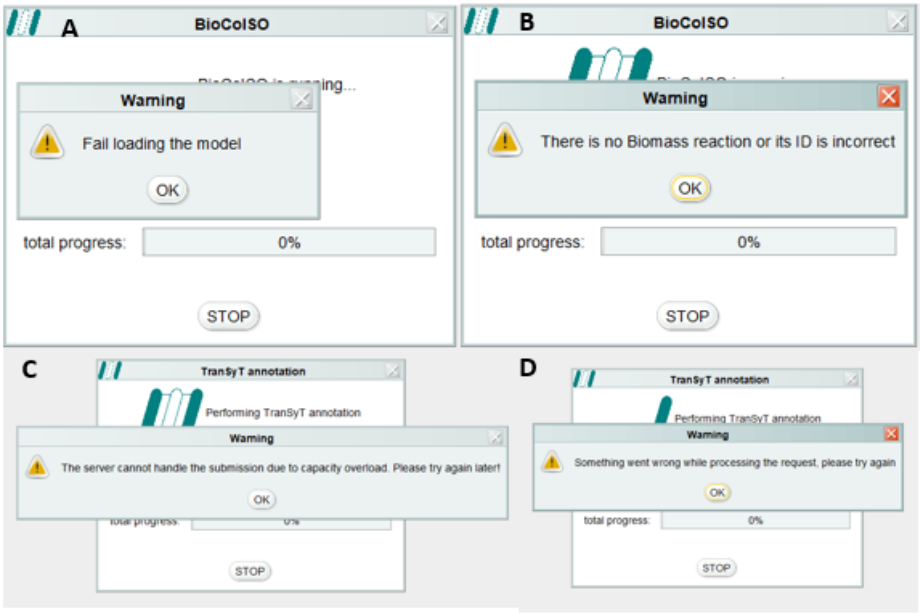

Fig. 6. Some warning messages displayed by both plugins within *merlin* GUI.

5 Conclusion

merlin is an user-friendly academic software in constant development, in which it is possible to conduct the reconstruction of a metabolic network at genome-scale. Hence, improving *merlin* with new tools and features is extremely relevant given its uniqueness and resourcefulness.

To the best of our knowledge, *TRIAGE* and *TranSyT* are the only tools capable of performing the classification of transporter systems while generating transport reactions at the same time [13]. Due to its limitations, *TRIAGE* will be no longer available in *merlin*'s new version (*merlin* 4.0). As its next iteration, *TranSyT* shall replace it.

BioCoISO was not integrated in *merlin* before the present work has been conducted. Alternatively, the functionalities of this tool were only available when run through the command line. It shall be mentioned that, for this end, several python packages had to be installed on the users machine as well as both python compiler and interpreter. Moreover, users should download the SBML [11] model and track the reaction ID for either biomass or another reaction. Additionally, the results had to be visualized in an excel spreadsheet exported for a given directory. Nevertheless, this tool helps in network evaluation and debugging, gleaning information from the biomass formulation. This information is pertinent for easing the reconstruction of GSM models, which makes its integration in *merlin* pertinent towards a more user-friendly usage. From now

on, *BioCoISO* can be used by any user without leaving *merlin* GUI. Also, the results are displayed on a view inside the workspace.

As result of this work, from this time forth, *TranSyT* and *BioCoISO* are functionally available in *merlin*, thus becoming available in an user-friendly GUI. As for *merlin*, the increased number of functionalities made this software more robust, providing more services to its users.

Overall, the plugins and their interaction with the server are working correctly and steadily, however, it shall be mentioned that some improvements are worthwhile. Although one of the plugins is almost prepared for the next version of *BioCoISO*, slight modifications must be performed in order to be perfectly adapted to it. Besides, this software is still not faster and accurate as intended, mainly because the hosting server does not yet supports IBM's CPLEX solver. Therefore, future efforts must encompass the integration of IBM's CPLEX solver into the host server, so that *BioCoISO* could generate results in a short amount of time and with increased precision. Lastly, since the queue system is not yet applied to *TranSyT*, the implementation of this type of web service is a task that shall be employed in the future too.

References

1. Altschul, S.F., et al.: Basic local alignment search tool. Journal of Molecular Biology (1990)
2. Arkin, A.P., et al.: KBase: The United States Department of Energy Systems Biology Knowledgebase. Nature Biotechnology (2018)
3. Cruz, F.: Genome-Scale Metabolic Network Reconstruction of the dairy bacterium Streptococcus thermophilus. Msc thesis, Universidade do Minho (2017)
4. Dias, O., Gomes, D., Vilaça, P., Cardoso, J., Rocha, M., Ferreira, E.C., Rocha, I.: Genome-Wide Semi-Automated Annotation of Transporter Systems. IEEE/ACM Transactions on Computational Biology and Bioinformatics (2017)
5. Dias, O., Rocha, M., Ferreira, E.C., Rocha, I.: Reconstructing genome-scale metabolic models with merlin. Nucleic Acids Research (2015)
6. Ebrahim, A., Lerman, J.A., Palsson, B.O., Hyduke, D.R.: COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Systems Biology (2013)
7. Eddy, S.R.: Profile hidden Markov models (1998)
8. Funahashi, A., et al.: CellDesigner 3.5: A versatile modeling tool for biochemical networks. Proceedings of the IEEE (2008)
9. Glez-Peña, D., Reboiro-Jato, M., Maia, P., Rocha, M., Díaz, F., Fdez-Riverola, F.: AIBench: A rapid application development framework for translational research in biomedicine. Computer Methods and Programs in Biomedicine (2010)
10. Henry, C.S., et al.: High-throughput generation, optimization and analysis of genome-scale metabolic models. Nature Biotechnology (2010)
11. Hucka, M., et al.: The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. Bioinformatics (2003)
12. King, Z.A., et al.: Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways. PLoS Computational Biology (2015)
13. Lagoa, D.: Development of Bioinformatics tools for the classication of transporter systems. Msc thesis, Universidade do Minho (2019)

14. Liu, F.: Evaluation and development of algorithms and computational tools for metabolic pathway optimization. Ph.D. thesis, Universidade do Minho (2018)
15. Machado, D., et al.: Fast automated reconstruction of genome-scale metabolic models for microbial species and communities. Nucleic Acids Research (2018)
16. Mukherjee, S., et al.: Genomes OnLine Database (GOLD) v.6: Data updates and feature enhancements. Nucleic Acids Research (2017)
17. Oberhardt, M.A., Palsson, B., Papin, J.A.: Applications of genome-scale metabolic reconstructions (2009)
18. Rocha, I., Förster, J., Nielsen, J.: Design and Application of Genome-Scale Reconstructed Metabolic Models (2008)
19. Schellenberger, J., et al.: Quantitative prediction of cellular metabolism with constraint-based models: The COBRA Toolbox v2.0. Nature Protocols (2011)
20. Snoep, J.L., Westerhoff, H.V.: From isolation to integration, a systems biology approach for building the Silicon Cell. In: Systems Biology (2005)
21. Thiele, I., Palsson, B.: A protocol for generating a high-quality genome-scale metabolic reconstruction. Nature Protocols (2010)