neo4jsbml: import systems biology markup language data into the graph database Neo4j

Guillaume Gricourt[1], Thomas Duigou[1], Sandra Dérozier[2] and Jean-Loup Faulon[1]

[1] Université Paris-Saclay, INRAE, AgroParisTech, Micalis Institute, Jouy-en-Josas, France
[2] Université Paris-Saclay, INRAE, MaIAGE, Jouy-en-Josas, France

ABSTRACT

Systems Biology Markup Language (SBML) has emerged as a standard for representing biological models, facilitating model sharing and interoperability. It stores many types of data and complex relationships, complicating data management and analysis. Traditional database management systems struggle to effectively capture these complex networks of interactions within biological systems. Graph-oriented databases perform well in managing interactions between different entities. We present neo4jsbml, a new solution that bridges the gap between the Systems Biology Markup Language data and the Neo4j database, for storing, querying and analyzing data. The Systems Biology Markup Language organizes biological entities in a hierarchical structure, reflecting their interdependencies. The inherent graphical structure represents these hierarchical relationships, offering a natural and efficient means of navigating and exploring the model's components. Neo4j is an excellent solution for handling this type of data. By representing entities as nodes and their relationships as edges, Cypher, Neo4j's query language, efficiently traverses this type of graph representing complex biological networks. We have developed neo4jsbml, a Python library for importing Systems Biology Markup Language data into a Neo4j database using a user-defined schema. By leveraging Neo4j's graphical database technology, exploration of complex biological networks becomes intuitive and information retrieval efficient. Neo4jsbml is a tool designed to import Systems Biology Markup Language data into a Neo4j database. Only the desired data is loaded into the Neo4j database. neo4jsbml is user-friendly and can become a useful new companion for visualizing and analyzing metabolic models through the Neo4j graphical database. neo4jsbml is open source software and available at https://github.com/brsynth/neo4jsbml.

Subjects Bioinformatics, Computational Biology, Computational Science
Keywords SBML, Neo4j, Database, Neo4jsbml, Genome-scale metabolic model

Submitted 22 June 2023
Accepted 5 December 2023
Published 16 January 2024

Corresponding author
Guillaume Gricourt,
guillaume.gricourt@inrae.fr

Academic editor
Ravi Kant

Additional Information and Declarations can be found on page 11

DOI 10.7717/peerj.16726

Copyright
2024 Gricourt et al.

Distributed under
Creative Commons CC-BY 4.0

OPEN ACCESS

INTRODUCTION

Genome-Scale Metabolic Models (GEMs) of biological systems are commonly represented using the Systems Biology Markup Language (SBML) format. This format is actively maintained and updated, especially, by adding new features to meet new needs. Using a standard format facilitates interoperability repeatability, and reproducibility. Several standards have emerged in systems biology (*Shin et al., 2023*). The CellML standard stores computer-based mathematical models (*Clerx et al., 2020*) and the Simulation

Experiment Description Markup Language (SED-ML) (*Waltemath et al., 2011*) enables the reproduction of simulation experiments. The use of a standard format to represent data enables the creation of a rich software environment such as COPASI (*Hoops et al., 2006*), Tellurium (*Choi et al., 2018*) and MASSpy (*Haiman et al., 2021*) for the construction, simulation, and visualization of dynamic metabolic models. A wide range of issues can be addressed in this way, including the construction of metabolic pathways (*Shen et al., 2020*), study of secondary metabolism in bacteria (*Qiu, Yang & Zeng, 2023*) and representation of cell behavior and interactions in cancer (*Kazerouni et al., 2020*). To represent these types of biological processes, the SBML format encodes several nested components and their interactions in an XML-based document. The standard represents complex systems by organizing the data as components. To illustrate, biological systems can contain thousands of metabolites and reactions, leading to a high level of complexity in their interactions (*Hucka et al., 2019*). The SBML Level 3 standard is well-suited for describing reaction-based models. To represent different types of models, such as constraint-based, logical network, and rule-based models, the SBML standard has been enhanced using different packages. As described in their specifications, these packages add components or properties to the model.

Neo4j is a powerful graph database that enables the storage, querying, and analysis of large amounts of data. Unlike relational databases, Neo4j leverages the power of graph theory, representing data as nodes or relationships which embed properties to store additional information. The graph-based structure is well-suited to tackle intricate relationships and decipher complex problems, such as social networks and network analysis. The graph query language, named Cypher, allows querying the graph to retrieve connected data and perform create, read, update, or delete operations on the database. Neo4j ensures data integrity and consistency by respecting the ACID principle (*Meier & Kaufmann, 2019*). Specific capabilities can be added to the Neo4j database through a software extension, a plugin, enabling users to customize and enhance its features for dedicated applications. The Neo4j database comes with a broad ecosystem which includes tools like Arrows for designing the database structure, numerous drivers for communicating with the database, building queries intuitively with Cypher and Bloom for visualizing data.

With Neo4j, metabolic models can be represented as a network of nodes and relationships, where each node represents a biological or supporting object within a component, such as a metabolite, reaction, or unit definition. Each relationship serves as a connection between two metabolic reactions, enabling easy exploration of different pathways and interactions between human metabolic data (*Balaur et al., 2016*). Using Neo4j with metabolic models can provide valuable insights into complex interactions, as it allows for the integration of heterogeneous data, such as chemical species, reactions, enzymes, and taxonomic data. This integration can aid in identifying key players across a wide range of biological application domains (*Swainston et al., 2017*).

More recently, based on the graph database MaSyMoS, which stores SBML and CellML models to represent biological systems in terms of functional, behavioral, and structural aspects (*Henkel, Wolkenhauer & Waltemath, 2015*), a protocol was developed to compare biochemical reaction networks (*Lambusch et al., 2018*). Nonetheless, analyzing its own

SBML data using Neo4j remains a challenge. Biochem4j provides a web interface that does not allow users to analyze their data in their database. Recon2Neo4j and MaSyMoS created Java software using the JSBML library (*Rodriguez et al., 2015*) to import the data into their database. However, data loading is performed according to a predefined schema that retain several SBML components or properties that are targeted by their applications.

We are presenting neo4jsbml, a user-friendly Python package to import SBML files into a Neo4j database. Neo4jsbml utilizes tools developed by Neo4j to define the database schema, load data, query entities and relationships, and visualize the created graph. This package offers flexibility, extensibility, and ease of use in combining graph databases with SBML files. Flexibility is provided by leaving the choice of entities to be analyzed by the researcher. Extensibility is based on the intrinsic operation of neo4jsbml *via* introspection. Finally, neo4jsbml can be used through the command line and requires no programming skill. The graph can be enriched with heterogeneous data, allowing researchers to build, manipulate, annotate, and store their data with greater efficiency.

MATERIAL AND METHODS

Implementation

SBML data are structured according to the specifications defined by *Hucka et al. (2019)*. This file format has a skeleton of main components, but optional data can be added through the use of packages. The SBML specification Level 3 Version 2 describes eleven components: function definitions, unit definitions, compartments, species, parameters, initial assignments, rules, constraints, reactions, events and the model. Each component has a specific role and stores specific and generic information. The model component is special, it serves as a container for the other components but also supports some information, such as the identifier of the model, which can be exploited.

Sometimes, not all components are suitable for analysis. To retain the desirable data from SBML documents, neo4jsbml utilizes the concept of introspection. Introspection allows a program to examine an object's characteristics, such as its name, properties, and methods, making the program more efficient and robust. It is a powerful feature of object-oriented languages that exposes details about objects at runtime, and Python ships with a few built-in functions for this purpose. When importing SBML data into Neo4j, introspection is used to automate the extraction of properties and associated structured values of items mentioned in a definition schema. In this way, the user filters the main components and attributes to be loaded into the database.

Running method

The usage of the tool is described in Fig. 1.

It takes place in two stages. First of all, the user creates a schema defining which entities will be selected from the SBML model (Fig. 2). Building a schema requires certain rules, such as mapping the names of different items found in the schema to their corresponding names as defined in the SBML specifications. To illustrate this principle, node labels and node properties indicated in the schema need to match the name of the SBML component and the properties belonging to the component, respectively. Given this information,

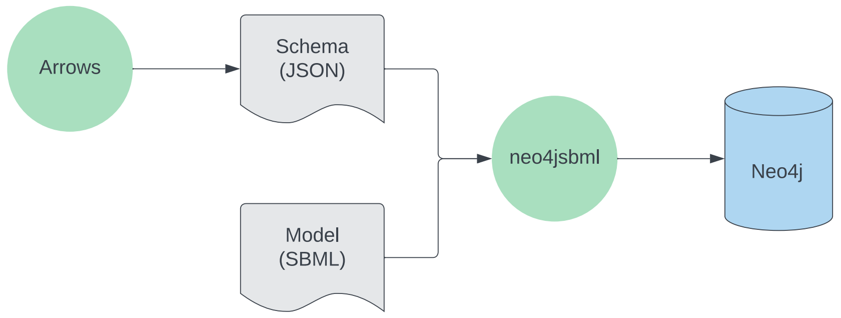

Figure 1 Workflow for utilizing neo4jsbml. First, a schema is created with https://arrows.app. Next, aided by the schema, the SBML data is loaded into the Neo4j database using neo4jsbml. Softwares are green, files are gray, the database is blue.

Full-size DOI: 10.7717/peerj.16726/fig-1

neo4jsbml identifies the objects and attributes to extract from the SBML, thanks to the library libsbml. Thus, only the entities required for analysis were loaded into Neo4j. SBML specifications indicate which components are linked to each other by storing an identifier corresponding to another component as an attribute. Therefore, neo4jsbml checks whether the two components are linked by following the strategies shown in Fig. 3. However, the Neo4j database operates on a directed graph; therefore, neo4jsbml infers directionality as indicated by the schema. Once the data are imported into Neo4j, the entities and relationships from the SBML model can be queried by Cypher through Neo4j.

Use cases

To illustrate the power of neo4jsbml, three use cases were performed involving three *Escherichia coli* GEMs. The first one, called iAF1260, was published in 2007 (*Feist et al., 2007*). The second one, named iML1515, was established in 2017 (*Monk et al., 2017*). The last one, a small-scale model baptized e_coli_core was derived from an *E. coli* model. All models were downloaded from the BiGG Models knowledgebase (*King et al., 2016*) and the MetaNetX database (*Moretti et al., 2021*).

A Neo4j plugin, named neo4jefmtool, based on efmtool (*Terzer & Stelling, 2008*) was created to enumerate the metabolic pathways, available at https://github.com/brsynth/neo4jsbml.

The schema described in Fig. 2A was used to import data into Neo4j for the first two use cases, whereas the schema corresponding to Fig. 2B was used for the last case. Both schemas were created using Arrows, available at https://arrows.app/. To provide a general overview of the use of neo4jsbml, some examples of models embedded in the SBML specifications are shown in Figs. S1, S2, and S3.

Neo4j version 5.12 and neo4jsbml version 0.12.0 (10.5281/zenodo.8419209) were used for this study.

RESULTS

We developed neo4jsbml as a modular Python package that can be used as a standalone program or incorporated as a library in other programs. It is available through the Conda

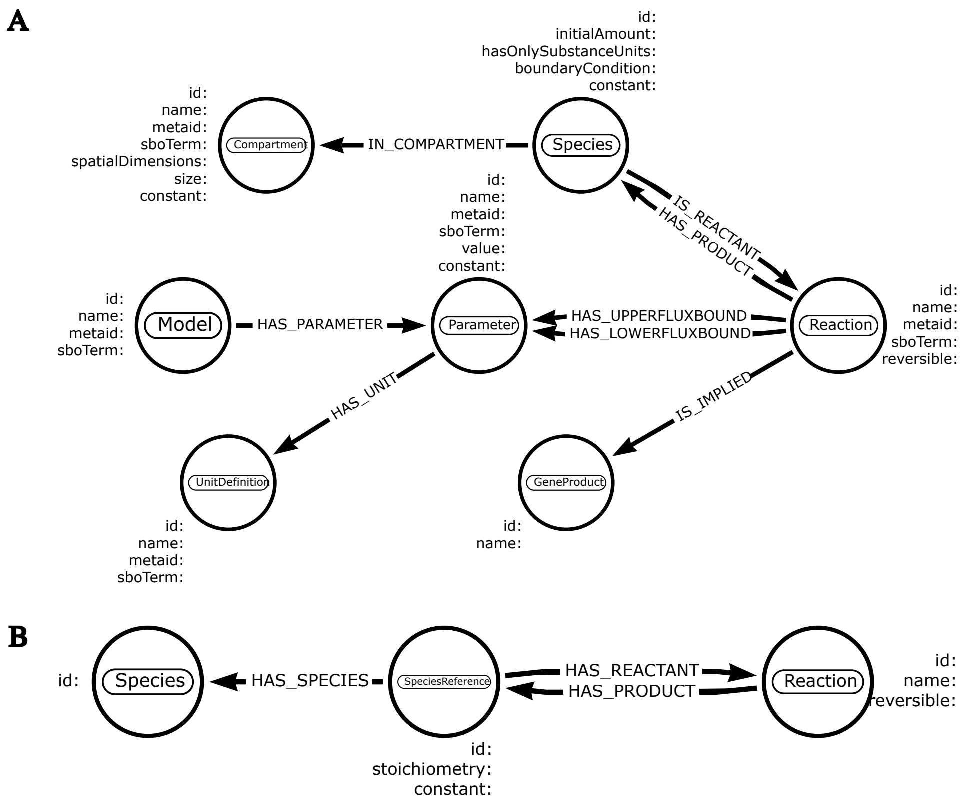

Figure 2 Schemas created from https://arrows.app. The entities and their relationships are represented by a circle and an arrow, respectively. Node label matches the name of a SBML component. Each node embeds the targeted properties. Types associated to each property are optional. (A) This schema represents various entities found in genome-scale models. (B) This schema is focused on the extraction of a metabolic network.

Full-size DOI: 10.7717/peerj.16726/fig-2

package management system (*conda-forge community, 2015*) and runs on all platforms without requiring extra privileges. SBML files are loaded by neo4jsbml, using the libsbml library (*Bornstein et al., 2008*), and the selected data will be loaded into the Neo4j database using the Python Neo4j driver, based on a database schema. Twelve packages are part or will be part of the SBML standard, but only eight have been fully implemented by the library libsbml. Neo4jsbml can consider four plugins: Flux Balance Constraints (*Olivier & Bergmann, 2018*), Groups (*Hucka & Smith, 2016*), Layout (*Gauges et al., 2015*) and Qualitative Models (*Chaouiya et al., 2015*) (Table S1). Importing data into Neo4j is done through pure Cypher queries, with no additional Neo4j plugins necessary.

To demonstrate the usefulness of neo4jsbml, we conducted a proof of concept study with three use cases.

Comparing two models

We compared two versions of the *E. coli* genome-scale metabolic model to highlight improvements between an older and a newer version from two repositories: BiGG and MetaNetX. Firstly, we compared the number of nodes and relationships in both models

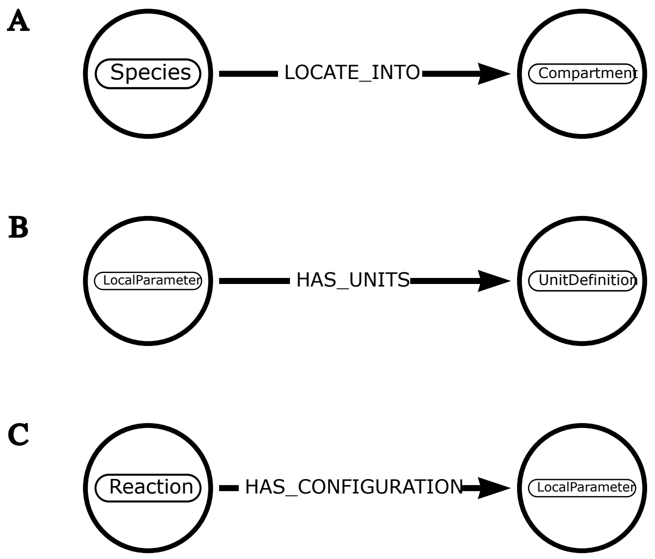

Figure 3 Examples of schemas illustrating how neo4jsbml uses introspection to associate one entity with another. (A) The object *libsbml.Species* has a method named *getCompartment()* that retrieves the identifier of the *libsbml.Compartment* object. The mapping between these entities is established by calling a method based on the name of one entity. (B) The object *libsbml.LocalParameter* has a method named *getUnits()* that retrieves the identifier of the *libsbml.UnitDefinition* object. The linkage between these entities is possible thanks to the name of the relationship. (C) The objects in the libsbml library have a method named *getListOfAllElements()* that leads to list some nested components. The *libsbml.Reaction* object can host several objects, including a *libsbml.LocalParameter* object. The relationship is created according to the name of the entities.

Full-size DOI: 10.7717/peerj.16726/fig-3

(Table 1). More than four thousand entities and twenty thousand relationships were loaded into Neo4j from the *E. coli* genome-scale models, iML1515 and iAF1260. Given one version of *E. coli*, the number of chemical species and reactions differed between the BiGG and MetaNetX databases. In addition, the iML1515 genome-scale metabolic model contained more nodes and relationships compared to the iAF1260 genome-scale metabolic model.

Next, we analyzed the reactions associated with the fumarate metabolite (Fig. 4). The Cypher query is described in Eq. (1).

```
MATCH p=((c:Compartment)<-(n:Species {id: "M_fum_c"})->(r:Reaction))          (1)
RETURN p
```

One additional reaction was linked to the fumarate metabolite in the iML1515 model compared to the iAF1260 model.

Viewing a metabolic pathway

A common application with a metabolic network is to visualize biological pathways derived from genome-scale models. In this case, the phosphoenolpyruvate metabolite was targeted to identify the chemical species in the extracellular compartment that produce it through

Table 1 **Number of entities and relationships loaded in Neo4j for the iAF1260 and iML1515 GEMs provided by the BiGG and MetaNetX databases.** Nodes are in lowercase, relationships are in uppercase.

	BiGG		MetaNetX	
Node/Relationship	**iAF1260**	**iML1515**	**iAF1260**	**iML1515**
Compartment	3	3	4	4
Species	1668	1877	1976	2217
Parameter	8	5	7	5
Reaction	2382	2712	2374	2704
Model	1	1	1	1
UnitDefinition	1	1	1	1
GeneProduct	1261	1516	1262	1517
IN_COMPARTMENT	1668	1877	1976	2217
HAS_PRODUCT	4714	5328	4920	5529
HAS_LOWERFLUXBOUND	2382	2712	2374	2704
HAS_UPPERFLUXBOUND	2382	2712	2374	2704
HAS_PARAMETER	8	5	7	5
HAS_UNIT	8	5	7	5
IS_IMPLIED	3747	4624	3750	4627
IS_REACTANT	4517	5247	4553	5242

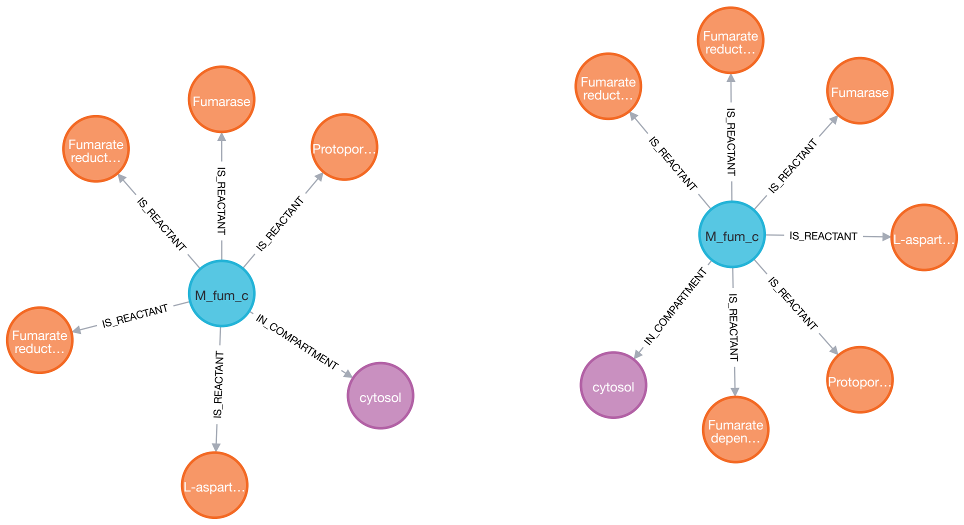

Figure 4 **Visualization of the fumarate metabolite (blue) and its nearest neighbors: reactions (orange) and compartment (purple) in two models (iAF1260 on the left and iML1515 on the right) sequentially imported into Neo4j by neo4jsbml.** The reaction, *Fumarate dependent DHORD*, was only shown in the iML1515 model.

Full-size DOI: 10.7717/peerj.16726/fig-4 exactly two reactions in the core model of *E. coli*. Firstly, the entire genome-scale model was imported into Neo4j.

```
MATCH (s:Species)
WITH s, size ([p=(s)-[:IS_SUBSTRATE]->() | p])
  AS sz ORDER BY sz DESC
WHERE sz > 10
SET s:Hub
```
(2)

```
MATCH path=(:Compartment)<-[:IN_COMPARTMENT]-(s1:Species {id: "M_pep_c"})
  <-[:HAS_PRODUCT|IS_SUBSTRATE*2]-(s2:Species)
  <-[:HAS_PRODUCT|IS_SUBSTRATE*2]-(s3:Species)
  -[:IN_COMPARTMENT]->(:Compartment)
WHERE
  NOT "Hub"
    IN apoc.coll.flatten([n in nodes(path) | labels(n)])
  AND (s3)-[:IN_COMPARTMENT]->(:Compartment {id: "e"})
  AND NOT (s2)-[:IN_COMPARTMENT]->(:Compartment {id: "e"})
RETURN path
```
(3)

Then, the reactions, metabolites, and compartments involved in the metabolic pathway were extracted using two consecutive Cypher queries: dense nodes were flagged Eq. (2) and the paths were selected Eq. (3). With the constraints defined above, the Pyruvate metabolite was identified as a key player (Fig. 5).

Enumerate metabolic pathways

The enumeration of metabolic pathways identifies and lists all the possible routes of biochemical reactions within a metabolic network. From the core model of *E. coli*, pathways involving the Formate and the Acetaldehyde metabolites were searched. *Species*, *SpeciesReference* and *Reaction* entities were loaded into Neo4j from the genome-scale model.

```
MATCH (n:SpeciesReference)-[:HAS_SPECIES]->(s:Species)
  SET n.id = s.id
MATCH (n:Species) DETACH DELETE n
MATCH (n:SpeciesReference) REMOVE n:SpeciesReference
  SET n:Species
```
(4)

```
MATCH (s:Species)
WITH s, size ([p=(s)-[:HAS_REACTANT]->() | p])
  AS sz ORDER BY sz DESC
WHERE sz > 4
SET s.hub = true
MATCH (s:Species)
WITH s, size ([p=(s)<-[:HAS_PRODUCT]-() | p])
  AS sz ORDER BY sz DESC
WHERE sz > 4
SET s.hub = true
MATCH (s:Species) WHERE s.hub DETACH DELETE s
```
(5)

(6)
```
CALL brsynth.enumeratePathway(["M_for_c", "M_acald_c"], "ep")
```

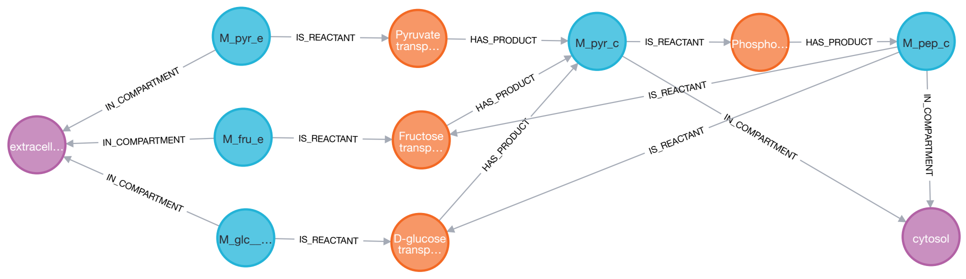

Figure 5 **Thanks to Neo4j and neo4jsbml, the phosphoenolpyruvate metabolite and its precursors implicated in exactly two reactions in the *Escherichia coli* core model (e_coli_core) were visualized.** The Pyruvate metabolite was directly involved in the production of Phosphoenolpyruvate without any circular dependency.

Full-size DOI: 10.7717/peerj.16726/fig-5

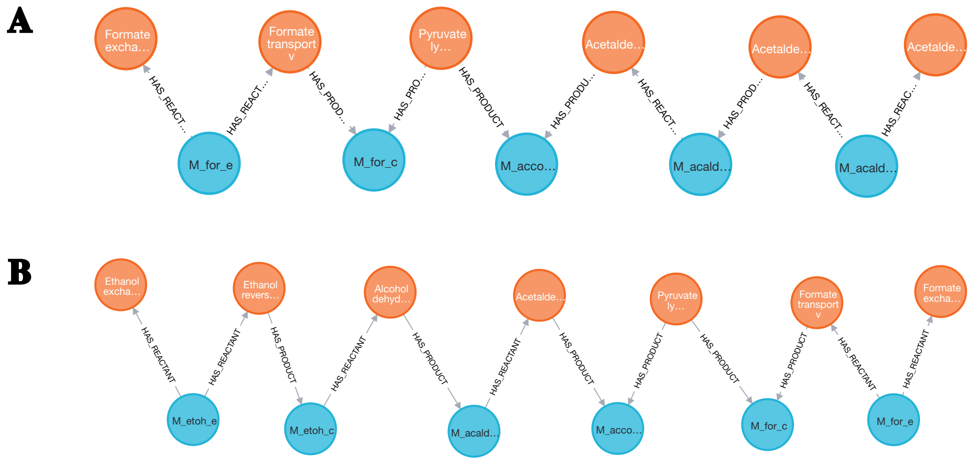

Figure 6 **Two pathways were identified in the *E. coli* core model.** (A) A first pathway linking the Formate to the Acetaldehyde metabolites was extracted. (B) A second pathway linking the Formate to the Ethanol metabolites was identified.

Full-size DOI: 10.7717/peerj.16726/fig-6

Then, graph reconstruction was performed by merging the *Species* and *SpeciesReference* nodes Eq. (4) and removing *Species* that were linked to fewer than four reactions Eq. (5). Next, the neo4jefmtool plugin was used to enumerate the pathways involving the Fumarate and the Acetaldehyde metabolites Eq. (6). The two pathways identified are shown in the Fig. 6.

DISCUSSIONS

Comparing genome-scale models, viewing metabolic networks and enumerating pathways are common applications in systems biology. On the one hand, biological processes are stored in the SBML format. On the other hand, Neo4j is a powerful graph database suited to link related entities, using the Cypher language. Neo4jsbml aims to conciliate both. The first use case showed the capability to perform quantitative and qualitative analysis on genome-scale models. Neo4j has a comprehensive set of features, as described by

King et al. (2015) which include the ability to navigate and search through visualizations, design and customize pathway maps, and represent diverse data types. The second example highlights the ability of Neo4j to serve as an alternative viewer for visualizing metabolic pathways. The last illustration emphasizes the capacity of neo4jsbml to retain the required data and the strength of Neo4j to conduct specialized analyses.

Neo4jsbml employs a novel approach based on the creation of schema entities to identify entities that are useful for a dedicated application. Introspection is a programmatic principle that maps the schema of certain entities to a model to extract desired data. Therefore, neo4jsbml can deal with Level 2 Version 5 of the SBML standard (Fig. S3) and partially with SBML packages (Fig. S2). Our aim was to develop a tool that is as simple as possible, requiring only command-line skills.

However, as we observed for the use cases, the more complex the application, the more intricate the Cypher queries to perform graph refactoring or to extract the data. Additionally, programming skills may be required to develop extensions to perform dedicated analyses. In rare circumstances, data imported into Neo4j can produce unexpected results. The first scenario is shown in Fig. S3J. The model has two *Event* entities, each with one *Event Assignment* entity that supports the *variable* property. Without setting an *id* property for each *Event Assignment*, in the model, it would be difficult to infer the wishes of the user. The second scenario concerns the model presented in Fig. S2C. The *Layout* entity is linked to its *Dimensions* entity as well as to all other *Dimensions* entities found in the model. In these cases, a prior modification of the model or graph refactoring in Neo4j is required to meet expectations. In addition, if several models are loaded sequentially into the database, some entities can be the same, and the relationships between the models are mixed. Neo4jsbml alleviates this difficulty by adding an additional identifier to all imported entities.

The SBML standard represents mathematical expressions using MathML and annotations using the Resource Description Framework by adding XML elements to the model. Neo4jsbml extracts mathematical expressions as strings whereas annotations are loaded with XML tags.

However, neo4jsbml is highly dependent on the web application Arrows to create the schema of entities. To ensure a long-term future, neo4jsbml will need to support at least one other tool to provide the schema of entities. In addition, neo4jsbml uses simple Cypher queries to import data into Neo4j. It would be valuable to load data into another graph database using Cypher.

More broadly, introspection-based programming patterns can be used to extract information from other standards and import the required data into sql, no-sql or graph-oriented databases.

CONCLUSIONS

Neo4jsbml is a user-friendly Python package and it takes advantage of existing software to define the database schema, parse SBML files, and import the data into the Neo4j database. The three use cases demonstrated the benefits of visualizing SBML data as a graph. The biological processes, specified thanks to SBML standard, contain intricate interactions between biological entities, and Neo4j's graph represents these relationships with nodes and edges, making it easier to analyze and visualize complex biological networks. Using Cypher, exploring data through statistics or creating subgraphs was intuitive. With the help of the schema database and introspection concept, only relevant information is loaded into the database. We foresee neo4jsbml as an essential tool for bridging the gap between SBML data and Neo4j. It will enable researchers to visualize and analyze relationships between different entities and fully leverage the capabilities of Neo4j.

ACKNOWLEDGEMENTS

We thank the CATI SysMics workgroup from INRAE. We thank the editor, Ron Henkel, and the anonymous reviewers for their valuable insights.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by a French government grant managed by the Agence Nationale de la Recherche under the France 2030 program, reference ANR-22-PEBB-0008. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
A French government grant managed by the Agence Nationale de la Recherche under the France 2030 program: ANR-22-PEBB-0008.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Guillaume Gricourt conceived and designed the experiments, performed the experiments, analyzed the data, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Thomas Duigou conceived and designed the experiments, performed the experiments, analyzed the data, authored or reviewed drafts of the article, and approved the final draft.
- Sandra Dérozier conceived and designed the experiments, performed the experiments, analyzed the data, authored or reviewed drafts of the article, and approved the final draft.
- Jean-Loup Faulon conceived and designed the experiments, analyzed the data, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The software is available at GitHub, Zenodo, and Anaconda:

- https://github.com/brsynth/neo4jsbml

- Gricourt, G. (2023). neo4jsbml dataset [Data set]. Zenodo. https://doi.org/10.5281/zenodo.10245426

- https://anaconda.org/conda-forge/neo4jsbml.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj.16726#supplemental-information.

REFERENCES

Balaur I, Mazein A, Saqi M, Lysenko A, Rawlings CJ, Auffray C. 2016. Recon2Neo4j: applying graph database technologies for managing comprehensive genome-scale networks. *Bioinformatics* **33(7)**:1096–1098 DOI 10.1093/bioinformatics/btw731.

Bornstein BJ, Keating SM, Jouraku A, Hucka M. 2008. LibSBML: an API Library for SBML. *Bioinformatics* **24(6)**:880–881 DOI 10.1093/bioinformatics/btn051.

Chaouiya C, Keating SM, Berenguier D, Naldi A, Thieffry D, Iersel MPV, Le Novère N, Helikar T. 2015. SBML level 3 package: qualitative models, version 1, release 1. *Journal of Integrative Bioinformatics* **12(2)**:691–730 DOI 10.1515/jib-2015-270.

Choi K, Medley JK, König M, Stocking K, Smith L, Gu S, Sauro HM. 2018. Tellurium: an extensible python-based modeling environment for systems and synthetic biology. *Biosystems* **171**:74–79 DOI 10.1016/j.biosystems.2018.07.006.

Clerx M, Cooling MT, Cooper J, Garny A, Moyle K, Nickerson DP, Nielsen PMF, Sorby H. 2020. CellML 2.0. *Journal of Integrative Bioinformatics* **17(2–3)**:20200021 DOI 10.1515/jib-2020-0021.

conda-forge community. 2015. The conda-forge project: community-based software distribution built on the conda package format and ecosystem. *Zenodo*. DOI 10.5281/ZENODO.4774216.

Feist AM, Henry CS, Reed JL, Krummenacker M, Joyce AR, Karp PD, Broadbelt LJ, Hatzimanikatis V, Palsson BO. 2007. A genome-scale metabolic reconstruction for *Escherichia coli* K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. *Molecular Systems Biology* **3(1)**:121 DOI 10.1038/msb4100155.

Gauges R, Rost U, Sahle S, Wengler K, Bergmann FT. 2015. The systems biology markup language (SBML) level 3 package: layout, version 1 core. *Journal of Integrative Bioinformatics* **12(2)**:550–602 DOI 10.1515/jib-2015-267.

Haiman ZB, Zielinski DC, Koike Y, Yurkovich JT, Palsson BO. 2021. MASSpy: building, simulating, and visualizing dynamic biological models in Python using mass action kinetics. *PLOS Computational Biology* **17(1)**:e1008208 DOI 10.1371/journal.pcbi.1008208.

Henkel R, Wolkenhauer O, Waltemath D. 2015. Combining computational models, semantic annotations and simulation experiments in a graph database. *Database* **2015**:bau130 DOI 10.1093/database/bau130.

Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U. 2006. COPASI—a COmplex PAthway SImulator. *Bioinformatics* **22**(**24**):3067–3074 DOI 10.1093/bioinformatics/btl485.

Hucka M, Bergmann FT, Chaouiya C, Dräger A, Hoops S, Keating SM, König M, Novère NL, Myers CJ, Olivier BG, Sahle S, Schaff JC, Sheriff R, Smith LP, Waltemath D, Wilkinson DJ, Zhang F. 2019. The systems biology markup language (SBML): language specification for level 3 version 2 core release 2. *Journal of Integrative Bioinformatics* **16**(**2**):20190021 DOI 10.1515/jib-2019-0021.

Hucka M, Smith LP. 2016. SBML level 3 package: groups, version 1 release 1. *Journal of Integrative Bioinformatics* **13**(**3**):8–29 DOI 10.1515/jib-2016-290.

Kazerouni AS, Gadde M, Gardner A, Hormuth DA, Jarrett AM, Johnson KE, Lima EAF, Lorenzo G, Phillips C, Brock A, Yankeelov TE. 2020. Integrating quantitative assays with biologically based mathematical modeling for predictive oncology. *IScience* **23**(**12**):101807 DOI 10.1016/j.isci.2020.101807.

King ZA, Drager A, Ebrahim A, Sonnenschein N, Lewis NE, Palsson BO. 2015. Escher: a web application for building, sharing, and embedding data rich visualizations of biological pathways. *PLOS Computational Biology* **11**(**8**):e1004321 DOI 10.1371/journal.pcbi.1004321.

King ZA, Lu J, Dräger A, Miller P, Federowicz S, Lerman JA, Ebrahim A, Palsson BO, Lewis NE. 2016. BiGG models: a platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research* **44**(**D1**):D515–D522 DOI 10.1093/nar/gkv1049.

Lambusch F, Waltemath D, Wolkenhauer O, Sandkuhl K, Rosenke C, Henkel R. 2018. Identifying frequent patterns in biochemical reaction networks: a workflow. *Database* **2018**:bay051 DOI 10.1093/database/bay051.

Meier A, Kaufmann M. 2019. Ensuring data consistency. In: *SQL & NoSQL databases*. Wiesbaden: Springer Fachmedien Wiesbaden, 123–142 DOI 10.1007/978-3-658-24549-8_4.

Monk JM, Lloyd CJ, Brunk E, Mih N, Sastry A, King Z, Takeuchi R, Nomura W, Zhang Z, Mori H, Feist AM, Palsson BO. 2017. iML1515, a knowledgebase that computes Escherichia coli traits. *Nature Biotechnology* **35**(**10**):904–908 DOI 10.1038/nbt.3956.

Moretti S, Tran V, Mehl F, Ibberson M, Pagni M. 2021. MetaNetX/MNXref: unified namespace for metabolites and biochemical reactions in the context of metabolic models. *Nucleic Acids Research* **49**(**D1**):D570–D574 DOI 10.1093/nar/gkaa992.

Olivier BG, Bergmann FT. 2018. SBML level 3 package: flux balance constraints version 2. *Journal of Integrative Bioinformatics* **15**(**1**):20170081 DOI 10.1515/jib-2017-0082.

Qiu S, Yang A, Zeng H. 2023. Flux balance analysis-based metabolic modeling of microbial secondary metabolism: current status and outlook. *PLOS Computational Biology* **19**(**8**):e1011391 DOI 10.1371/journal.pcbi.1011391.

Rodriguez N, Thomas A, Watanabe L, Vazirabad IY, Kofia V, Gómez HF, Mittag F, Matthes J, Rudolph J, Wrzodek F, Netz E, Diamantikos A, Eichner J, Keller R, Wrzodek C, Fröhlich S, Lewis NE, Myers CJ, Le Novère N, Palsson BO, Hucka M, Dräger A. 2015. JSBML 1.0: providing a smorgasbord of options to encode systems biology models. *Bioinformatics* **31(20)**:3383–3386 DOI 10.1093/bioinformatics/btv341.

Shen L, Kohlhaas M, Enoki J, Meier R, Schönenberger B, Wohlgemuth R, Kourist R, Niemeyer F, van Niekerk D, Bräsen C, Niemeyer J, Snoep J, Siebers B. 2020. A combined experimental and modelling approach for the Weimberg pathway optimisation. *Nature Communications* **11(1)**:1098 DOI 10.1038/s41467-020-14830-y.

Shin J, Porubsky V, Carothers J, Sauro HM. 2023. Standards, dissemination, and best practices in systems biology. *Current Opinion in Biotechnology* **81**:102922 DOI 10.1016/j.copbio.2023.102922.

Swainston N, Batista-Navarro R, Carbonell P, Dobson PD, Dunstan M, Jervis AJ, Vinaixa M, Williams AR, Ananiadou S, Faulon J-L, Mendes P, Kell DB, Scrutton NS, Breitling R. 2017. biochem4j: integrated and extensible biochemical knowledge through graph databases. *PLOS ONE* **12(7)**:e0179130 DOI 10.1371/journal.pone.0179130.

Terzer M, Stelling J. 2008. Large-scale computation of elementary flux modes with bit pattern trees. *Bioinformatics* **24(19)**:2229–2235 DOI 10.1093/bioinformatics/btn401.

Waltemath D, Adams R, Bergmann FT, Hucka M, Kolpakov F, Miller AK, Moraru II, Nickerson D, Sahle S, Snoep JL, Le Novère N. 2011. Reproducible computational biology experiments with SED-ML—the simulation experiment description markup language. *BMC Systems Biology* **5(1)**:198 DOI 10.1186/1752-0509-5-198.