PLOS COMPUTATIONAL BIOLOGY

SOFTWARE

SBMLNetwork: A framework for standards-based visualization of biochemical models

Adel Heydarabadipour[1], **Lucian Smith**[1], **Joseph L. Hellerstein**[1,2,3], **Herbert M. Sauro**[1,3]*

1 Department of Bioengineering, University of Washington, Seattle, Washington, United States of America, **2** Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, Washington, United States of America, **3** eScience Institute, University of Washington, Seattle, Washington, United States of America

* hsauro@uw.edu

OPEN ACCESS

Citation: Heydarabadipour A, Smith L, Hellerstein JL, Sauro HM (2025) SBMLNetwork: A framework for standards-based visualization of biochemical models. PLoS Comput Biol 21(9): e1013128. https://doi.org/10.1371/journal.pcbi.1013128

Editor: Marc R. Birtwistle, Clemson University, UNITED STATES OF AMERICA

Received: May 8, 2025

Accepted: September 5, 2025

Published: September 22, 2025

Data availability statement: SBMLNetwork's source code, documentation, and scripts used to generate all figures (including SBML files with embedded Layout and Render data) are publicly available under the MIT License at https://github.com/sys-bio/SBMLNetwork. A version of the repository has been archived on Zenodo and assigned the DOI: 10.5281/zenodo.16657743. The repository includes detailed build instructions, binary

Abstract

SBMLNetwork is an open-source software library that makes the SBML Layout and Render packages practical for standards-based visualization of biochemical models. Current tools often manage model visualization data in custom-designed, tool-specific formats and store it separately from the model itself, hindering interoperability, reproducibility, and the seamless integration of visualization with model data. SBMLNetwork addresses these limitations by building directly on the SBML Layout and Render specifications, automating the generation of standards-compliant visualization data, offering a modular implementation with broad integration support, and providing a robust API tailored to the needs of systems biology researchers. We illustrate the capabilities of SBMLNetwork across key visualization tasks, including SBGN-compliant visualization, application of predefined style templates, layout arrangement to reflect pathway logic, and integration of model data into network diagrams. These examples demonstrate how SBMLNetwork enables high-level visualization features and seamlessly translate user intent into reproducible outputs that support both structural representation and dynamic data visualization within the SBML model. SBMLNetwork is freely available at https://github.com/sys-bio/SBMLNetwork under the MIT license.

Author summary

We developed SBMLNetwork, a software tool that enables researchers to create standardized visualizations of biological models with minimal effort. Currently, scientists studying complex biological systems face the major challenge that different software tools store visualization data in incompatible formats. This makes it difficult to share model diagrams or reproduce them across platforms, creating unnecessary barriers to

releases for the C/C++ libraries, WebAssembly and JavaScript bindings, and usage examples.

Funding: This work was supported by the Center for Reproducible Biomedical Modeling, funded by the National Institutes of Health under award P41EB023912 with support provided to HMS, JLH, LM, and AH. HMS and AH also gratefully acknowledge generous support from National Institutes of Health award U24EB028887, which funded much of the early work. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

collaboration and slowing scientific progress. Our solution builds on an established standard format for visualizing biological models, while removing the technical complexity that has limited its widespread use. SBMLNetwork automatically generates diagrams of biological networks and stores all visualization details in the standard format alongside the model data in a single file. We demonstrated our tool's capabilities by producing clear, standards-based diagrams that accurately represent biological pathways and can incorporate relevant data. By making standards-based model diagrams easier to create and share, we aim to enhance reproducibility and accelerate the communication of biological models within the systems biology community.

Introduction

Visualization is an essential element in systems biology, particularly for interpreting complex biological models. It transforms multifaceted datasets—ranging from network topologies to simulation results—into clear, intuitive graphical representations that reveal the underlying interactions and dynamic behaviors of the models. By visually depicting biological models, researchers can deepen their understanding of biological systems through the observation of network topology, system dynamics, and biological responses. Beyond enhancing model comprehension, visualization also enables effective collaboration by allowing researchers to clearly communicate their insights and findings to a broader audience.

In recent years, rapid advances in high-throughput technologies have led to an exponential increase in the scale and complexity of biological data, shifting the research emphasis from data acquisition to data interpretation [1]. This increased focus has reemphasized the significance of data visualization and spurred the development of specialized tools and standards for the visualization of biological models. Notable examples include Escher, which enables rapid and semi-automated design as well as visualization of biological pathways and associated data [1]; CellDesigner, a tool that facilitates the construction of both gene-regulatory and biochemical networks [2]; MINERVA, a comprehensive platform for curating, annotating, and visualizing molecular interaction networks [3]; CySBML and cySBGN, which support the seamless import and management of SBML (Systems Biology Markup Language) [4] and SBGN (Systems Biological Graphical Notation) [5] formats in Cytoscape [6,7]; and Newt, designed for creating pathways in SBGN format [8]. However, the concurrent development of the tools and standards has introduced various data formats for storing visualization data, leading to challenges such as limited interoperability, cumbersome data exchange, and difficulties in reproducing model visuals across different platforms [9]. The emergence of these challenges underscores the importance of adopting existing standards for biological model visualization.

Among the developed standards for storing visualization data, SBML with Layout [10] and Render [11] packages stands alone in ensuring that all generated visualization data remain seamlessly exchangeable among different tools, reproducible in future analyses, and extensible for further developments, while also supporting both the positioning and graphical styling of model elements. It also provides explicit mapping between each visual element and its corresponding model component. This mapping enables straightforward cross-referencing between each model visual and its underlying model entity, a critical step in seamlessly integrating simulation and experimental data with the graphical representation of the model. In addition, it offers the functionality to store visualization data in the same file as the SBML model data. This simplifies model management by eliminating the need to manage multiple files and

reduces errors when linking corresponding elements to their graphical representations. However, despite its significant potential to serve as the de facto standard for model visualization, SBML with Layout and Render packages has remained underutilized. Its limited adoption can largely be attributed to the fact that the process for generating and editing visualization data for a biological model using this standard is very tedious-mainly because its steep learning curve demands a significant investment of time and technical expertise, and its API is not particularly intuitive for non-technical users.

SBMLDiagrams [12] was an important early effort to address some of these limitations by providing a higher-level Python interface to the Layout and Render specifications. However, it relies on a generic, non-domain-aware auto-layout engine that often requires substantial manual adjustment of visual elements. In addition, its single-language, lightweight wrapper architecture exposes only a subset of the Layout and Render primitives, limiting its reuse across different ecosystems and restricting fine-grained control over model visualizations for users.

In this paper, we present SBMLNetwork—a software library designed to make the SBML Layout and Render packages more accessible and practical for standard-based visualization of biochemical models. Unlike generic auto-layout methods that treat biochemical networks as simple node-edge graphs, SBMLNetwork employs a force-directed auto-layout algorithm enhanced with biochemistry-specific heuristics, where reactions are represented as hyperedges anchored to centroid nodes, species involved in many reactions are aliased to reduce visual clutter, and connections are drawn as role-aware Bézier curves that preserve reaction semantics while minimizing edge crossings. In addition, its multi-level API condenses frequent visualization tasks into single-line commands while still providing full, low-level access to Layout and Render primitives, enabling users to move seamlessly from high-level operations to meticulous, fine-grained customization. Moreover, unlike lightweight, single-language wrapper tools that limit extensibility, SBMLNetwork is built on a modular C/C++ core, exposed through a stable C API, with native bindings available in multiple languages. This design makes the library easily embeddable in third-party tools and a wide range of computational workflows.

Design and implementation

SBMLNetwork is built on a modular, layered architecture that organizes functionalities into discrete, interrelated levels. This design—illustrated in Fig 1—enables us to isolate key responsibilities such as standard compliance, input/output data management, core processing, cross-language integration, and user interaction, so that each module can be developed, maintained, and enhanced independently. This separation promotes robustness, scalability, and seamless interoperability across diverse platforms. In the following sections, we detail each layer, outlining its specific role, the technologies employed, and its interactions with adjacent layers.

Standards specification layer

At the core of our architectural design, is the SBML Layout and Render packages that provides a standardized visualization of reaction networks, expressed in biochemical terms. The SBML Layout package comprises specific layout elements that mirror the structure of the core SBML model and record the positions and sizes of its elements. It also allows a single model element to be represented by multiple layout elements (alias elements) [13,14] and supports cases where a layout element does not directly correspond to any model element, for example when depicting sources and sinks in a biological reaction. Complementary to the Layout package, the SBML Render package independently manages visual styling, such as colors, node shapes,

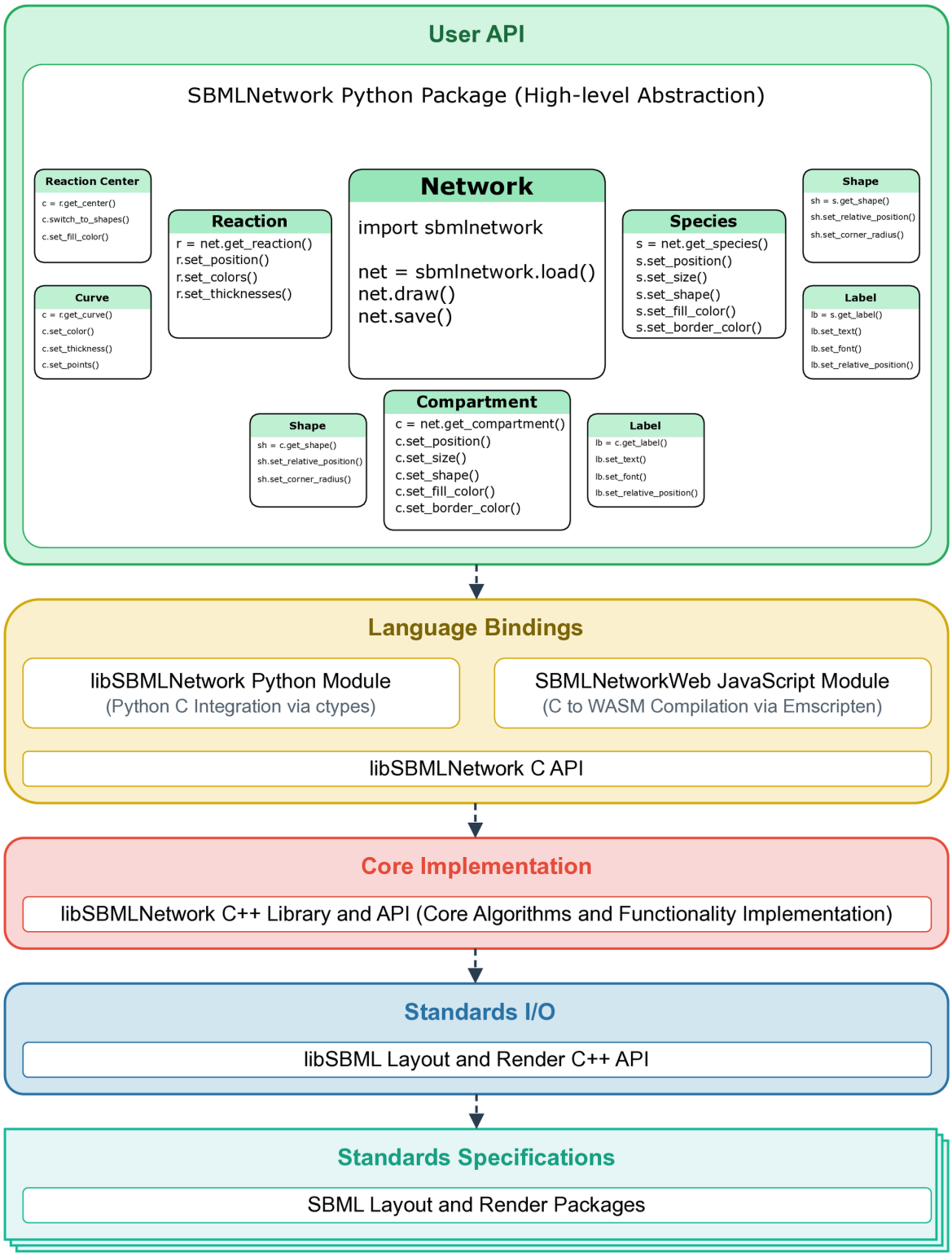

Fig 1. SBMLNetwork Layered Architecture. The figure illustrates the modular, multi-layered design of SBMLNetwork, which is organized into discrete levels responsible for standard compliance, I/O operations, core processing, cross-language integration, and user interaction. In particular, the User API Layer is outlined with its three sub-layers that offer a seamless progression from high-level operations to granular control over visualization details and provide support for both novice and expert users.

https://doi.org/10.1371/journal.pcbi.1013128.g001

line styles, and fonts, by organizing a set of styles that can be applied to layout elements. The style attributes in these sets are primarily derived from corresponding elements in the SVG specification [11]. Using the SBML Layout and Render packages, our system reliably supports both the positioning and visual styling of model elements, explicitly links each visual element with its corresponding model component, and stores visualization data together with the core SBML model data.

Standards I/O layer

Building on the foundation provided by the SBML Layout and Render packages, this layer ensures that our system's input and output operations strictly adhere to the standards defined by these packages. To achieve this, we leverage libSBML [15]—a mature, high-level API library with built-in support for the Layout and Render packages—to efficiently read, write, manipulate, and validate SBML Layout and Render data. This approach guarantees that our system seamlessly and reliably manages the visualization of biological models according to these established standards.

Core implementation layer

At the heart of our system lies the Core Implementation Layer, which encompasses the fundamental algorithms and functionalities required for the automated generation and management of SBML Layout and Render elements. This layer offers several key capabilities that work together to ensure that visualization data for each model element is generated and managed with high accuracy and reliability. Examples of these capabilities include: the automatic generation of Layout and Render elements for each model element in SBML models that lack such data; the creation of alias elements when a single model element requires multiple visual representations [13]; the automatic assignment of default or element-specific styles to each model element; providing users with access to a set of HTML colors (with corresponding hex codes) [16] and harmonic stylistic features for setting a consistent theme across the network; and a robust error logging system that captures function failures and provides diagnostic feedback, guiding users toward identifying and resolving issues encountered during operation.

Another key capability incorporated in this layer is the automatic initial arrangement of visual elements through an enhanced force-directed auto-layout algorithm. Our implementation adapts the classical Fruchterman–Reingold algorithm [17] and augments it with heuristics tailored to biochemical networks [18]. Unlike generic graph layout methods that treat biochemical networks as simple node–edge graphs, we implemented reaction-aware positioning, modeling each reaction as a hyper-edge with a dedicated centroid node. This approach faithfully represents the multi-reactant, multi-product reactions typical of biological models. To reduce visual clutter, the system automatically generates alias elements whenever a single species participates in numerous reactions [13], which mitigates edge crossings and improves readability. The algorithm also introduces specialized curve generation: Bézier curves are drawn from each species to its reaction centroid, where the inclination of each curve at the centroid is adjusted according to both the number of attached curves and their biochemical roles; when multiple reactions converge in the same species, the attachment points are redistributed around the species node to prevent congestion; and, for common UniUni (one reactant, one product) reactions, the curves are straightened into a direct reactant-to-product line to produce a clear and intuitive layout. Collectively, these enhancements deliver an initial layout that preserves biochemical semantics and remains clear and readable at scale, providing a reliable foundation for subsequent manual curation of the biological network.

Language bindings layer

This layer bridges the gap between the underlying C++ implementation and higher-level applications by exposing core functionalities via a robust C API and native interfaces. The provided C API abstracts the complexities of the core implementation and enables seamless integration with multiple programming languages, ensuring that the main capabilities of our system are reliably accessible regardless of the language used. Then, by leveraging the provided C API, we make our system accessible to various programming environments through native interfaces. For example, to offer our users a simple interface that facilitates incorporation into scientific computing and data analysis workflows, our Python bindings—developed using ctypes [19]—integrate the core C API with Python applications. Additionally, to meet the rising demand for web-based applications, we provide JavaScript bindings by compiling the core C/C++ source code into WebAssembly (WASM) [20] using Emscripten [21], which delivers our core functionality directly in the browser and enables seamless integration into web-based applications. This design approach not only promotes reusability but also simplifies the development and maintenance of language-specific bindings and further enhances our system's extensibility.

User API layer

Our User API Layer exposes the core functionalities of our system to end users by leveraging the Python bindings from the Language Bindings Layer and abstracting its intrinsic complexity into an intuitive API. Its design approach targets systems biology researchers as the primary users and aims to significantly reduce the learning curve for them to start using our system. As shown in Fig 1, this layer is organized into three sub-layers, each offering a different level of abstraction.

At the top-most sub-layer, called Network, users can perform essential operations such as loading an SBML model, rendering its figure, and saving the model along with added or modified visualization data. This sub-layer also provides highly abstracted visualization features critical for model analysis and interpretation, including seamless data integration, dynamic grouping of model elements based on stylistic attributes, and batch application of modifications across multiple elements.

The middle sub-layer is designed for power users who require direct access to the visual representations of the biological model's components, including Species, Reactions, and Compartments. With this layer, users can directly access network components and manipulate both their arrangement and stylistic attributes.

Finally, for expert users, the lowest sub-layer offers direct access to the graphical building blocks that constitute the model's visual representation, such as geometric shapes, curves, and labels. This granular level of control is essential for users who need to rigorously customize or extend the default visualization and manipulate every detail of the rendered network.

Additionally, this layer integrates skia-python [22]— a robust Python binding for the Skia 2D rendering engine—to deliver high-quality outputs of the visualized model in multiple formats, including JPEG, PNG, SVG, and PDF.

Overall, our layered architecture delivers a robust and flexible system for generating and managing visualization data for biological models by leveraging the SBML Layout and Render standards. Performance-critical operations are implemented in C++, while tools such as ctypes and Emscripten enable seamless interaction across diverse programming environments and enhance cross-platform compatibility. At the forefront of this architecture, the User API Layer plays a pivotal role. It is a carefully structured interface that abstracts the underlying complexity into an accessible, multi-tiered framework. By offering a seamless progression

from high-level operations to granular control over model elements and visualization details, the User API Layer meets the specific needs of both novice and expert users. Furthermore, the system's modular design allows each layer to be updated or replaced independently, which facilitates the integration of new language bindings without impacting core functionalities. Given the evolving nature of visualization requirements, this approach not only addresses current needs but also provides a solid foundation for future enhancements, ensuring that the system evolves alongside emerging technologies and user demands.

Results

In this section, we first benchmark SBMLNetwork's C++ auto-layout engine and then demonstrate how SBMLNetwork addresses a variety of high-demand visualization tasks for biological models by providing seamless support for SBGN-compliant representations, advanced styling capabilities, flexible network element arrangements, and direct data integration on top of the visualized network—all while preserving arrangement and styling information within the SBML file.

Auto-layout engine performance

To evaluate the performance of the auto-layout engine, we benchmarked SBMLNetwork's C++-based auto-layout algorithm against SBMLDiagrams, which uses the pure-Python NetworkX `spring_layout` force-directed algorithm [23]. By using the same underlying algorithm in both tools, we ensure that this comparison isolates implementation efficiency from methodological differences.

To compare runtime scaling, we generated synthetic SBML models on the fly with the number of species ranging from 20 to 2,000, while maintaining a fixed ratio of 4:1 between the number of species and the number of reactions. Each tool's auto-layout algorithm was run with identical settings, `50` relaxation iterations and no parallelism, on an Apple M3 Mac (8-core CPU: 4 performance and 4 efficiency cores, 16 GB RAM) running macOS 15.1.1. Each experiment was repeated `10` times, and the median wall-clock time was recorded for each model size.

Fig 2 compares the runtime scaling of SBMLNetwork's C++-based auto-layout engine with SBMLDiagrams' implementation of pure-Python NetworkX's `spring_layout` algorithm. The results indicate that SBMLNetwork's auto-layout engine is roughly **240×** faster than SBMLDiagrams for a small model with 20 species and 5 reactions, and retains a sizable lead as the size of the model grows, still running about **18×** faster for a network of 2,000 species and 500 reactions, which is beyond the scale of typical biochemical models.

Visualization of SBGN-compliant networks

SBGN [5] is a widely adopted standard for visualizing biological models, enabling the unambiguous representation of complex biological knowledge through a set of easily recognizable glyphs. By providing the following visualization features, SBMLNetwork supports the generation of SBGN-compliant visualizations: (i) support for multi-compartment models; (ii) the ability to assign multiple shapes and labels to each network element; (iii) support for visualizing reaction centers as distinct geometric shapes; (iv) the capability to render multi-segmented curves for reaction paths; (v) support for demonstrating empty species; and (vi) the flexibility to use various shapes for arrowheads on reaction curves. To demonstrate this capability, we recreated two SBGN-compliant networks, the repressilator and iNOS pathway

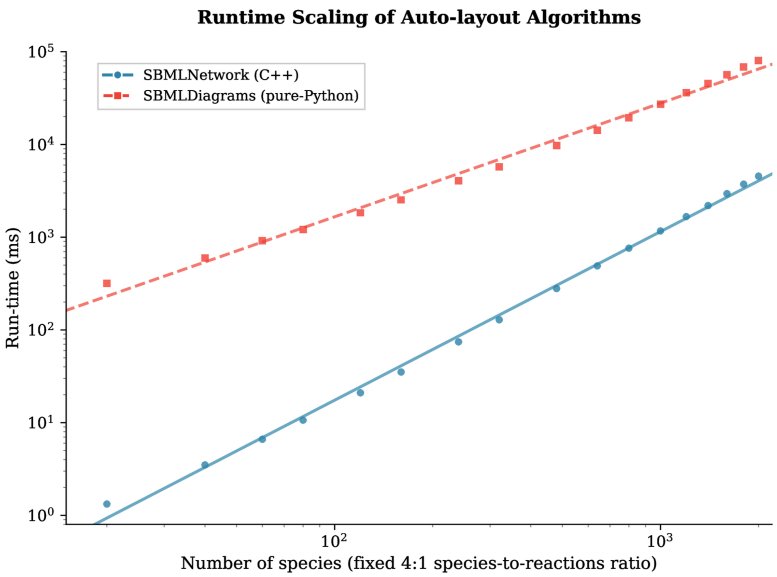

Fig 2. Performance Benchmark: SBMLNetwork vs. SBMLDiagrams Auto-layout. Log–log plot of median wall-clock time for SBMLNetwork's C++-based auto-layout engine (blue circles, solid fit) and SBMLDiagrams' implementation of the pure-Python NetworkX `spring_layout` algorithm (red squares, dashed fit), applied to synthetic SBML models containing 20–2,000 species, with a fixed 4:1 species-to-reaction ratio. Each point represents the median of `10` runs (error bars are smaller than the markers). SBMLNetwork achieves a ∼**240**× speed-up for the smallest model and maintains a ∼**18**× advantage at the model with 2,000 species.

https://doi.org/10.1371/journal.pcbi.1013128.g002

diagrams, from [24]. Figs 3 and 4 present each case as a side-by-side comparison of the original published pathway diagram and the version generated by SBMLNetwork. To create these renderings, we ensured SBGN compliance by creating a predefined styling object for each glyph defined in the SBGN Process-Description Reference Card [24]. Then each element in the SBGN-ML model [25] associated with each diagram was styled according to its `class` attribute, ensuring that all visual components follow the official SBGN specification. These results demonstrate that SBMLNetwork can faithfully and efficiently produce high-quality SBGN visualizations, enabling modelers to fully leverage the advantages of the SBGN stylistic format while storing visualization data alongside the core SBML model.

Visual styling

Visual styling is an effective technique for achieving aesthetically consistent and insightful visualizations of biological models. SBMLNetwork, with its comprehensive customization capabilities, enables users to apply a variety of styles to their models. To do so, SBMLNetwork provides a set of predefined style templates that users can select to alter the overall appearance of their network visualizations. Among these style templates is Escher [1]—a widely recognized scheme for depicting metabolic networks. The capability of our tool to apply the Escher style template is especially advantageous for users because the visualization data produced by tools that export to the Escher format omits explicit styling information. With support for the following features: (i) full control over the color and thickness of reaction curves; (ii) customization of font colors and sizes for network element labels; (iii) flexible configuration of multiple shapes for reaction centers; and (iv) configurable settings for arrowhead designs on reaction curves, SBMLNetwork is capable of setting the Escher style template on biological models stored in SBML format. To demonstrate this capability, as shown in Fig 5, we applied

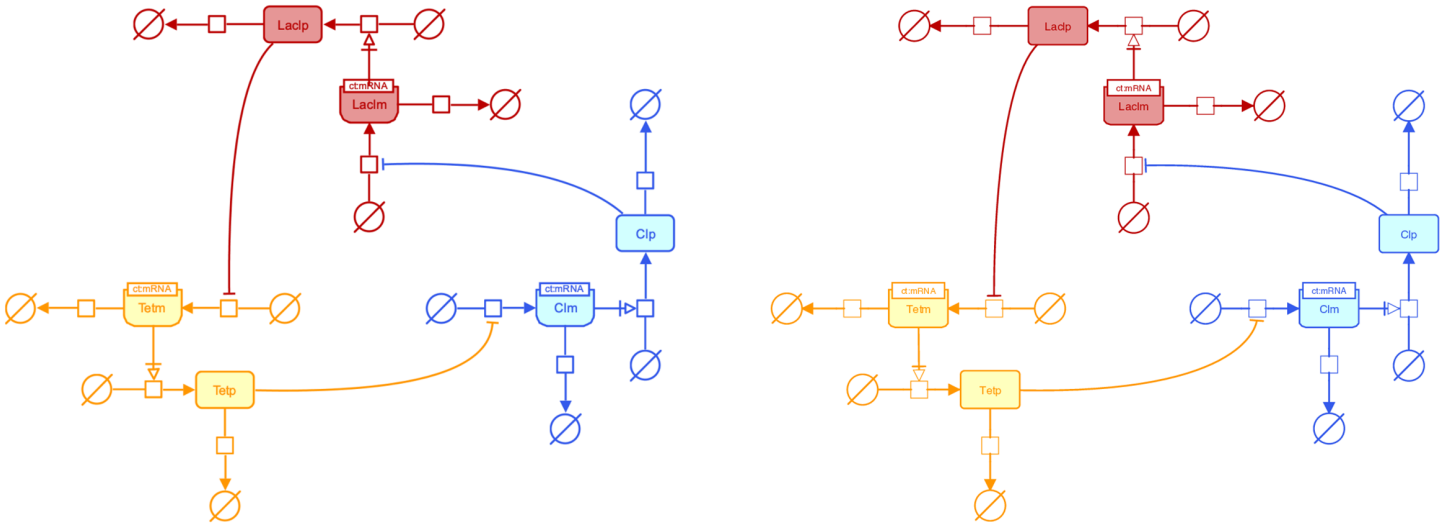

Fig 3. Original vs. SBMLNetwork-generated repressilator pathway in SBGN format. The left panel shows the original repressilator pathway from [24]; the right panel shows the same pathway rendered by SBMLNetwork, which embeds SBGN-compliant visualization data directly into the SBML model. This comparison highlights SBMLNetwork's capability to generate high-quality, SBGN-compliant visualizations for models featuring distinct geometric shapes for reaction centers, explicit depiction of empty species, and flexible arrowhead representations for reaction curves.

https://doi.org/10.1371/journal.pcbi.1013128.g003

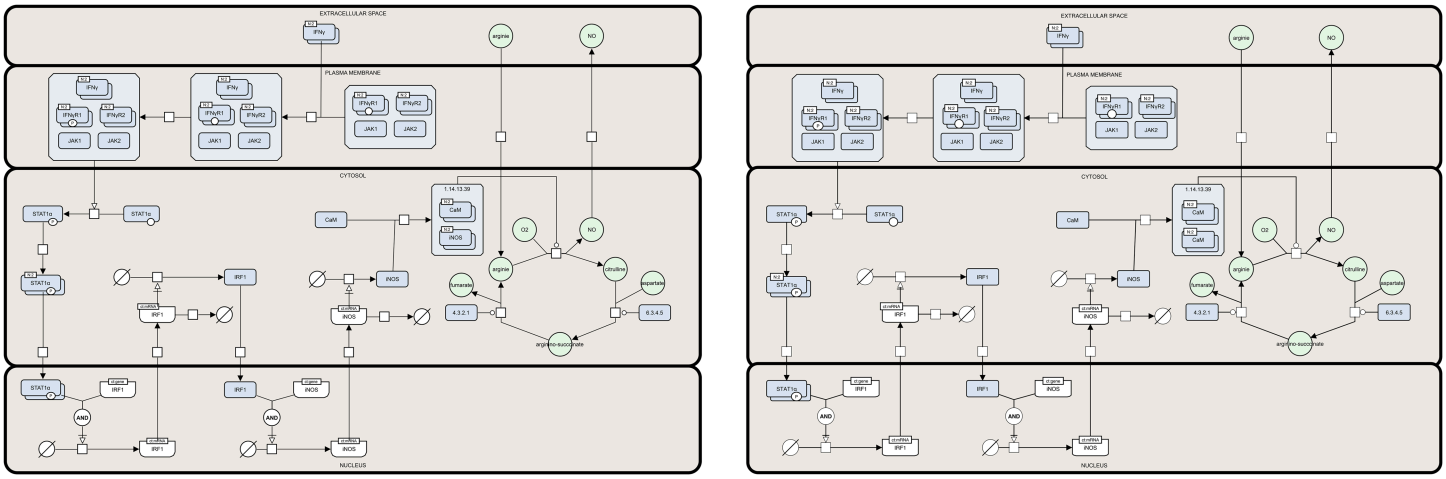

Fig 4. Original vs. SBMLNetwork-generated iNOS pathway in SBGN format. The left panel shows the original iNOS pathway from [24]; the right panel shows the same pathway rendered by SBMLNetwork, which embeds SBGN-compliant visualization data directly into the SBML model. This comparison highlights SBMLNetwork's capability to generate high-quality, SBGN-compliant visualizations for models with multiple compartments, multi-shaped and multi-labeled species, and multi-segmented reaction curves.

https://doi.org/10.1371/journal.pcbi.1013128.g004

the Escher style to the core *E. coli* metabolic model [26], where we first generated the SBML model with its Layout data using EscherConvertor [1], and then used SBMLNetwork to create Render data and apply the Escher style template on it. The resulting visualization demonstrates that SBMLNetwork effectively supports advanced visual styling for complex biological models, enabling modelers to achieve highly customized visualizations and maintain uniform styling across the entire model down to its finest details.

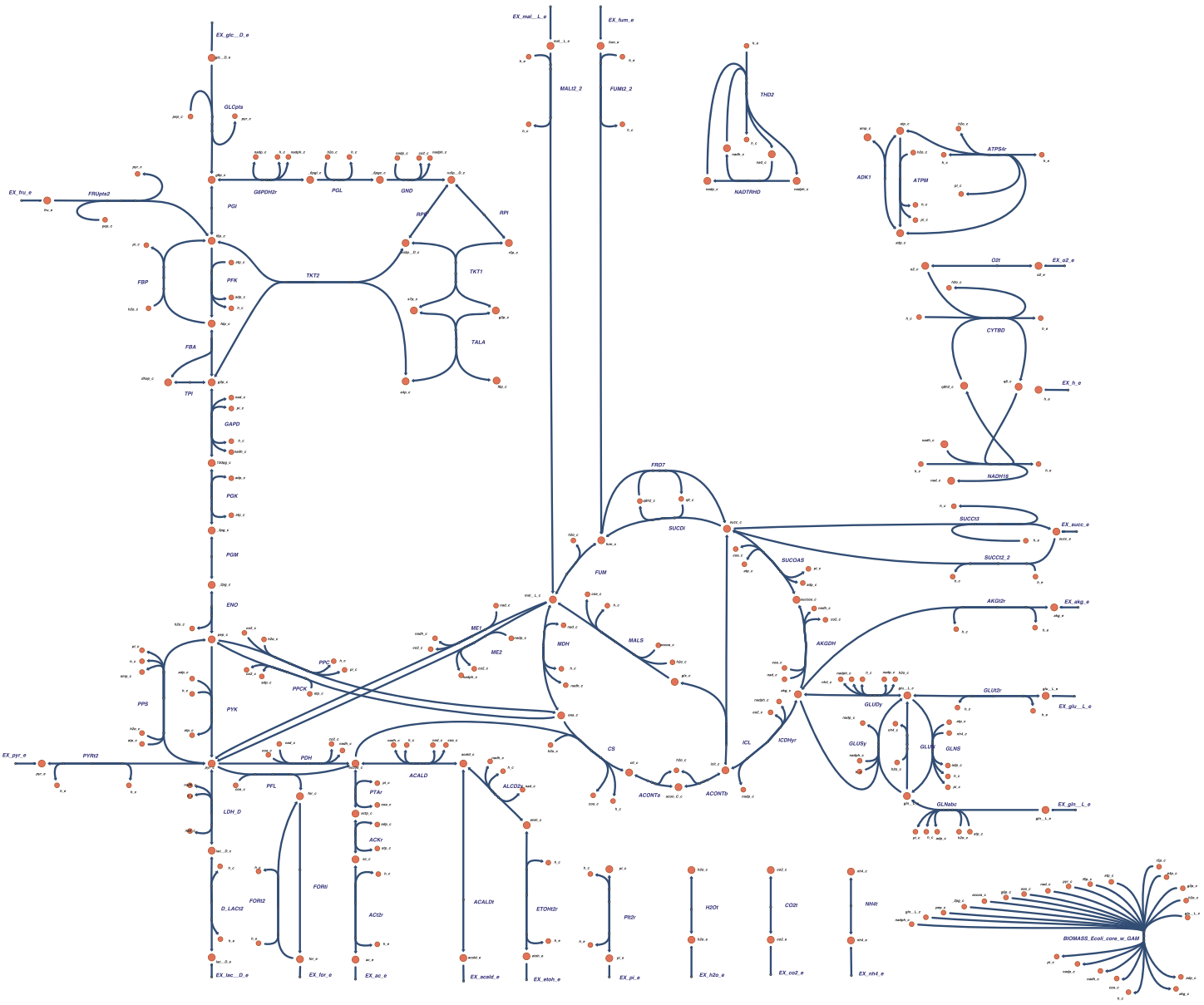

Fig 5. Applying Escher-Style Template on the Core E. coli Metabolic Model. This figure demonstrates the capability of SBMLNetwork to apply a style template to a complex biological model. The core *E. coli* metabolic model [26] was chosen, and its SBML Layout data was generated using EscherConvertor [1]. Subsequently, SBML-Network was employed to create SBML Render data and apply the well-known Escher style on the model. The resulting visualization showcases the advanced styling capabilities of SBMLNetwork, ensuring a uniform and fine-grained representation of complex biological networks in specific styles.

https://doi.org/10.1371/journal.pcbi.1013128.g005

Network arrangement

A well-organized layout is essential for conveying the logic of reactions in a biological model. By carefully arranging network elements, users can highlight relationships, directional flows, and hierarchical structures. Inspired by widely used pathway maps such as KEGG [27] and Escher [1] pathway maps, SBMLNetwork provides an advanced alignment feature that positions reactions horizontally, vertically, or along circular arcs.

We demonstrated this capability by visualizing the tricarboxylic acid (TCA) cycle. The Python script in S1 Listing is used to arrange the core TCA reactions in a circular layout and align the pyruvate dehydrogenase step vertically above the cycle. The resulting visualization, shown in Fig 6, highlights pathway flow and its cyclic nature. Notably, this precise arrangement is achieved with only a handful of SBMLNetwork commands, demonstrating that even complex positioning tasks can be carried out with minimal coding effort using the SBMLNetwork API. Moreover, because the alignment functionality can be applied to arbitrary subsets of reactions, it scales naturally to larger networks, enabling users to orient pathways and expose underlying biochemical logic with minimal effort.

Data integration

Effective visualization not only illustrates the structure of biological models but also integrates model data, enabling researchers to explore both network topology and dynamic behaviors. SBMLNetwork supports this integration by overlaying dynamic values—such as reaction fluxes and species concentrations—onto the visualized network through color-coded and size-coded mappings, providing an intuitive view of network activity over time. To ensure accurate and accessible displays, SBMLNetwork offers user-friendly helper functions, a fully customizable color bar for gradient encodings, and size-scaling utilities that map model data to network elements. Importantly, all visual elements related to data integration are seamlessly preserved alongside the model within the SBML Layout and Render specifications.

To demonstrate this feature, we used SBMLNetwork to render reaction flux data for the glycolysis pathway in a model of Saccharomyces cerevisiae [28]. In SBMLNetwork, flux values can be supplied in two ways: (i) by passing a single simulation-time point, which prompts SBMLNetwork to invoke its integrated simulation engine, Tellurium [29,30], and compute the corresponding flux magnitudes; or (ii) by providing an external mapping of SBML reaction identifiers to numeric flux values, which offers a simulator-agnostic alternative for data integration. In our demonstration, the flux magnitudes obtained via the first method were mapped to a gradient color scale—vivid hues for high flux and muted tones for low—and applied to the reaction curves, yielding the clear snapshot shown in Fig 7. SBMLNetwork offers the same workflow for mapping species concentrations to the fill color of species nodes in the network. In addition, it can encode reaction fluxes in the thickness of reaction curves and species concentrations in the size of species nodes. By integrating both reaction-flux and species-concentration data into network visualizations, SBMLNetwork provides a robust platform for dynamic exploration of biological models, ensuring that structural context and dynamic behavior are conveyed cohesively and with clear visual emphasis.

Discussion

In this work, we introduced SBMLNetwork, a software library that makes SBML's Layout and Render capabilities accessible for visualizing complex biological models. By building on SBML's Layout and Render packages and removing the technical hurdles that have limited their use, SBMLNetwork addresses three persistent challenges of the visualization of biochemical models: (i) poor interoperability across current visualization tools, (ii) visualization formats that cannot be seamlessly linked to their underlying model data, and (iii) the lack of a unified way to store model and visualization information together in a single file.

Building on this foundation, SBMLNetwork is architected as a multi-layered system. In the lower layers, performance-critical operations, including the force-directed auto-layout

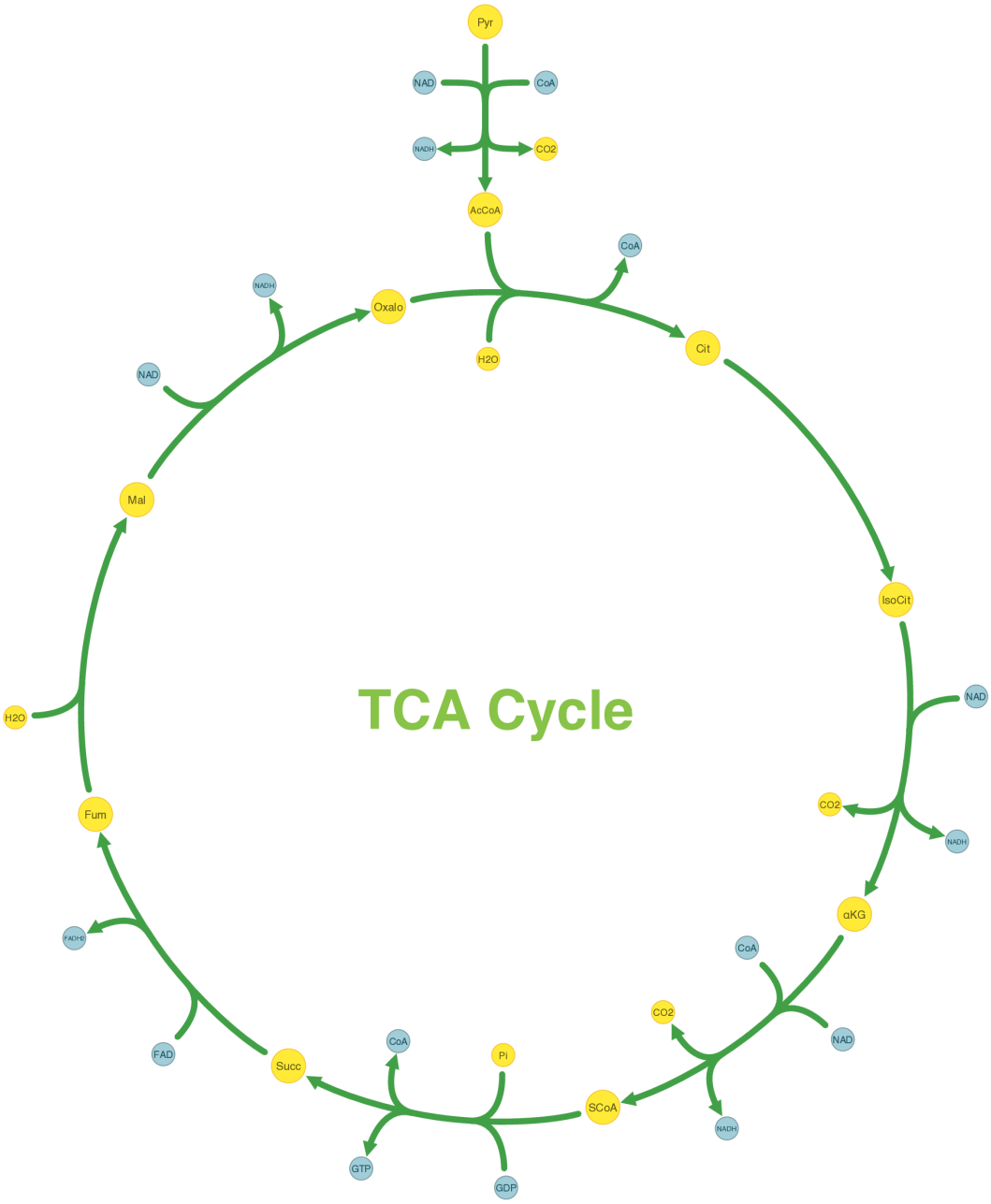

Fig 6. Reaction Alignment of the Tricarboxylic Acid (TCA) Cycle Network. Generated using the Python script in S1 Listing, SBMLNetwork's arrangement feature positioned the pyruvate dehydrogenase reaction vertically at the top and arranged the core TCA-cycle reactions in a circular layout to emphasize pathway flow and its cyclic nature. Metabolites are shown in yellow, cofactors in blue, and reactions in green to distinguish the different components of the pathway.

https://doi.org/10.1371/journal.pcbi.1013128.g006

algorithm, are implemented in C++ and ensure strict adherence to SBML Layout and Render standards. The intermediate layers facilitate adaptation and extension through language-specific bindings, with support provided for both Python and JavaScript. At its highest level, the user-facing APIs are tailored for non-technical users and high-demand use cases where an intuitive interface to the underlying functionalities is provided for end users.

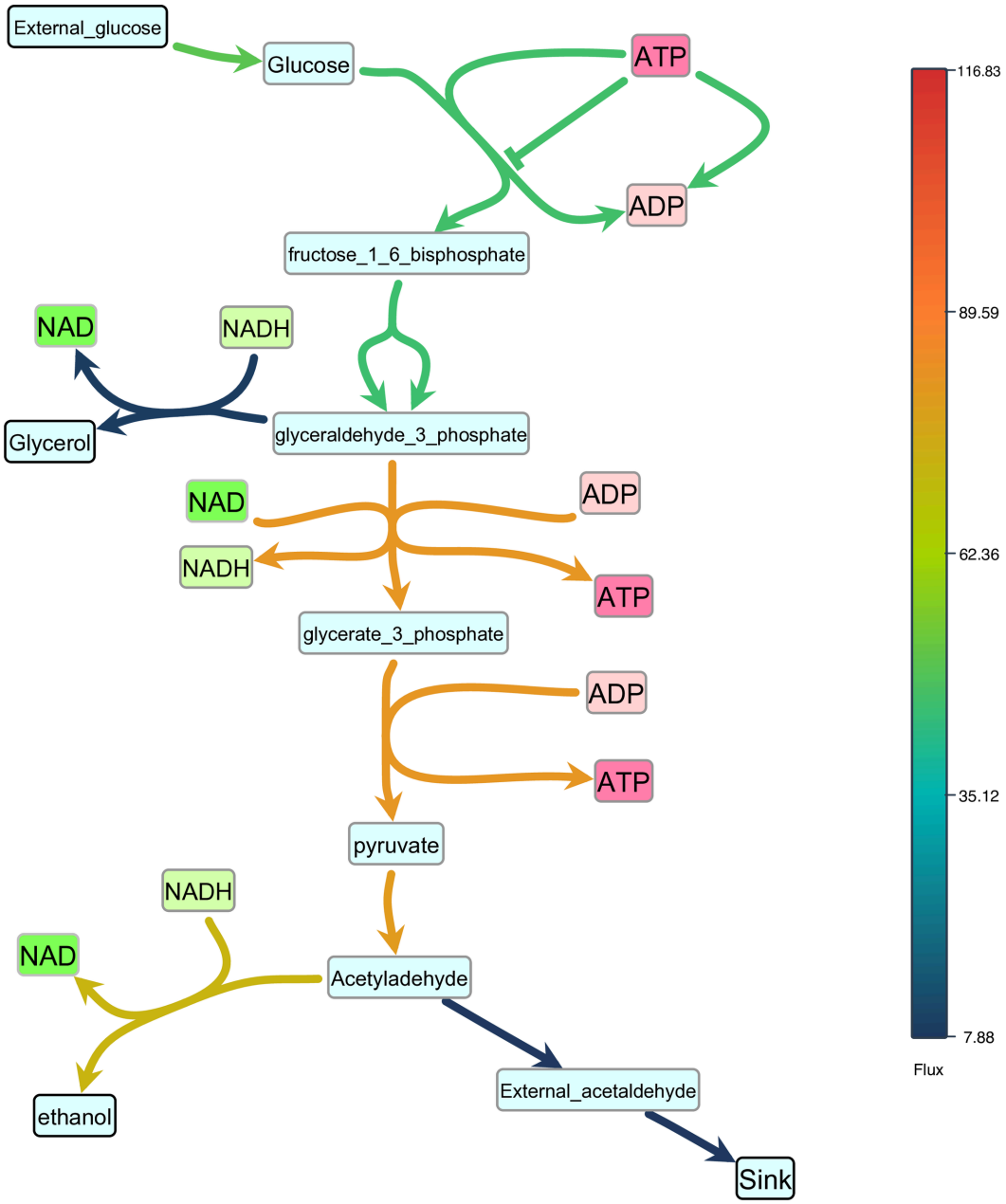

Fig 7. Color-encoded reaction fluxes in the *S. cerevisiae* glycolysis pathway. SBMLNetwork maps reaction-flux values from the glycolysis pathway in a model of Saccharomyces cerevisiae [28] to a continuous color gradient—vivid hues for high fluxes and muted tones for low, which offers a visual snapshot of pathway activity at a selected simulation time point.

https://doi.org/10.1371/journal.pcbi.1013128.g007

To assess the performance and scalability of SBMLNetwork's auto-layout engine, we benchmarked its C++-based auto-layout engine against a pure-Python implementation of the same algorithm across a wide range of synthetic model sizes. SBMLNetwork consistently generated layouts substantially faster, an advantage it retained even for the largest networks we tested, underscoring its efficiency as a platform for visualizing large-scale biological models.

To demonstrate SBMLNetwork's capability to address key visualization challenges in biological modeling, we applied the tool to several common use cases. For instance, our recreated repressilator and iNOS pathways diagrams demonstrate SBMLNetwork's ability to generate visualizations that adhere to the SBGN standard—a widely adopted approach for depicting biological models—while preserving every detail of the model's layout and styling information. These results show that embedding SBGN-style visualizations directly into the SBML file is both feasible and practical. A minor consideration is the mapping of certain niche SBGN glyphs, such as the "Annotation Node," which is not currently supported by the Render specification. It is expected that, with a wider adoption of tools like SBMLNetwork, such gaps can be revisited and potentially addressed in future updates to the standard. In another example, applying an Escher-style template to a model underscores that SBMLNetwork's advanced styling features can be leveraged to achieve the desired visual format with minimal effort, effectively compensating for the lack of built-in styling options in the available tools. Additionally, the network arrangement features—exemplified by the visualization of the TCA Cycle network—eliminate the tedious task of manually positioning reactions by aligning them in orientations that reveal the underlying biochemical logic with minimal coding effort. Finally, by overlaying simulation data—such as reaction fluxes encoded with gradient color mappings—onto network visualizations and including informative elements such as a fully customizable color bar, SBMLNetwork seamlessly integrates a model's structure with its dynamic behavior. Collectively, these examples demonstrate that SBMLNetwork meets high-demand visualization requirements and provides a robust platform for exploring and interpreting complex biological models.

In summary, SBMLNetwork addresses a number of critical visualization challenges in systems biology while unifying structural and stylistic data in a single file alongside the core SBML model. By lowering the technical barriers associated with using the SBML Layout and Render packages and providing a variety of user-focused functionalities, our tool facilitates reproducible, interoperable, and seamless model visualizations. Furthermore, SBMLNetwork lays a solid foundation for future developments in standards-based biological model visualization—an increasingly important component of large-scale, data-driven research in systems biology—owing to its highly extensible design that enables seamless integration with a wide range of tools and workflows.

Availability and future directions

SBMLNetwork is maintained as an open-source project on GitHub (https://github.com/sys-bio/SBMLNetwork), and is distributed under the MIT License. Detailed build instructions for compiling the binaries from source and guidance on how to link against its libraries are available for developers on its documentation page (https://sbmlnetwork.readthedocs.io/en/latest/).

The GitHub release page provides both static and dynamic binary distributions of the underlying C/C++ libraries, as well as the WebAssembly and JavaScript bindings. Additionally, SBMLNetwork is distributed as a Python package and can be installed from PyPI using the command `pip install sbmlnetwork`. The source code used to conduct the auto-layout performance benchmarks (Fig 2) and generate Figs 3–7, along with the output SBML file containing the visualization data in SBML Layout and Render format, is also available on the GitHub repository.

As we look ahead to the future of SBMLNetwork, its immense potential—built primarily upon its robust technical foundation—sets the stage for next-generation advancements in biological model visualization. This strong foundation is reflected in its robust C++ backend,

which provides the high-performance computations required for processing complex models; its multi-layered architecture, which offers outstanding modularity and scalability; and its integrated high-level language bindings, which facilitate seamless integration into a variety of workflows. Leveraging these foundational strengths, the following outline the prospective directions for further research:

- **High-Performance Visualization for Large-Scale Biological Models:** One promising future direction is to fully exploit SBMLNetwork's robust C++ backend for handling large-scale models. Its high-performance computational capacity offers a unique opportunity to efficiently visualize extensive models—such as those available in the BiGG Models database [31]. This direction should be actively pursued to leverage SBMLNetwork's powerful capabilities for efficiently scaling the visualization of complex biological models without compromising performance.
- **Next-Generation Simulation Data Integration:** Another promising future direction is to explore innovative methods for overlaying simulation data onto visualized models in SBMLNetwork. For example, recent advances in high-throughput technologies have led to a shift toward representing data as distributions rather than single values. Implementing this approach would enable more effective analysis of multi-omics datasets—capturing variations across different experimental conditions—and result in visualizations that are richer in detail and context [32]. Leveraging its user-friendly API, which provides seamless access to the intricate details of the network, along with its extensible design, SBMLNetwork offers substantial potential as a platform for exploring novel techniques for integrating simulation data onto the visualized model.
- **Rendering Gene Regulatory Networks**. Although SBMLNetwork is primarily geared towards displaying reaction networks, the Render standard is flexible enough to render other types of networks. We have already seen its use to render SBGN networks; however, for the future, one area that is currently poorly served, is rendering gene regulatory networks in a reproducible manner. One of the more useful styles for rendering gene regulatory networks is that generated by BioTapestry [33,34], and future work could develop a specific BioTapestry style.
- **Predefined Element Styles and Publication-Ready Templates**: Building on SBMLNetwork's demonstrated capabilities for visual styling, another compelling direction for future developments is to further expand its styling capabilities. One future enhancement involves creating predefined visual styles for individual network elements which allow users to employ more detailed and specialized representations with minimum effort rather than relying solely on basic geometric shapes. For instance, preconfigured skeletal formula depictions could be developed to illustrate carbon-based compounds, or off-the-shelf SBGN-compatible styles could be established for each type of network component. Another styling opportunity is to develop customized visualization templates for the entire network that conform to the formatting standards used by scientific journals. Such templates would significantly facilitate the creation of high-quality, publication-ready figures that are both easy to generate and highly reproducible.
- **AI-Driven Visualization Automation**: SBMLNetwork's fully scriptable API offers programmatic control over every layout and styling operation, making it an ideal platform for AI-driven visualization automation. Generative AI systems can implement complete visualization pipelines using SBMLNetwork which will produce fully reproducible model visualizations in SBML Layout and Render formats. In addition, machine-learning models trained on curated pathway maps—such as KEGG Pathways [27]—can deliver data-driven layout and styling recommendations through the same interface.

Together, these capabilities position SBMLNetwork at the forefront of AI-assisted biological network visualization tools, and we anticipate that its growing adoption inspires the community to develop increasingly adaptable, AI-powered workflows in the years ahead.

Supporting information

S1 Listing. TCA-cycle arrangement script.
(PDF)

Acknowledgments

AH acknowledges the contributions of Margaret Cook, Stella Anastasakis, Diego Alba Burbano, and Janis Shin, whose thorough testing of pre-release versions and thoughtful feedback significantly enhanced the robustness and reliability of this work. The contribution of Frank T. Bergmann in resolving compatibility issues with other tools is also gratefully acknowledged.

Author contributions

Conceptualization: Adel Heydarabadipour, Herbert M. Sauro.

Data curation: Adel Heydarabadipour.

Formal analysis: Adel Heydarabadipour, Herbert M. Sauro.

Funding acquisition: Herbert M. Sauro.

Investigation: Adel Heydarabadipour, Herbert M. Sauro.

Methodology: Adel Heydarabadipour, Lucian Smith, Herbert M. Sauro.

Project administration: Herbert M. Sauro.

Resources: Herbert M. Sauro.

Software: Adel Heydarabadipour.

Supervision: Joseph L. Hellerstein, Herbert M. Sauro.

Validation: Adel Heydarabadipour, Herbert M. Sauro.

Visualization: Adel Heydarabadipour.

Writing – original draft: Adel Heydarabadipour.

Writing – review & editing: Adel Heydarabadipour, Herbert M. Sauro.

References

1. King ZA, Dräger A, Ebrahim A, Sonnenschein N, Lewis NE, Palsson BO. Escher: a web application for building, sharing, and embedding data-rich visualizations of biological pathways. PLoS Comput Biol. 2015;11(8):e1004321. https://doi.org/10.1371/journal.pcbi.1004321 PMID: 26313928
2. Funahashi A, Morohashi M, Kitano H, Tanimura N. CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. Biosilico. 2003;1(5):159–62.
3. Gawron P, Ostaszewski M, Satagopam V, Gebel S, Mazein A, Kuzma M, et al. MINERVA-a platform for visualization and curation of molecular interaction networks. NPJ Syst Biol Appl. 2016;2:16020. https://doi.org/10.1038/npjsba.2016.20 PMID: 28725475

4. Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics. 2003;19(4):524–31. https://doi.org/10.1093/bioinformatics/btg015 PMID: 12611808
5. Le Novère N, Hucka M, Mi H, Moodie S, Schreiber F, Sorokin A, et al. The systems biology graphical notation. Nat Biotechnol. 2009;27(8):735–41. https://doi.org/10.1038/nbt.1558 PMID: 19668183
6. König M, Dräger A, Holzhütter HG. CySBML: a Cytoscape plugin for SBML. Bioinformatics. 2012;28(18):2402–3.
7. Gonçalves E, van Iersel M, Saez-Rodriguez J. CySBGN: a Cytoscape plug-in to integrate SBGN maps. BMC Bioinformatics. 2013;14:17. https://doi.org/10.1186/1471-2105-14-17 PMID: 23324051
8. Balci H, Siper MC, Saleh N, Safarli I, Roy L, Kilicarslan M, et al. Newt: a comprehensive web-based tool for viewing, constructing and analyzing biological maps. Bioinformatics. 2021;37(10):1475–7. https://doi.org/10.1093/bioinformatics/btaa850 PMID: 33010165
9. Hoksza D, Gawron P, Ostaszewski M, Hasenauer J, Schneider R. Closing the gap between formats for storing layout information in systems biology. Brief Bioinform. 2020;21(4):1249–60. https://doi.org/10.1093/bib/bbz067 PMID: 31273380
10. Gauges R, Rost U, Sahle S, Wengler K, Bergmann FT. The Systems Biology Markup Language (SBML) Level 3 Package: Layout, Version 1 Core. J Integr Bioinform. 2015;12(2):267. https://doi.org/10.2390/biecoll-jib-2015-267 PMID: 26528565
11. Bergmann FT, Keating SM, Gauges R, Sahle S, Wengler K. SBML Level 3 package: Render, Version 1, Release 1. J Integr Bioinform. 2018;15(1):20170078. https://doi.org/10.1515/jib-2017-0078 PMID: 29605822
12. Xu J, Jiang J, Sauro HM. SBMLDiagrams: a python package to process and visualize SBML layout and render. Bioinformatics. 2023;39(1):btac730. https://doi.org/10.1093/bioinformatics/btac730 PMID: 36370074
13. Heydarabadipour A, Sauro HM. Alias4SBML: a python package for generating alias nodes in sbml models. arXiv preprint 2025. https://arxiv.org/abs/2502.11318
14. Sauro HM, Hucka M, Finney A, Wellock C, Bolouri H, Doyle J, et al. Next generation simulation tools: the systems biology workbench and BioSPICE integration. OMICS. 2003;7(4):355–72. https://doi.org/10.1089/153623103322637670 PMID: 14683609
15. Bornstein BJ, Keating SM, Jouraku A, Hucka M. LibSBML: an API library for SBML. Bioinformatics. 2008;24(6):880–1. https://doi.org/10.1093/bioinformatics/btn051 PMID: 18252737
16. W3Schools. HTML Colors. W3Schools. [cited 2025 April 9]. https://www.w3schools.com/tags/ref_colornames.asp
17. Fruchterman TM, Reingold EM. Graph drawing by force-directed placement. Software: Practice and Experience. 1991;21(11):1129–64.
18. Deckard A, Bergmann FT, Sauro HM. Supporting the SBML layout extension. Bioinformatics. 2006;22(23):2966–7. https://doi.org/10.1093/bioinformatics/btl520 PMID: 17038346
19. Python. Ctypes – A foreign function library for Python. Python Documentation. [cited 2025 April 9]. https://docs.python.org/3/library/ctypes.html
20. WebAssembly. WebAssembly.org. [cited 2025 April 9]. https://webassembly.org/
21. Zakai A. Emscripten: an LLVM-to-JavaScript compiler. In: Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion. 2011. p. 301–12.
22. Skia Python. skia-python. [cited 2025 April 9]. https://kyamagu.github.io/skia-python/
23. Hagberg A, Swart PJ, Schult DA. Exploring network structure, dynamics, and function using NetworkX. Los Alamos, NM (United States): Los Alamos National Laboratory (LANL). 2008.
24. SBGN. Learning SBGN. [cited 2025 April 9]. https://sbgn.github.io/learning
25. Bergmann FT, Czauderna T, Dogrusoz U, Rougny A, Dräger A, Touré V, et al. Systems biology graphical notation markup language (SBGNML) version 0.3. J Integr Bioinform. 2020;17(2–3):20200016. https://doi.org/10.1515/jib-2020-0016 PMID: 32568733
26. Escher. Escher app for core metabolism of e_coli_core in viewer mode. [cited 2025 April 9]. https://escher.github.io/#/app?map=e_coli_core.Core
27. Kanehisa M, Furumichi M, Sato Y, Matsuura Y, Ishiguro-Watanabe M. KEGG: biological systems database as a model of the real world. Nucleic Acids Res. 2025;53(D1):D672–7. https://doi.org/10.1093/nar/gkae909 PMID: 39417505
28. Wolf J, Sohn H, Heinrich R, Kuriyama H. Mathematical analysis of a mechanism for autonomous metabolic oscillations in continuous culture of Saccharomyces cerevisiae. FEBS Lett. 2001;499(3):230–4. https://doi.org/10.1016/s0014-5793(01)02562-5 PMID: 11423122

29. Choi K, Medley JK, König M, Stocking K, Smith L, Gu S, et al. Tellurium: An extensible python-based modeling environment for systems and synthetic biology. Biosystems. 2018;171:74–9. https://doi.org/10.1016/j.biosystems.2018.07.006 PMID: 30053414
30. Smith LP, Bergmann FT, Chandran D, Sauro HM. Antimony: a modular model definition language. Bioinformatics. 2009;25(18):2452–4. https://doi.org/10.1093/bioinformatics/btp401 PMID: 19578039
31. King ZA, Lu J, Dräger A, Miller P, Federowicz S, Lerman JA, et al. BiGG models: a platform for integrating, standardizing and sharing genome-scale models. Nucleic Acids Res. 2016;44(D1):D515-22. https://doi.org/10.1093/nar/gkv1049 PMID: 26476456
32. Carrasco Muriel J, Cowie N, Taylor Parkins S, Mansouvar M, Groves T, Nielsen LK. Shu: visualization of high-dimensional biological pathways. Bioinformatics. 2024;40(3):btae140. https://doi.org/10.1093/bioinformatics/btae140 PMID: 38452346
33. Longabaugh WJR, Davidson EH, Bolouri H. Computational representation of developmental genetic regulatory networks. Dev Biol. 2005;283(1):1–16. https://doi.org/10.1016/j.ydbio.2005.04.023 PMID: 15907831
34. Longabaugh WJR, Davidson EH, Bolouri H. Visualization, documentation, analysis, and communication of large-scale gene regulatory networks. Biochim Biophys Acta. 2009;1789(4):363–74. https://doi.org/10.1016/j.bbagrm.2008.07.014 PMID: 18757046