


RESEARCH ARTICLE

HIF: The hypergraph interchange format for higher-order networks

Martín Coll¹, Cliff A. Joslyn^{2,3}, Nicholas W. Landry^{4,5,6} , Quintino Francesco Lotito^{7,8}, Audun Myers², Joshua Pickard⁹, Brenda Praggastis² and Przemysław Szufel¹⁰

¹Department of Computer Science, University of Buenos Aires, CABA, Argentina, ²Pacific Northwest National Laboratory, Seattle, WA, USA, ³Department of Systems Science and Industrial Engineering, Binghamton University, Binghamton, NY, USA, ⁴Department of Biology, University of Virginia, Charlottesville, VA, USA, ⁵School of Data Science, University of Virginia, Charlottesville, VA, USA, ⁶Vermont Complex Systems Institute, University of Vermont, Burlington, VT, USA, ⁷Department of Information Engineering and Computer Science, University of Trento, Trento, Italy, ⁸Department of Network and Data Science, Central European University, Vienna, Austria, ⁹Eric and Wendy Schmidt Center at the Broad Institute of MIT and Harvard, Cambridge, MA, USA, and ¹⁰SGH Warsaw School of Economics, Warsaw, Poland

Corresponding author: Nicholas W. Landry; Email: nicholas.landry@virginia.edu

Abstract

Many empirical systems contain complex interactions of arbitrary size, representing, for example, chemical reactions, social groups, co-authorship relationships, and ecological dependencies. These interactions are known as higher-order interactions, and the collection of these interactions comprise a higher-order network, or hypergraph. Hypergraphs have established themselves as a popular and versatile mathematical representation of such systems, and a number of software packages written in various programming languages have been designed to analyze these networks. However, the ecosystem of higher-order network analysis software is fragmented due to specialization of each software's programming interface and compatible data representations. To enable seamless data exchange between higher-order network analysis software packages, we introduce the Hypergraph Interchange Format (HIF), a standardized format for storing higher-order network data. HIF supports multiple types of higher-order networks, including undirected hypergraphs, directed hypergraphs, and abstract simplicial complexes, while actively exploring extensions to represent multiplex hypergraphs, temporal hypergraphs, and ordered hypergraphs. To accommodate the wide variety of metadata used in different contexts, HIF also includes support for attributes associated with nodes, edges, and incidences. This initiative is a collaborative effort involving authors, maintainers, and contributors from prominent hypergraph software packages. This project introduces a JSON schema with corresponding documentation and unit tests, example HIF-compliant datasets, and tutorials demonstrating the use of HIF with several popular higher-order network analysis software packages.

Keywords: network; higher-order; hypergraph; directed hypergraph; simplicial complex; data standard

1. Introduction

The growth of network science as a field has been fueled by the availability of large-scale datasets from a variety of sources, including the internet (Albert et al., 1999), co-authorship networks (Barabási et al., 2002; Newman, 2001), global-scale transportation (Brockmann and Helbing, 2013), social interactions (adams, 2019; Cattuto et al., 2010; Stehlé et al., 2011), biological processes (Barabási and Oltvai, 2004; Jeong et al., 2000), social media (Jackson et al., 2020; Morstatter et al., 2013; Ugander et al., 2012), and brain connectivity (Sporns et al., 2004), among others. As the plethora of data has grown, standardized formats for pairwise network datasets have been

developed, such as GraphML (Brandes et al., 2010) and Pajek (Nooy et al., 2018) as well as network dataset repositories such as SNAP (Leskovec and Krevl, 2014), Netzschleuder (Peixoto, 2021), ICON (Clauset et al., 2016), and Konect (Kunegis, 2013). These advances, however, have been developed with networks primarily in mind, which solely model dyadic interactions.

Higher-order networks generalize the notion of networks, representing interactions of arbitrary size. These networks can more naturally model certain empirical systems such as co-authorship networks, chemical reactions, human social networks, and biological interactions, which abound in multi-way interactions. This can be useful for studying nonlinear dynamical processes (Neuhäuser et al., 2020), the evolution of groups (Iacopini et al., 2024), and protein-protein interactions (Murgas et al., 2022), for example. There are many robust software packages for storing, analyzing, and visualizing higher-order networks (Badie-Modiri and Kivelä, 2023; Diaz and Stumpf, 2022; Failla et al., 2023; Hajij et al., 2024; Landry et al., 2023; Lotito et al., 2023; Pickard et al., 2023; Praggastis et al., 2024; Spagnuolo et al., 2020), but no corresponding data standards for facilitating cross-package compatibility. In this paper, co-authored by leaders and representatives of a number of these software packages, we explain the need for such a standard and the context in which it fits; introduce the Hypergraph Interchange Format (HIF) as a data standard for higher-order networks; describe integration with existing higher-order network software libraries; and lastly, demonstrate how it can be used to unlock the strengths of different higher-order network software packages.

1.1 Existing higher-order network data

Higher-order network data remains relatively scattered, with multiple versions of the same datasets often distributed across several repositories and lacking a consistent format. For example, the webpage curated by Austin Benson (Benson, 2021) features many datasets, primarily from the computer science domain, but is no longer actively maintained. There exist several actively maintained data repositories: XGI-DATA (Landry et al., 2023), which hosts many datasets on Zenodo¹ and is integrated with the XGI software API; HypergraphRepository (Antelmi et al., 2024), which offers a collection of empirical hypergraph datasets along with a web application for filtering and downloading them; and the Hypergraphx (Lotito et al., 2023) software library, which is accompanied by a diverse collection of datasets spanning multiple domains and hypergraph types (HGXTTeam, 2025). Additionally, higher-order network datasets can sometimes be found in traditional network repositories such as SNAP (Leskovec and Krevl, 2014), ICON (Clauset et al., 2016), and Netzschleuder (Peixoto, 2021), typically represented as bipartite graphs (see discussion in Section 2 below). Notably, there may be repetitions and overlapping datasets across these repositories, resulting in a redundant and fragile ecosystem. Furthermore, each repository currently uses its own proprietary format, which is not directly interoperable with other software libraries.

As research and development in hypergraph-based methods expand across diverse domains such as data mining, computational biology, and network analysis, the absence of a common data interchange format has become an obstacle to interoperability and reproducibility. Most existing libraries and tools rely on ad hoc representations of hypergraphs, making it difficult to share data, compare results, or construct integrated workflows. In practical applications, hypergraphs typically pass through multiple stages of analysis, such as construction, transformation, and visualization, with each stage potentially handled by a different tool. Without a standardized format, these transitions require custom conversion scripts and increase the risk of semantic loss or data misinterpretation.

1.2 Contributions

HIF addresses this challenge by offering a consistent, expressive, and extensible JSON-based schema for encoding both the topology and metadata of hypergraphs. Its language-agnostic design

enables seamless exchange between libraries implemented in Python, C++, Julia, JavaScript, and other environments, promoting modular, interoperable workflows while reducing redundant engineering effort. HIF serves as a unifying layer for the higher-order network science ecosystem, allowing software tools and pipelines to communicate through a shared representation. This facilitates reproducible research and fosters a more cohesive, collaborative development environment, similar to the role of the Open Neural Network Exchange (ONNX) format in machine learning (ONNX Community, 2024) and GraphML in traditional graph processing (Brandes et al., 2010).

2. Higher-order networks as hypergraphs

Network science is grounded on the mathematical representation of networks as undirected or directed graphs. Similarly, the term “higher-order network” has become associated with a variety of data objects related to undirected and directed hypergraphs. The literature on hypergraphs is extensive, and we refer the reader to two classic publications about hypergraphs (Berge, 1989; Bretto, 2013) as well as more recent treatments of the larger space of higher-order networks (Battiston et al., 2020; Bick et al., 2023; Joslyn et al., 2021; Torres et al., 2021).

The foundational mathematics of hypergraphs continues to evolve, including in a category-theoretical framework (Grilliette and Rusnak, 2023), producing multiple ways to axiomatize hypergraph formalisms. Our approach for this paper is to seek an appropriate general grounding sufficient to accommodate the widest range of both current hypergraph modeling approaches and current data science needs, while avoiding some of the more subtle complexities which can arise especially around edge cases like empty and duplicate hyperedges, isolated and redundant vertices, empty hypergraphs, self-loops, and related things. To that end, we now introduce just a few concepts of the most direct relevance to HIF formally, while noting both similarities and differences with some of the most widely circulated concepts as used in the literature.

Thus for our purposes, a **hypergraph** is a system $H = (V, E, I)$ where $V = \{v\}$ is a finite, non-empty set of **vertices** or **nodes**, $E = \{e\}$ is a finite, non-empty set of **edges** or **hyperedges**,² and $I \subseteq V \times E$ is a set of **incidences**, that is, pairs (v, e) of nodes and edges.

It may be valuable to understand how this definition may differ from some readers' expectations. First, note that an edge $e \in E$ can be mapped to the collection of vertices with which it has an incidence: $e \mapsto \{v \in V : (v, e) \in I\}$. With this in mind, we will also feel free to simply think of an edge as a subset of vertices and notate $e \subseteq V$, thereby recovering the traditional sense of a hyperedge as a collection of vertices of arbitrary size. And indeed, we could also notate $H = (V, E)$ and simply understand that each edge $e \in E$ is a subset $e \subseteq V$. But what's critical to note is that then this E is not actually a set, but rather, using our definition E would be a multiset or bag of hyperedges, and so possibly containing duplicates: there can be two distinct edges $e, f \in E$ which map to the same set of vertices, so that $e = f$. So while mathematically, our structure may be properly called a **multi-hypergraph**, for this paper, we will simply call these all hypergraphs, while still keeping this issue firmly in mind.

Rather more well understood is how any particular edge is a subset $e \subseteq V$ of arbitrary size. If e maps to only two vertices, then that e is a traditional (undirected) graph edge; and indeed, if this is true for *all* edges $e \in E$, then H is called a **2-uniform hypergraph**, which is just a **graph**: all graphs are hypergraphs. But e may also have more than two nodes (the traditional sense of “higher-order”), one node (singleton hyperedges), or even none (empty hyperedges are permitted).

HIF supports **directed hypergraphs** (Ausiello et al., 2001; Gallo et al., 1993), which comprise directed hyperedges $e \in E$. Each directed hyperedge is an ordered pair $e = (t, h)$ of subsets of vertices: the “tail” $t \subseteq e$ specifies the inputs in the multi-way interaction, and the “head” $h \subseteq e$, specifies the outputs. While the tail and head cover the hyperedge in that $t \cup h = e$, in contrast to some researchers (Jost and Mulas, 2019) who require that the head and tail be disjoint, here we generalize this framework and allow nodes to belong to both the tail and head.

Another prominent structure in higher-order network theory is that of an **abstract simplicial complex (ASC)**. An ASC is a hypergraph, but one which is “downward closed” in that for every hyperedge $e \in E$, then all nonempty collections of vertices $f \subseteq e$ contained in e are also in the hypergraph: $f \in E$. ASCs are used extensively in higher-order network science (Joslyn *et al.* (2021)), for example, when we seek to analyze the topological properties of hypergraphs mathematically as structures with multiple multi-dimensional components glued together, or when semantically it is understood in some application that when a relationship holds for a set of entities, it also holds for all its subsets.

Essential to HIF is that it supports **attributed hypergraphs**: nodes $v \in V$ and edges $e \in E$ can have arbitrary properties. But of great significance is that in addition, HIF also supports properties on the *incidences* $(v, e) \in I$. In other words, a node $v \in V$ can have one property when associated with an incidence (v, e) involving one edge $e \in E$ in which it is contained ($v \in e$), but could have quite a different property when associated with a different incidence (v, f) for a different edge $f \in E$, for which also $v \in f$. An incidence property can be used to represent directed hyperedges, in that for any particular hyperedge $e \in E$ and vertex $v \in e$, an incidence property can be used to encode whether v is in the tail $v \in t$, the head $v \in h$, or in both. Incidence properties are also essential for supporting critical features like “edge-dependent weights” (Chitra and Raphael, 2019) used in, e.g., random walk models on hypergraphs (Hayashi *et al.*, 2020).

It is also worth noting a few additional properties of hypergraphs and mathematical structures related to hypergraphs. While these aren’t represented in HIF proper, they are important for applications and understanding, and are mentioned elsewhere in this paper in the context of their implementation in some of the hypergraph software systems supporting HIF.

First, the incidence relation $I \subseteq E \times V$ is commonly represented as a Boolean **incidence matrix**, sometimes also notated (perhaps abusively) as just I , with $|V|$ rows and $|E|$ columns, where $I[i, j] = 1$ when $(v, e) \in I$ and 0 when $(v, e) \notin I$. Matrix operations on I can be very useful in applications, as we will see below. Additionally, every hypergraph $H = (V, E, I)$ has an equivalent **dual** form $H^* = (E, V, I^{-1})$ where nodes and edges are swapped, and I^{-1} is the inverse of the incidence relation, yielding the dual incidence matrix I^T as the transpose.

Every hypergraph H is also equivalently represented as a **bipartite graph** $B = (V \sqcup E, A)$ on a new set of vertices as the disjoint union of the nodes V and hyperedges E of the original hypergraph, and with a new set of graph edges $A \subseteq \binom{V \sqcup E}{2}$, but with the bipartite condition that for all edges $\{x, y\} \in A$, exactly one of x or y are in V , with the other in E . Undirected hypergraphs yield undirected bipartite graphs; directed hypergraphs yield directed bipartite graphs; and vice versa in both cases. While hypergraphs are bijective with bipartite graphs, they can represent different concepts and functions, and can be amenable to different algorithms: hypergraphs are about connectivity and path following, where bipartite graphs are about combinatoric and matching questions. And from a mathematical perspective, they may also have different transformations which yield them in different categories (Grilliette and Rusnak (2023)).

Hypergraphs operate in a broader mathematical ecosystem which also includes graphs. We have already seen how every graph is a 2-uniform hypergraph. Additionally, given a hypergraph $H = (V, E, I)$, it is common to construct a graph (V, C) on its vertex set, called a **clique expansion**, **2-section**, or **underlying graph**. Here C is the union of the complete graphs (cliques) formed on each hyperedge $e \in E$. Thereby C consists of all the pairs of vertices (graph edges on V) included in some hyperedge, such that for all $\{u, v\} \in C$, $\exists e \in E$, $\{u, v\} \subseteq e$. Note that the adjacency matrix of the 2-section is given by $I \times I^T$ as a matrix operation. Finally, it is also common to build the **line graph** of a hypergraph $H = (V, E, I)$ as a graph (E, L) on the edges taken as vertices, where a pair of edges $\{e, f\}$ is included in L if they intersect: $L = \{e, f \in E : e \cap f \neq \emptyset\}$. The line graph structure allows reasoning about the relationships between the hyperedges, as opposed to the vertices, and is thus closely related to the dual hypergraph H^* : indeed, the line graph of H is just the 2-section of the dual H^* , and vice versa, where now the adjacency matrix of the line graph is $I^T \times I$.

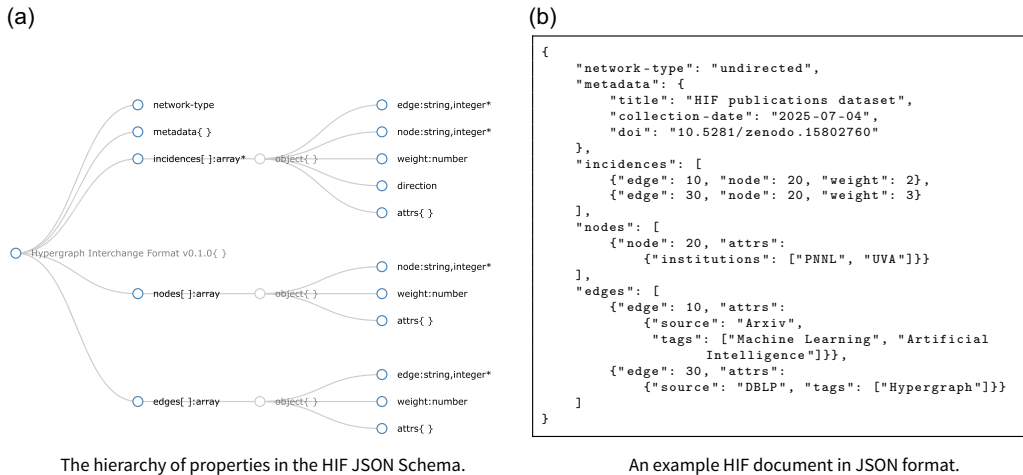


Figure 1. The JSON schema is depicted as a graph, showing the hierarchy of properties for each data record. The example demonstrates the use of most properties, including metadata for interpretability.

3. The hypergraph interchange format

The Hypergraph Interchange Format is a standard for storing higher-order networks in JavaScript Object Notation (JSON) format. JSON is chosen for its interpretability; while there are more efficient formats such as Parquet, CSV, GraphQL, etc., these data structures either lack human readability or support for the rich attributes often accompanying higher-order data. JSON is supported by all major programming languages, and JSON validation libraries written in Python, R, and Julia provide a user-friendly way to verify datasets, as seen in Figure 2, making this a suitably popular file format. HIF offers support not only for network-level metadata but also for attributes corresponding to nodes, hyperedges, and incidences, and it does so in an interpretable way. This specification also handles several different types of higher-order networks and is flexible enough to accommodate additional higher-order representations in the future.

HIF is the first cross-platform standardized file format for datasets of higher-order networks, hoping to unify the higher-order network science community around a set of standards and best practices. This standard was crystallized through exploration of the practical needs of the community and limitations of implementation. Not only do we describe a data standard, but we also provide a workflow for updating this schema as the needs of the higher-order network science community evolve and JSON schema specifications are updated.

3.1 The hypergraph interchange format specification

The HIF specification organizes data about a higher-order network in five different categories via five top-level fields in the JSON file: `network-type`, `metadata`, `nodes`, `edges`, and `incidences`, as can be seen in Table 1. The text corresponding to the `network-type` field indicates the type of the higher-order network and offers additional guidance for processing the dataset, which we will discuss in detail below. The data corresponding to the `metadata` field is a dictionary-like object representing network-level attributes such as the name of the dataset, when it was collected, the author of the dataset, corresponding references, and other high-level information enabling greater dataset interpretability. The `nodes` and `edges` fields store data records for the nodes and edges, respectively. These records can be used to specify metadata corresponding to each node and hyperedge or, as we will discuss in more detail below, to specify empty edges or isolated nodes. Lastly, the `incidences` field stores data records detailing node-hyperedge

Table 1. A description of the HIF schema. The schema is separated into five types of network data: the type of the higher-order network (encoded in `network-type`), network-level metadata (encoded in `metadata`), lists of nodes and edges with their attributes (encoded in `nodes` and `edges`, respectively), and structural information comprising information about the constituent nodes, hyperedges, and incidences

Field	Type	Description
Top-level attributes		
<code>network-type</code> (optional)	enum	“directed,” “undirected,” or “asc”
<code>metadata</code> (optional)	object	network-level metadata
<code>incidences</code> (required)	array	lists of records representing edge-node pairs
<code>nodes</code> (optional)	array	lists of node records
<code>edges</code> (optional)	array	lists of edge records
Entries in the <code>incidences</code> array		
<code>edge</code> (required)	string or integer	an edge ID
<code>node</code> (required)	string or integer	a node ID belonging to that edge
<code>weight</code> (optional)	number	incidence weight
<code>direction</code> (optional)	enum	“head” or “tail”
<code>attrs</code> (optional)	object	library-specific non-structural attributes
Entries in the <code>nodes</code> array		
<code>node</code> (required)	string or integer	a global node ID
<code>weight</code> (optional)	number	node weight
<code>attrs</code> (optional)	object	library-specific non-structural attributes
Entries in the <code>edges</code> array		
<code>edge</code> (required)	string or integer	a global hyperedge ID
<code>weight</code> (optional)	number	edge weight
<code>attrs</code> (optional)	object	library-specific non-structural attributes

relationships (known as an *incidence*). Below, we detail more of the structure as well as some of the implications of this specification.

We start with the *incidence* data, because this is the only top-level attribute which explicitly encodes the complex relationships between nodes via hyperedges. The `incidences` field is the *only* required field in HIF-compliant files, and a file only including this field corresponds to a higher-order network without any network, node, or hyperedge attributes. The incidence data are composed of a list of entries known as *records*, where each record indicates a relationship between a node and a hyperedge, known as an *incidence*. For example, {“edge”: 1, “node”: 3} indicates that node 3 belongs to hyperedge 1. The collection of all records where “edge” is 1 forms a hyperedge, e.g., the collection of records {“edge”: 1, “node”: 3}, {“edge”: 1, “node”: 4}, and {“edge”: 1, “node”: 7} corresponds to a hyperedge comprising the interaction between nodes 3, 4, and 7. Additional structural features which can be specified are `weight` and, in the case of directed hypergraphs, `direction`. These properties are added as top-level attributes in each record; e.g., {“edge”: 1, “node”: 3, “weight”: 2} specifies an incidence with an edge-dependent node weight of 2, and {“edge”: 1, “node”: 3, “direction”: “tail”} indicates that node 3 is in the tail of directed hyperedge 1. Incidences can also have properties, which indicate hyperedge-dependent nodal attributes.

Python

```
import fastjsonschema
import json
import requests

schema = requests.get(url).json()
validator = fastjsonschema.compile(schema)
hiftext = json.load(open(filename, 'r'))
try:
    validator(hiftext)
    print("HIF-Compliant JSON.")
except Exception as e:
    print(f"Invalid JSON: {e}")
```

R

```
library(jsonvalidate)
library(jsonlite)

schema <- paste(readLines(url, warn = FALSE))
validator <- json_validator(schema)
if (validator(filepath)) {
    print("HIF-Compliant JSON.")
} else {
    print("Invalid JSON.")
}
```

Julia

```
using HTTP
using JSON3
using JSONSchema

schema = String(HTTP.get(url).body)
validator = Schema(schema)
hiftext = JSON3.read(filepath)
result = JSONSchema.validate(validator, hiftext)
println(
    result === nothing ?
    "HIF-Compliant JSON." :
    "Invalid JSON: " * "$result"
)
```

Figure 2. Example code snippets for validating the schema in different languages. In all cases, `url` = "https://raw.githubusercontent.com/pszufer/HIF-standard/main/schemas/hif_schema.json" and `filepath` is the local filepath of the file being validated.

All of these attributes are bundled as dictionary-like objects in each incidence record under the `attrs` field, e.g., `{"edge": 1, "node": 3, "attrs": {"role": "PI"}}` might indicate node 3's role as a PI on project 1.

Similarly, we can specify attributes for nodes and hyperedges by creating records corresponding to the nodes and edges fields, respectively. For example, `{"node": 0, "attrs": {"height": 176, "weight": 143}}` corresponds to node 0 with a height of 176 and weight of 143, and `{"edge": 1, "attrs": {"duration": 93, "setting": "coffee shop"}}` corresponds to a 93-minute interaction at a coffee shop. Another use of these fields is to specify empty hyperedges and isolated nodes which don't participate in these incidence relationships and thus are not listed in the incidence records. We can add these nodes and edges as node and edge records with or without attributes to ensure that they are present in the higher-order network.

The `network-type` field can take one of three values: “asc,” “directed,” and “undirected.” This field offers additional guidance for processing the higher-order network. For example, using the “asc” keyword indicates that, even if all subfaces are not specified, that the network should be downward closed. The best practice for storing an abstract simplicial complex is to only store the maximal faces and subfaces with attributes to minimize the storage required, but this is a guideline, not a hard requirement. When using the “directed” keyword, this indicates that the library should expect the “direction” field indicating whether the incidence forms part of the head or tail of the directed hyperedge. When the `network type` is not specified, it is assumed that the type is an undirected hypergraph.

The `metadata` field can take arbitrary information to support the dataset under study. The schema allows any object in the `metadata` property, including deeply nested structures. Dataset authors are encouraged to use a flat object and to use only lowercase characters and hyphens, a convention commonly known as “dash case.” This can be seen in Figure 1 where the “Creation Date” property name is encoded as “creation-date,” and “DOI” is encoded as “doi.”

3.2 Checking HIF compliance

A central contribution of the HIF project is a JSON schema allowing simple validation of datasets against the standard. There are simple JSON validators in several different languages, and we illustrate simple ways to validate datasets against the HIF Standard in Figure 2.

We emphasize that the validating the schema should be the first step in the implementation of an independent software packages which utilize this standard are ultimately responsible for reading and writing datasets to and from the appropriate network representation. For example, the standard uses the “asc” keyword to specify that every interaction present in the hypergraph is downward closed, but checking the dataset against the schema doesn’t explicitly check this. In addition, best practices are that only the maximal simplex faces and subfaces with corresponding attributes are stored for abstract simplicial complexes, but this informal standard is not enforced by the schema. Lastly, while end users are able to, in principle, specify the `direction` keyword for network datasets even if the `network-type` keyword is not `directed` and to omit the `direction` keyword even when `network type` is `directed`, for simplicity, we simply say that `direction` field is optional.

3.3 Updating the schema

Changes made to the schema are documented in a `CHANGELOG.md` document. When a new version is released, the version process is followed:

1. Decide on the new version number based on the changes made since the last version and the Semantic Versioning guidelines (Preston-Werner, 2023).
2. Copy the schema in the `hif_schema.json` file to a file named `hif_schema_<version>.json` where `version` indicates the new version.
3. Add the changes made since the last release to `CHANGELOG.md` in a section with the new version as the name.
4. Upload the new stable version to Zenodo as a persistent reference.

The `hif_schema.json` schema will always have version “latest” and all unit tests are based on this schema.

4. Integration with software libraries

A strength of the HIF standard is its integration with five prominent software libraries for higher-order network analysis: the Hypergraph Analysis Toolbox (HAT) (Pickard et al., 2023),


```
import json
from HAT import Hypergraph
with open("example_hif.json") as file:
    hif_data = json.load(file)
HG = Hypergraph.from_hif(hif_data)
hif_output = HG.to_hif()
```

Figure 3. The example code demonstrates creating a HAT hypergraph from an example HIF JSON file and exporting it back to python.

Hypergraphx (Lotito et al., 2023), HyperNetX (Praggastis et al., 2024), SimpleHypergraphs.jl (Spagnuolo et al., 2020), and XGI (Landry et al., 2023). Here we detail the ways in which each library supports the HIF standard as well as brief descriptions of their use.

4.1 Hypergraph Analysis Toolbox

The Hypergraph Analysis Toolbox (HAT)³ (Pickard et al., 2023) is a general-purpose software for constructing, visualizing, and analyzing hypergraphs and higher-order structures, with a focus on their structure and dynamics. HAT implements tensor-based methods for the analysis of higher-order observability/controllability (Pickard et al., 2024; Pickard et al., 2023), coupling (Pickard et al., 2024), and structural properties suitable for analysis of a wide range of data (Chen and Rajapakse, 2020; Dotson et al., 2022). It provides HIF-compliant read and write capabilities in the HAT. Hypergraphs module. Figure 3 presents a sample using the HAT library. The `Hypergraph.to_hif` method converts the hypergraph object back to a Python dictionary according to the HIF schema, which can be saved to a JSON file.

4.2 Hypergraphx

Hypergraphx (HGX)⁴ is an open-source Python library designed for the analysis of complex systems characterized by higher-order interactions. HGX offers an extensive collection of tools and algorithms for constructing, manipulating, analyzing, and visualizing hypergraphs, enabling users to capture the rich structure and study the dynamics of real-world systems beyond pairwise relationships. HGX enables seamless storage and conversion of higher-order data across multiple representations, including hypergraphs, abstract simplicial complexes, bipartite graphs, line graphs, and clique expansions, all while supporting hyperedges with weights, direction, sign, temporal dynamics, and multiplicity. The library provides tools for basic statistical characterizations (e.g., hyperedge size distribution), higher-order centrality, motif analysis (with scalable sampling), community detection, filtering, and synthetic hypergraph generation. HGX also supports the simulation of dynamical processes and offers advanced visualization functionalities for exploring the structure of real-world systems.

On the data and I/O side, HGX is accompanied by HGX-data, a collection of ready-to-use datasets for higher-order network analysis in Python, including collaboration, face-to-face, biological data, and more. HGX offers dedicated input and output functions, including a binary format to enable fast and efficient I/O over large-scale higher-order datasets. Hypergraphx has integrated HIF support through the `read_hif` function, allowing users to load hypergraphs directly from HIF-compliant JSON files and automatically handling the conversion to HGX's internal data structures. Similarly, the `write_hif` function enables users to export an HGX `Hypergraph` object to the HIF format, preserving both the structural information and any associated node, edge, or incidence attributes. This interoperability allows users to seamlessly leverage HGX functionalities while moving back and forth across different libraries. In the future, HIF will accommodate additional hypergraph representations available in Hypergraphx,

```

import hypergraphx as hgx

# Load HIF hypergraph data to HGX object
from hypergraphx.readwrite import read_hif
HG = read_hif(filename="example_hif.json")

# Compute and evaluate motifs
from hypergraphx.motifs import compute_motifs
m = compute_motifs(HG, runs_config_model=20)

# Save HGX hypergraph object to HIF file
from hypergraphx.readwrite import write_hif
write_hif(HG, filename="example_hif.json")

```

Figure 4. Example demonstrating how to load a HIF-compliant file, use hypergraphx (HGX) functionalities and export the file.

including multiplex hypergraphs (Lotito et al., 2024). Moreover, HGX-data is planned to be fully HIF-compatible in order to be accessed by the whole higher-order software ecosystem.

An example demonstrating how to load a hypergraph from a HIF file, leverage HGX functionalities such as motif analysis, and export the data is provided in Figure 4.

4.3 HyperNetX

HyperNetX (HNX) (Praggastis et al., 2024)⁵ is a python package focused on developing tools for the analysis and visualization of hypergraphs. HNX provides a user-friendly interface for creating, manipulating, and studying hypergraphs, with an emphasis on visualization capabilities that capture higher-order relationships. Some of the key features of HNX include:

- Support for incidence, node, and hyperedge attributes, allowing for richer data representation.
- A suite of network analytical functions, including line graphs, degree sequences, connectivity measures, and centrality metrics specifically modified to be appropriate for hypernetworks (Aksoy, Joslyn et al., (2020)).
- Support for topological measures of hypergraphs, including simplicial and zigzag homology of temporal hypergraphs (Myers et al., 2023).
- Tutorials for both basic and advanced hypergraph analysis methods with a contributors guide for the addition of new modules and tutorials.
- Visualization tools for rendering hypergraphs in various layouts, highlighting different aspects of their structure.
- Advanced analytical capabilities through supplementary modules including generative models, Laplacian clustering, and hypergraph modularity.

HyperNetX has integrated HIF through the `read_hif` and `to_hif` functions: `read_hif` loads hypergraphs directly from HIF-compliant JSON files, handling the conversion to HNX's internal data structure and `to_hif` exports an HNX hypergraph object to the HIF format, preserving both the structural information and any associated node, edge, or incidence attributes. An example demonstrating how to load and save an HNX hypergraph is shown in Figure 5 where the filename is provided. Alternatively, the JSON object can be loaded first and then the JSON object converted to an HNX hypergraph can be done.

```
import json
import hypernetx as hnx
# Load HIF hypergraph data to hnx object
HG = hnx.from_hif(filename = "example_hif.json")
# Save HIF data to file
hnx.to_hif(HG, filename="example_hif.json")
```

Figure 5. Example demonstrating how to load and export and HIF file using HyperNetX (HNX).

```
using SimpleHypergraphs
hg = Hypergraph{Int, String, String}({
    [1 nothing      2 nothing;
     3      1 nothing      4])
set_vertex_meta!(hg, "vertex 1", 1)
set_hyperedge_meta!(hg, "h-edge 2", 2)
g_save("hg_hif.json", hg; format=HIF_Format())
loaded_hg = hg_load(
    "hg_hif.json";
    format=HIF_Format(),
    HType=Hypergraph,
    T=Int, V=String, E=String
)
```

Figure 6. The example code demonstrates creating a hypergraph with node and hyperedge metadata represented as strings and exporting the hypergraph to a HIF-compliant JSON file.

4.4 SimpleHypergraphs.jl

SimpleHypergraphs.jl⁶ is a Julia package designed to efficiently work with hypergraphs (Spagnuolo et al., 2020). Its key capabilities include (1) creating hypergraphs from incidence matrices, random hypergraph models, a Graphs.jl graph, or external data; (2) manipulating hypergraphs; (3) analyzing hypergraphs with fast algorithms for computing, for example, connected components or modularity-based clustering; (4) visualizing hypergraphs; (5) integration with the Graphs.jl Julia package via bipartite graphs or a hypergraph projection; and (6) and file I/O. SimpleHypergraphs.jl is designed to support (1) two-way of storage of node and hyperedge incidence information, which enables efficient traversal of the hypergraph in both directions—from nodes to hyperedges and vice versa; (2) hyperedge-dependent node weights, i.e., incidence weights; (3) node and hyperedge metadata; (4) compatibility with Julia's AbstractMatrix interface, enabling efficient manipulation of hypergraphs using standard Julia matrix operations; and (5) lazy representation of hypergraph views, which are compatible with the AbstractGraph interface of the Graphs.jl library, enabling seamless integration and direct usage of Graphs.jl algorithms. Figure 6 presents an example of SimpleHypergraphs.jl library usage with HIF.

4.5 XGI

The Complex Group Interactions (XGI)⁷ (Landry et al., 2023) software package provides support for hypergraphs, directed hypergraphs, and abstract simplicial complexes. It provides a comprehensive suite of analysis and visualization tools, including algorithms, generative higher-order network models, linear algebra representations, conversions between many different data structures, and an integrated statistical interface. Integrated with this software environment is the XGI-DATA repository, which hosts many higher-order datasets. XGI provides not only visualization and analysis tools for higher-order networks but the ability to read and write numerous different file formats, such as bipartite edge lists, hyperedge lists, and incidence matrices.

```

import xgi
H = xgi.Hypergraph(
    [[1, 2, 3], [2, 3, 4], [1, 4]]
)
example_file = "example_hif.json"
xgi.write_hif(H, example_file)
H = xgi.read_hif(example_file, nodetype=str)
H.edges.members()
[{"1", "2", "3"}, {"2", "3", "4"}, {"1", "4"}]

```

Figure 7. The code example demonstrates using XGI to read and write HIF-compliant files. First, a hypergraph with hyperedges {1, 2, 3}, {2, 3, 4}, and {1, 4} is generated and written to an HIF-compliant JSON file. Second, the hypergraph is read from that file, casting the names of the nodes from integers to strings, and the hyperedge list is returned.

XGI currently supports the HIF standard in two ways: first, it provides HIF-compliant read and write capabilities, and second, it provides HIF-compliant datasets in the XGI-DATA collection. XGI has implemented an `hif` sub-module in its `readwrite` module, providing two methods: `read_hif` and `write_hif`. These methods allow XGI to store undirected hypergraphs, directed hypergraphs, and abstract simplicial complexes according to the HIF standard. When storing directed hypergraphs, XGI makes use of the `direction` field in the incidence records. When storing abstract simplicial complexes, XGI only stores the maximal faces and hyperedges with associated attributes, with the expectation that the `asc` keyword will indicate that XGI should enforce downward closure when reading in the file. An example demonstrating how to create a hypergraph, save the hypergraph as an HIF file, and then loading that file is shown in Figure 7.

The `load_xgi_data` function in XGI supports reading HIF-compliant files from the XGI-DATA repository, and several datasets in XGI-DATA are stored in XGI-DATA. Future plans include converting all datasets to HIF to make XGI-DATA a shared resource for all higher-order software libraries.

5. Case study

To illustrate the practical application and interoperability enabled by the HIF, we present a case study. Figure 9 provides an overview of how HIF standard facilitates the exchange of hypergraph data across several hypergraph analysis software packages—Hypergraphx, HyperNetX, SimpleHypergraphs.jl, and XGI—enabling more integrated and sophisticated data analysis pipelines.

5.1 Dataset

We create the `publications.hif.json` HIF-compliant file and interact with this dataset by loading the file through each software library's API.

We use a publication dataset for our case study because of its rich attributes and tractable size. This dataset consists of open-source publications with the keyword “Hypergraph” and was collected from ArXiv, BioRxiv, the DBLP computer science bibliography, and the U.S. Office of Scientific and Technical Information (OSTI). The resulting hypergraph is composed of hyperedges, which represent scientific publications, and nodes, which represent authors and can be seen in Figure 8. Each hyperedge (publication) has a number of attributes, including funding agencies, abstract, publication date, keyword tags, and the source, while each node (author) has an associated institution have the attributes of institutions (there are no incidence attributes). The hypergraph has 1,960 nodes (authors) and 533 edges (publications). A detailed description and the following analysis of this dataset using each of the hypergraph packages is provided in the

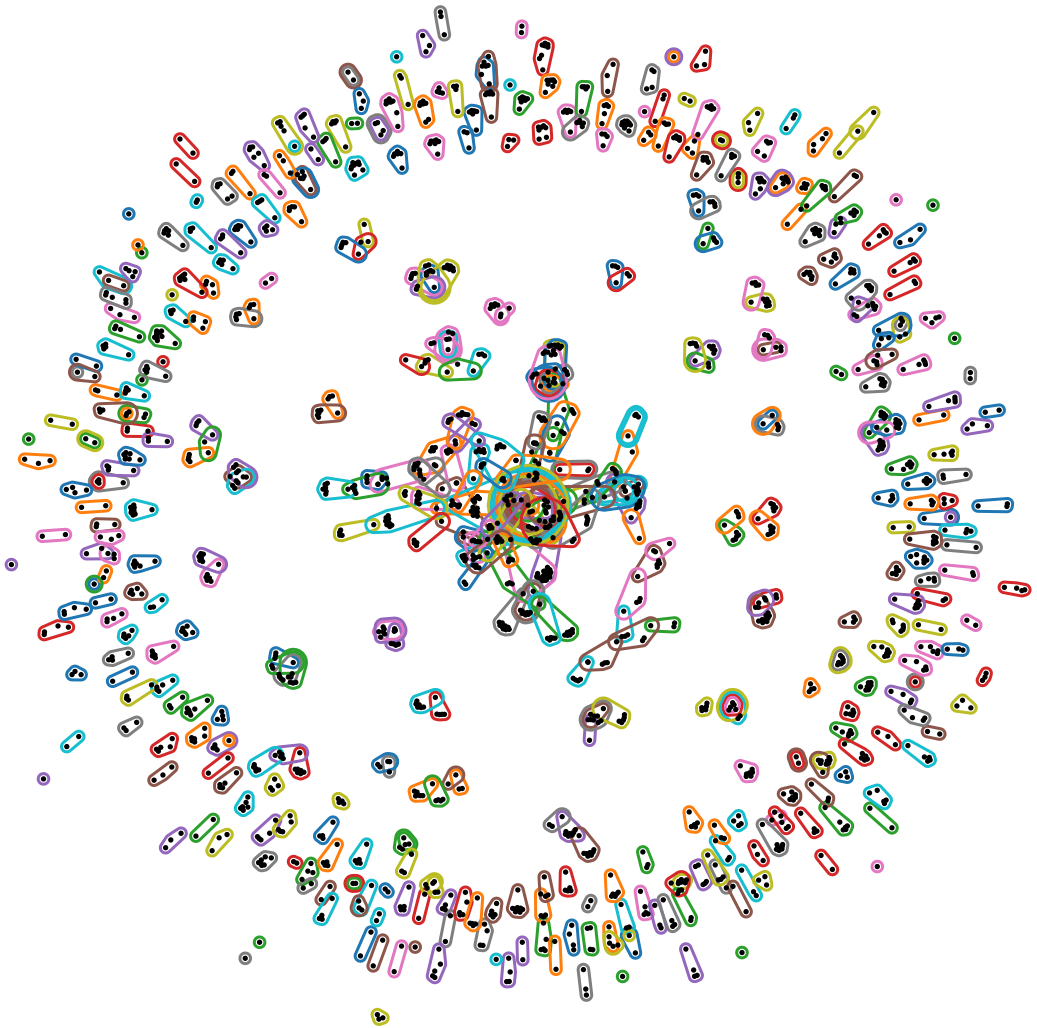


Figure 8. Illustration of the full publications hypergraph, including disconnected components.

notebook `HIF-demo.ipynb` in the tutorials folder of the HIF GitHub page.⁸ We briefly describe examples of the types of analysis that one can perform on this dataset for each software package.

5.1.1 Hypergraph Analysis Toolbox

For HAT we demonstrate analyzing the publications hypergraph using the eigenvector centrality measure. This allows us to analyze node and edge centrality, indicating the contributions of each author and publication. In Figure 9(a), the incidence matrix of the largest connected component of the hypergraph is shown, along with the node and hyperedge centrality measures. Each row of the incidence matrix indicates an author (node), and the columns of the matrix correspond to publications (hyperedges). HAT computes the centralities of the nodes and hyperedges, displaying them along the corresponding modes of the incidence matrix.

5.1.2 Hypergraphx

To demonstrate the capabilities of Hypergraphx, we consider our example publications dataset and perform an exploratory analysis combining community detection, motifs, and visualization.

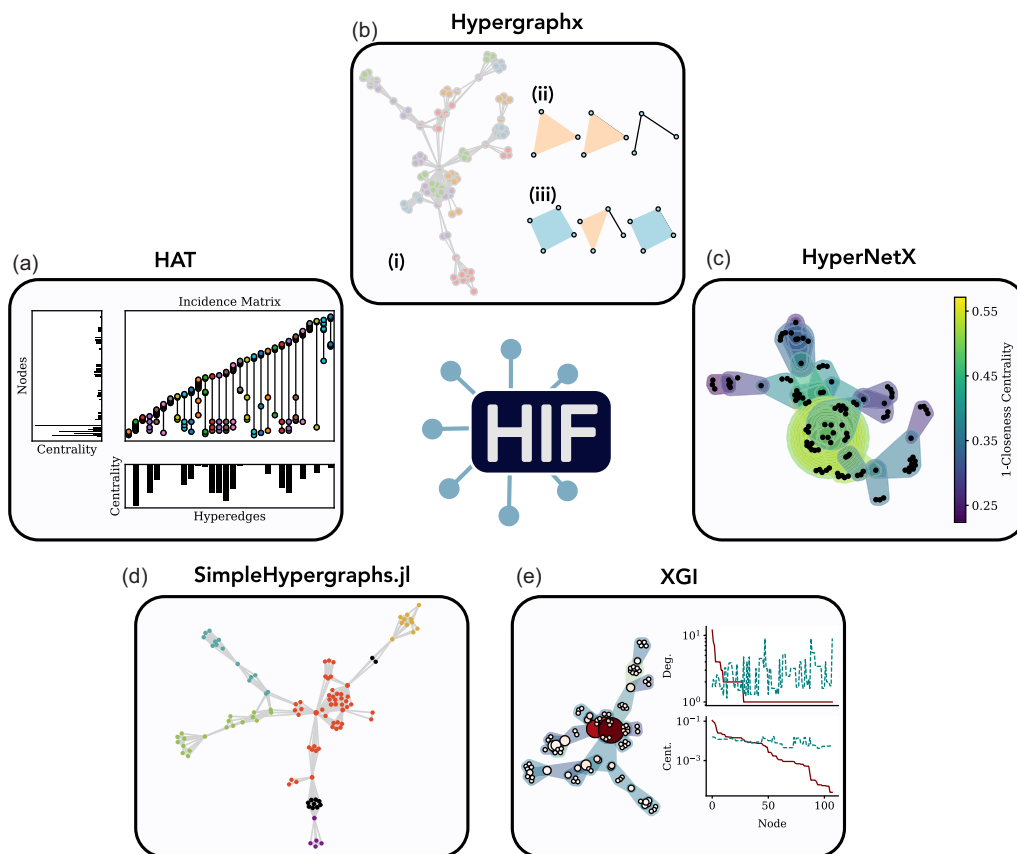


Figure 9. The hypergraph interchange format enables interoperability between many popular higher-order network analysis libraries, as seen in this case study, which analyzes a publication dataset. All code to reproduce this figure is available on the HIF-standard repository (see <https://github.com/pszufte/HIF-standard>). The notebook that creates each visualization can be found in the `HIF-demo.ipynb` file. In the center, we display the HIF logo. In panel (a), we use HAT to compute the nonlinear eigenvector centralities of the nodes and hyperedges. In panel (b), we use hypergraphx to visualize the largest connected component of the hypergraph (i), with node colors indicating the community structure. We also visualize the most frequent patterns of group interactions involving three (ii) and four nodes (iii). In panel (c), we use hyperNetX to solely examine the largest component of the hypergraph, compute the closeness centrality for each hyperedge, and color each hyperedge according to its centrality as we can see from the color bar. In panel (d), we use SimpleHypergraphs.jl to find the largest component, compute modularity-based community detection, and to visualize the network projected from the hypergraph, where each hyperedge becomes a clique. In panel (e), we use XGI to compute nodal statistics using the statistics interface; in the visualization, the node size is scaled by the nodal degree and the node color corresponds to the clique eigenvector centrality (CEC) (Benson, 2019). Similarly, in the top inset figure, we see the degree (solid line) and average neighbor degree (dashed line), while in the bottom inset figure, we see the CEC (solid line) and the H-eigenvector centrality (HEC) (Aksoy et al., 2024; Benson, 2019) (dashed line). Note that the node labels are sorted in descending order according to the degree (top) and CEC (bottom).

Specifically, in Figure 9(b), we apply (overlapping) community detection to the largest connected component, identifying groups of closely related nodes (Contisciani et al., 2022), and visualize the structure of the resulting hypergraph. We further use Hypergraphx to mine the frequency of subhypergraph patterns of sizes 3 and 4 (Lotito et al., 2024), describing the hypergraph of publications at its microscale and providing insights into the structural building blocks of higher-order interactions present in the data (Lotito et al., 2022). This use case illustrates how Hypergraphx can serve as a key component of an integrated workflow within the higher-order

network ecosystem, enabling complex tasks such as motif analysis and community detection to be performed efficiently and in combination with other tools.

5.1.3 HyperNetX

For HyperNetX, we demonstrate the usefulness of analyzing the main component of the publications hypergraph using the s -closeness centrality measure for $s = 1$, visualized as an Euler diagram in Figure 9(c). We did not use the full hypergraph for this centrality analysis as it is too dense and disconnected to visually discern hyperedge centrality. The s -closeness centrality measure shows the closeness of each hyperedge (in this case, a publication) based on the shared authors. The color scale indicates the centrality value, with warmer colors (yellow) representing hyperedges that are more central, meaning they share more authors with other publications. This analysis can highlight influential publications that are central to the collaborative structure of this publication hypergraph.

5.1.4 SimpleHypergraphs.jl

For the analysis with SimpleHypergraphs.jl, we have selected the largest connected component of the citation hypergraph that has 108 vertices and 30 hyperedges. Next, we use library's functionality hypergraph modularity-based clustering introduced in (Kamiński et al., 2019) to identify clusters within the citation hypergraph. Subsequently, we project the hypergraph to a pairwise representation where every hyperedge becomes a clique. We visualize the graph in Figure 9(d), scaling node sizes by their degrees and using colors to denote community labels. All communities with a single node are colored in black.

5.1.5 XGI

A strength of the XGI software package is its statistics interface, allowing practitioners to easily visualize, output as common Python data types, and compute statistics of node and edge properties. In Figure 9(e), we see a hypergraph with its nodes colored and sized according to its degree and the average degree of its neighbors. XGI can also easily filter datasets by hyperedge size according to the framework in (Landry et al., 2024). Another strength of the XGI software package is its ability to quickly remove data artifacts with its `cleanup` method, which has the ability to remove singletons, isolated nodes, multi-edges, and disconnected components. In Figure 9(e), we see the largest component of the hypergraph without any multi-edges, isolated nodes, or singletons.

6. Discussion

In this paper, we introduced the Hypergraph Interchange Format as a standard for sharing hypergraph data across different scientific workflows. We demonstrated its effectiveness with a case study showcasing the use of several software packages to analyze a single dataset in a variety of ways, each catering to a different library's strengths. In the current fragmented open-source ecosystem, the HIF standard offers interoperability between many higher-order network analysis libraries.

As part of this effort, we surveyed the landscape of open-source software for existing standards and potential adopters, reaching out to more than 20 open-source projects with support for hypergraphs and abstract simplicial complexes. Our exploration validated the hypothesis that the community of higher-order networks needs an open standard and that the HIF design supports many use cases. Future work will include integration with more software libraries to improve its reach in the higher-order network science community.

The HIF standard currently emphasizes human readability and support for richly attributed data, but one can imagine more efficient ways for storing datasets, which could be implemented in future iterations of HIF. For example, Javascript Object Notation Lines (JSONL) is a specification

defining how to transmit a top-level list of JSON objects as a series of one-line JSON documents, enabling streaming file input/output (I/O). When solely considering performance, one could consider a database architecture with a query language such as GraphQL as a performant alternative for storing and retrieving higher-order network data. As large higher-order datasets become increasingly more available, efficient storage and handling of higher-order network data will become more and more necessary.

Lastly, while the HIF standard directly supports undirected and directed hypergraphs and abstract simplicial complexes, support for additional higher-order representations will make HIF more widely useful. Examples include hypergraphs with ordered and partially ordered sets (posets), which are useful in legal analysis (Coupette *et al.*, 2024) and the science of science. Several types of higher-order networks are implicitly supported by the HIF standard: signed hypergraphs, which assign a sign \pm to each hyperedge; multilayer hypergraphs, which assign nodes and hyperedges to different layers representing, for example, modes of transportation, relationship types, or social media platforms; and temporal hypergraphs where nodes and hyperedges have associated time intervals in which they are active. The specific structural information necessary for each of these representations can be stored in the `attrs` field, but the schema excludes these as expected JSON fields. In the future, adding explicit support for these fields will improve compatibility with libraries such as ASH (Failla *et al.*, 2023), Raptory (Steer *et al.*, 2024), and Reticula (Badie-Modiri and Kivelä, 2023). In addition, we recommend that when releasing new higher-order datasets that an HIF-compliant file is included in the release [for example, see (Smith *et al.*, 2025)].

While there are many avenues for future growth of HIF, we believe that this is a significant first step in the standardization of the higher-order network data sharing ecosystem. It has a direct impact on the reproducibility of published works that adopt the standard. HIF will facilitate seamless data exchange, cross-software collaboration, and more compact higher-order data analysis pipelines and will be a significant advance for the coming years of higher-order network science.

Author contributions. Author order is alphabetical.

Conceptualization: C.A.J., N.W.L., B.P., P.S.

Methodology: M.C., C.A.J., N.W.L., Q.F.L., A.M., J.P., B.P., P.S.

Project Administration and Supervision: N.W.L.

Software: M.C., N.W.L., Q.F.L., A.M., J.P., B.P., P.S.

Writing – Original Draft: M.C., C.A.J., N.W.L., Q.F.L., A.M., J.P., B.P., P.S.

Writing – Review & Editing: M.C., C.A.J., N.W.L., Q.F.L., A.M., J.P., B.P., P.S.

Visualization: M.C., N.W.L., Q.F.L., A.M., J.P., P.S.

Funding statement. N.W.L. acknowledges support from the University of Virginia Prominence-to-Preeminence (P2PE) STEM Targeted Initiatives Fund, SIF176A Contagion Science. This work acknowledges support from Pacific Northwest National Laboratory with information release number PNNL-SA-213514. The research by P.S. was funded by the National Science Centre (NCN), Poland (grant number: 2021/41/B/HS4/03349). Pacific Northwest National Laboratory is a multiprogram national laboratory operated for the US Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

Data availability statement. All datasets, resources, documentation, and schemas are available on GitHub⁸. The stable schema is available on Zenodo (zenodo_2025) and subsequent stable versions will be released on Zenodo.

Ethical standards. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

Supplementary material. The supplementary material for this article can be found at <http://doi.org/10.1017/nws.2025.10018>.

Competing interests. None.

Notes

- 1 See <https://zenodo.org/communities/xgi>.
- 2 We will commonly refer to “edges” to mean hyperedges for convenience and when clear from context.
- 3 <https://hypergraph-analysis-toolbox.readthedocs.io>
- 4 <https://github.com/HGX-Team/hypergraphx>
- 5 <https://github.com/pnnl/HyperNetX>
- 6 <https://github.com/pszufe/SimpleHypergraphs.jl>
- 7 <https://xgi.readthedocs.io>
- 8 See <https://github.com/pszufe/HIF-standard>.

References

- Adams J (2019). Gathering social network data, 1st edn. Thousand Oaks: SAGE Publications, Inc.
- Aksoy SG, Amburg I and Young SJ (2024). Scalable tensor methods for nonuniform hypergraphs. *SIAM Journal on Mathematics of Data Science*, 6(2), 481–503. <https://doi.org/10.1137/23M1584472>.
- Aksoy SG, Joslyn C, Marrero CO, Praggastis B and Purvine E (2020). Hypernetwork science via high-order hypergraph walks. *EPJ Data Science*, 9(1), 1–34. <https://doi.org/10.1140/epjds/s13688-020-00231-0>.
- Albert R, Jeong H and Barabási AL (1999). Diameter of the World-wideWeb. *Nature*, 401(6749), 130–131. <https://doi.org/10.1038/43601>.
- Antelmi A, Daniele DV and Carmine S (2024). HypergraphRepository: a community-driven and interactive hypernetwork data collection. In Dewar M, Kaminski BI, Kaszynski D, Krainiski Ł, Prałat PI and Théberge F (eds), *Modelling and Mining Networks*. Cham, Switzerland: Springer Nature, pp. 159–173. https://doi.org/10.1007/978-3-031-59205-8_11.
- Ausiello G, Franciosa PG and Frigioni D (2001). Directed hypergraphs: problems, algorithmic results, and a novel decremental approach. In Restivo A, Rocca SRD and Roversi L (eds), *Theoretical Computer Science Berlin, Heidelberg*: Springer, pp. 312–328. https://doi.org/10.1007/3-540-45446-2_20.
- Badie-Modiri A and Kivelä M (2023). Reticula: a temporal network and hypergraph analysis software package. *SoftwareX*, 101301, 2352–7110. <https://doi.org/10.1016/j.softx.2022.101301>.
- Barabási AL, Jeong H, Neda Z, Ravasz E, Schubert A and Vicsek T (2002). Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3), 590–614. [https://doi.org/10.1016/S0378-4371\(02\)00736-7](https://doi.org/10.1016/S0378-4371(02)00736-7).
- Barabási AL and Oltvai ZN (2004). Network biology: understanding the cell’s functional organization. *Nature Reviews Genetics*, 5(2), 101–113. <https://doi.org/10.1038/nrg1272>.
- Battiston F, Cencetti G, Iacopini I, Latora V, Lucas M, Patania A, Young JG and Petri G (2020). Networks beyond pairwise interactions: structure and dynamics. *Physics Reports, Networks beyond Pairwise Interactions: Structure and Dynamics*, 874, 1–92. <https://doi.org/10.1016/j.physrep.2020.05.004>.
- Benson AR (2019). Three hypergraph eigenvector centralities. *SIAM Journal on Mathematics of Data Science*, 1(2), 293–312. <https://doi.org/10.1137/18M1203031>.
- Benson AR (2021). Data! Available at <https://www.cs.cornell.edu/~arb/data/>.
- Berge C (1989). *Hypergraphs: combinatorics of finite sets*, vol. 45. Amsterdam: Elsevier.
- Bick C, Gross E, Harrington HA and Schaub MT (2023). What are higher-order networks? *SIAM Review*, 65(3), 686–731. <https://doi.org/10.1137/21M1414024>.
- Brandes U, Eiglsperger M, Lerner J and Pich C (2010). Graph markup language (GraphML). In Tamassia, R., (ed), *Handbook of graph drawing and visualization*. London: Chapman & Hall, pp. 517–541.
- Bretto A (2013). *Hypergraph Theory: An Introduction*. Mathematical Engineering. Heidelberg: Springer International Publishing, <https://doi.org/10.1007/978-3-319-00080-0>.
- Brockmann D and Helbing D (2013). The hidden geometry of complex, network-driven contagion phenomena. *Science*, 342(6164), 1337–1342. <https://doi.org/10.1126/science.1245200>.
- Cattuto C, Van den Broeck W, Barrat A, Colizza V, Pinton JF and Vespignani A (2010). Dynamics of person-to-person interactions from distributed RFID sensor networks. *PLOS ONE*, 5(7), 1–9. <https://doi.org/10.1371/journal.pone.0011596>.
- Chen C. and Rajapakse I. (2020). Tensor entropy for uniform hypergraphs. *IEEE Transactions on Network Science and Engineering*, 7(4), 2889–2900. <https://doi.org/10.1109/TNSE.2020.3002963>.
- Chitra U and Raphael B (2019). RandomWalks on hypergraphs with edge-dependent vertex weights. In Proceedings of the 36th International Conference on Machine Learning, pp. 1172–1181. Available at <https://proceedings.mlr.press/v97/chitra19a.html>.
- Clauset A, Tucker E and Sainz M (2016). The colorado index of complex networks. Available at <https://icon.colorado.edu/>.
- Conticiani M, Battiston F and De Bacco C (2022). Inference of hyperedges and overlapping communities in hypergraphs. *Nature Communications*, 13(1), 7229. <https://doi.org/10.1038/s41467-022-34714-7>.

- Coupette C, Hartung D and Katz DM (2024). Legal hypergraphs. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 382(2270), 20230141. <https://doi.org/10.1098/rsta.2023.0141>.
- Diaz LPM and Stumpf MPH (2022). HyperGraphs.jl: representing higher-order relationships in Julia. *Bioinformatics*, 38(14), 3660–3661. doi: <https://doi.org/10.1093/bioinformatics/btac347>.
- Dotson GA, Chen C, Lindsly S, Cicalo A, Dilworth S, Ryan C, Jeyarajan S, Meixner W, Stansbury C, Pickard J, Beckloff N, Surana A, Wicha M, Muir LA, Rajapakse I (2022). Deciphering multi-way interactions in the human genome. *Nature Communications*, 13(1), 5498. <https://doi.org/10.1038/s41467-022-32980-z>.
- Failla A, Citraro S and Rossetti G (2023). Attributed stream hypergraphs: temporal modeling of node-attributed high-order interactions. *Applied Network Science*, 8(1), 1–19. <https://doi.org/10.1007/s41109-023-00555-6>.
- Gallo G, Longo G, Pallottino S and Nguyen S (1993). Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2), 177–201. [https://doi.org/10.1016/0166-218X\(93\)90045-P](https://doi.org/10.1016/0166-218X(93)90045-P).
- Grilliete W and Rusnak LJ (2023). Incidence hypergraphs: the categorical inconsistency of set-systems and a characterization of quiver exponentials. *Journal of Algebraic Combinatorics*, 58(1), 1–36. <https://doi.org/10.1007/s10801-023-01232-8>.
- Hajij M, Papillon M, Frantzen F, Agerberg J, Aljabea I, Ballester R, Battiloro C, Bernárdez G, Birdal T, Brent A, Chin P, Escalera S, Fiorellino S, Gardaa OH, Gopalakrishnan G, Govil D, Hoppe J, Karri MR, Khouja J, Lecha M, Livesay N, Meißner J, Mukherjee S, Nikitin A, Papamarkou T, Prilepok J, Ramamurthy KN, Rosen P, Guzmán-Sáenz A, Salatiello A, Samaga SN, Scardapane S, Schaub MT, Scofano L, Spinelli I, Telyatnikov L, Truong Q, Walters R, Yang M, Zaghen O, Zamzmi G, Zia A and Miolane N (2024). TopoX: a suite of python packages for machine learning on topological domains. *Journal of Machine Learning Research*, 25(374), 1–8. Available at <http://jmlr.org/papers/v25/24-0110.html>
- Hayashi K, Aksoy SG, Park CH and Park, H (2020). Hypergraph RandomWalks, Laplacians, and clustering. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20. New York, NY, USA: Association for Computing Machinery, pp. 495–504. <https://doi.org/10.1145/3340531.3412034>.
- HGX-Team(2025). HGX-data: collection of datasets for higher-order network analysis. Available at <https://github.com/HGX-Team/data>.
- Iacopini I, Karsai M and Barrat A (2024). The temporal dynamics of group interactions in higher-order social networks. *Nature Communications*, 15(1), 7391. <https://doi.org/10.1038/s41467-024-50918-5>.
- Jackson SJ, Bailey M and FoucaultWelles B (2020). *#HashtagActivism: Networks of Race and Gender Justice*. March: MIT Press.
- Jeong H, Tombor B, Albert R, Oltvai ZN and Barabási AL (2000). The large-scale organization of metabolic networks. *Nature*, 407(6804), 651–654. <https://doi.org/10.1038/35036627>.
- Joslyn C, Aksoy SG, Callahan TJ, Hunter LE, Jefferson B, Praggastis B and Tripodi IJ (2021). Hypernetwork science: from multidimensional networks to computational topology. In Braha D, de Aguiar MAM, Gershenson C, Morales AJ, Les K, Naumova EN, Minai AA and Bar-Yam Y (Eds), *Unifying Themes in Complex Systems X*. Cham: Springer, pp. 377–392, https://doi.org/10.1007/978-3-030-67318-5_25.
- Jost J and Mulas R (2019). Hypergraph Laplace operators for chemical reaction networks. *Advances in Mathematics*, 351, 870–896. <https://doi.org/10.1016/j.aim.2019.05.025>.
- Kaminski B, Poulin V, Pralat P, Szufel P and Théberge F (2019). Clustering via hypergraph modularity. *PLOS ONE*, 14(11), e0224307. <https://doi.org/10.1371/journal.pone.0224307>.
- Kunegis J (2013). KONECT: the Koblenz network collection. Proceedings of the 22nd International Conference on World-WideWeb, WWW'13 Companion. New York, NY, USA, May: Association for Computing Machinery. pp. 1343–1350. <https://doi.org/10.1145/2487788.2488173>.
- Landry NW, Amburg I, Shi M and Aksoy SG (2024). Filtering higher-order datasets. *Journal of Physics: Complexity*, 5(1), 2632–072X. <https://doi.org/10.1088/2632-072X/ad253a>.
- Landry NW, Lucas M, Iacopini I, Petri G, Schwarze A, Patania A and Torres L (2023). XGI: a Python package for higher-order interaction networks. *Journal of Open Source Software*, 8(85). <https://doi.org/10.21105/joss.05162>.
- Leskovec J, & Krevl A (2014). SNAP Datasets: Stanford large network dataset collection, June. Available at <http://snap.stanford.edu/data>.
- Lotito QF, Contisciani M, De Bacco C, Di Gaetano L, Gallo L, Montresor A and Battiston F (2023). Hypergraphx: a library for higher-order network analysis. *Journal of Complex Networks*, 11(3), cnad019. <https://doi.org/10.1093/comnet/cnad019>.
- Lotito QF, Montresor A and Battiston F (2024). Multiplex measures for higher-order networks. *Applied Network Science*, 9(1), 1–14. <https://doi.org/10.1007/s41109-024-00665-9>.
- Lotito QF, Musciotto F, Battiston F and Montresor A (2024). Exact and sampling methods for mining higher-order motifs in large hypergraphs. *Computing*, 106(2), 475–494. <https://doi.org/10.1007/s00607-023-01230-5>.
- Lotito QF, Musciotto F, Montresor A and Battiston F (2022). Higher-order motif analysis in hypergraphs. *Communications Physics*, 5(1), 1–8. <https://doi.org/10.1038/s42005-022-00858-7>.
- Morstatter F, Pfeffer J, Liu H and Carley K (2013). Is the sample good enough? Comparing data from twitter's streaming API with Twitter's Firehose. In *Proceedings of the International AAAI Conference on Web and Social Media*, 7(1), 400–408. <https://doi.org/10.1609/icwsm.v7i1.14401>.
- Murgas KA, Saucan E and Sandhu R (2022). Hypergraph geometry reflects higher-order dynamics in protein interaction networks. *Scientific Reports*, 12(1), 20879. <https://doi.org/10.1038/s41598-022-24584-w>.

- Myers A, Joslyn C, Kay B, Purvine E, Roek G and Shapiro M (2023). Topological analysis of temporal hypergraphs. In Dewar M, Prałat P, Szufel P and Théberge F (eds), *Algorithms and Models for the Web Graph*, Cham, Switzerland: Springer Nature, pp. 127–146. https://doi.org/10.1007/978-3-031-32296-9_9.
- Neuhäuser L, Mellor A and Lambiotte R (2020). Multibody interactions and nonlinear consensus dynamics on networked systems. *Physical Review E*, 101(3), 032310. <https://doi.org/10.1103/PhysRevE.101.032310>.
- Newman MEJ (2001). The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2), 404–409. <https://doi.org/10.1073/pnas.98.2.404>.
- Nooy WD, Mrvar A and Batagelj V (2018). *Exploratory Social Network Analysis with Pajek: Revised and Expanded Edition for Updated Software*. Cambridge University Press.
- ONNX Community. (2024). ONNX: Open neural network exchange. Available at <https://onnx.ai/>.
- Peixoto TP (2021). Netzschleuder. Available at <https://networks.skewed.de/>.
- Pickard J, Chen C, Salman R, Stansbury C, Kim S, Surana A, Bloch A and Rajapakse I (2023). HAT: hypergraph analysis toolbox. *PLOS Computational Biology*, 19(6), 1–7. <https://doi.org/10.1371/journal.pcbi.1011190>.
- Pickard J, Chen C, Stansbury C, Surana A, Bloch AM and Rajapakse I (2024). Kronecker product of tensors and hypergraphs: structure and dynamics. *SIAM Journal on Matrix Analysis and Applications*, 45(3), 1621–1642. <https://doi.org/10.1137/23M1592547>.
- Pickard J, Stansbury C, Surana A, Rajapakse I and Bloch A (2024). Geometric aspects of observability of hypergraphs. *IFAC-PapersOnLine*, 8th IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control LHMNC. 58(6), 321–326. <https://doi.org/10.1016/j.ifacol.2024.08.301>.
- Pickard J., Surana A., Bloch A. and Rajapakse I. (2023). Observability of hypergraphs. In 2023 62nd IEEE Conference on Decision and Control (CDC). pp. 2445–2451. <https://doi.org/10.1109/CDC49753.2023.10383387>.
- Praggastis B, Aksoy S, Arendt D, Bonicillo M, Joslyn C, Purvine E and Yun JY (2024). HyperNetX: a python package for modeling complex network data as hypergraphs. *Journal of Open Source Software* 9(95), 2475–9066. <https://doi.org/10.21105/joss.06016>.
- Preston-Werner T (2023). Semantic Versioning 2.0.0. Available at <https://semver.org/>.
- Smith A, Amburg I, Kumar S, Foucault Welles B and Landry NW (2025). A blue start: a large-scale pairwise and higher-order social network dataset. arXiv: 2505.11608. <https://doi.org/10.48550/arXiv.2505.11608>.
- Spagnuolo C, Cordasco G, Szufel P, Prałat P, Scarano V, Kaminski B and Antelmi A (2020). Analyzing, exploring, and visualizing complex networks via hypergraphs using SimpleHypergraphs.jl. *Internet Mathematics* 1(1). <https://doi.org/10.24166/im.01.2020>.
- Sporns O, Chialvo DR, Kaiser M and Hilgetag CC (2004). Organization, development and function of complex brain networks. *Trends in Cognitive Sciences* 8(9), 418–425. <https://doi.org/10.1016/j.tics.2004.07.008>.
- Steer B, Arnold NA, Ba CT, Lambiotte R, Yousaf H, Jeub L, Murariu F, Kapoor S, Rico P, Chan R, Chan L, Alford J, Clegg, RG, Cuadrado F, Barnes MR, Zhong P, Pougué-Biyong J and Alnaimi A (2024). Raphtory: the temporal graph engine for rust and python. *Journal of Open Source Software* 9(95), 5940. <https://doi.org/10.21105/joss.05940>.
- Stehlé J, Voirin N, Barrat A, Cattuto C, Isella L, Pinton JF, Quaggiotto M, Van den Broeck W, Régis C, Lina B, Vanhems P and Viboud C (2011). High-resolution measurements of face-to-face contact patterns in a primary school. *PLOS ONE* 6(8), e23176. <https://doi.org/10.1371/journal.pone.0023176>.
- Torres L, Blevins AS, Bassett D and Eliassi-Rad T (2021). The why, how, and when of representations for complex systems. *SIAM Review* 63(3), 435–485. <https://doi.org/10.1137/20M1355896>.
- Ugander J, Backstrom L, Marlow C and Kleinberg J (2012). Structural diversity in social contagion. *Proceedings of the National Academy of Sciences* 109(16), 5962–5966. <https://doi.org/10.1073/pnas.1116502109>.