A Declarative Framework for Constrained Clique Detection in Heterogeneous Knowledge Graphs via ASP

Mario Alviano[1,†], Eleonora Bilotta[2,†] and Pierpaolo Sestito[1,*,†]

[1]University of Calabria, Department of Mathematics and Computer Science
[2]University of Calabria, Department of Physics

Abstract

Modern data analysis requires tools that go beyond simple visualization, enabling users to explore and reason over complex relational structures. This work presents a data-driven framework designed to bridge the gap between interactive exploration and formal logic reasoning. At the core of the system is the integration of Answer Set Programming (ASP) as a real-time reasoning engine, which allows for the dynamic extraction of structural patterns and topological properties, including the Maximum Clique problem, within an interactive 3D environment. The proposed architecture adopts a declarative approach to manage the complexity of heterogeneous datasets. By leveraging a formal logic solver, the system enables semantic navigation and pattern discovery, ensuring that domain-specific constraints are consistently applied across the network's topology. This methodology allows researchers to interactively validate complex hypotheses and uncover latent clusters that would be difficult to identify through traditional imperative filtering. The effectiveness of the framework is demonstrated through a case study involving a historical and scientific dataset, showcasing how the symbiosis between automated reasoning and human-in-the-loop interaction enhances the process of knowledge discovery. This contribution highlights the potential of embedding logic programming paradigms into modern web-based analytical tools to provide a more rigorous and insightful exploration of complex systems.

Keywords

Declarative Problem, Network Analysis, Combinatorial Search, Clique Detection, Knowledge Graphs, Graph Encoding, Constraint-based Reasoning, Answer Set Programming, Logic Programming,

1. Introduction

Answer Set Programming (ASP) is a declarative programming paradigm based on nonmonotonic logic and stable model semantics [1, 2, 3, 4, 5]. It is particularly suitable for modeling combinatorial search problems, complex constraints, and reasoning tasks over heterogeneous data. For this reason, ASP has been successfully applied in several knowledge-intensive domains where the relevant information is structured, relational, and subject to domain-specific constraints [6, 7, 8, 9].

Knowledge Graphs (KGs) provide a natural representation for heterogeneous relational data, where entities are modeled as nodes and semantic relations as edges [10, 11]. In Digital Humanities and historical research, KGs are increasingly used to represent complex networks of people, institutions, places, and cultural movements [12, 13, 14, 15, 16, 17, 18]. However, the analysis of these graphs is not limited to data retrieval or visual inspection. Researchers often need to identify meaningful structural patterns under semantic or temporal constraints. For instance, in a historical network, one may be interested in groups of actors who are densely connected, active in a given period, and associated with a specific disciplinary or professional area [19].

A relevant example of this kind of task is clique detection. In graph theory, a clique is a set of nodes in which every pair of nodes is connected by an edge [20, 21]. Cliques can be used to identify strongly cohesive structures, such as intellectual circles, institutional clusters, or groups of actors sharing dense relational proximity [22, 23, 24]. Nevertheless, applying clique detection to historical Knowledge Graphs

CILC 2026: The 41st Italian Conference on Computational Logic, June 23–25, 2026, Ferrara, Italy
*Corresponding author.
†These authors contributed equally.
mario.alviano@unical.it (M. Alviano); eleonora.bilotta@unical.it (E. Bilotta); pierpaolo.sestito@unical.it (P. Sestito)
0000-0002-2052-2063 (M. Alviano); 0000-0002-1902-813X (E. Bilotta); 0009-0004-7685-6152 (P. Sestito)

is challenging because the graph structure alone is rarely sufficient [25]. The analysis often depends on additional dimensions, such as temporal intervals, node categories, professional roles, or domain-specific semantic filters.

Traditional graph algorithms, such as the Bron–Kerbosch family, are effective for purely structural clique enumeration, but they are less flexible when the analysis requires the integration of changing semantic constraints [26, 27, 28]. Adding such constraints often requires ad-hoc procedural modifications, making the resulting system harder to adapt, inspect, and reuse [29]. A declarative approach based on ASP offers a different perspective: the graph and its metadata can be translated into logical facts, while the analytical task can be expressed as a compact set of rules and constraints [30].

To address this problem, we present a browser-based, data-driven framework for interactive 3D Knowledge Graph exploration and ASP-based clique detection. The framework takes a JSON-based graph as input, optionally enriched with configuration parameters that define visual styles, temporal fields, semantic filters, and interaction behaviour. The current graph projection is translated into ASP facts and processed client-side through *clingo-wasm* executed inside a Web Worker. The resulting answer sets are then mapped back to the 3D visualization, allowing users to inspect the detected structures directly within the graph interface.

The framework is evaluated on a Knowledge Graph concerning the history of Italian psychology. The dataset contains 335 nodes and 1,365 edges, including people, institutions, places, scientific societies, and other contextual entities. We consider both the complete graph and a semantic projection filtered by the group *Psychology*, showing how the same ASP encoding can be applied to different graph contexts generated through user interaction.

Its novelty is not a new clique-detection algorithm or Knowledge Graph model, but the integration of visual exploration and declarative reasoning into a single browser-based loop: the analytical context fixes a graph projection, translated into ASP facts, solved client-side, and mapped back as answer sets. ASP thus acts as a reusable reasoning layer rather than an external offline step.

The main contributions of this work are:

- **An integrated visual-reasoning workflow:** a browser-based loop in which user interaction defines a graph projection, the projection is translated into ASP facts, and the resulting answer sets are mapped back to the 3D visualization.
- **A data-driven graph exploration framework:** a browser-based system that separates graph data, visual configuration, interaction mechanisms, and reasoning logic.
- **A client-side ASP reasoning workflow:** a pipeline that translates the current graph projection into ASP facts and executes reasoning tasks through *clingo-wasm* without requiring a dedicated backend.
- **Interactive clique detection over graph projections:** an implementation of maximum clique detection where the ASP instance depends on the currently selected temporal and semantic context.
- **A case study on a historical Knowledge Graph:** an evaluation on a heterogeneous dataset concerning Italian psychology, including both unfiltered and group-filtered reasoning scenarios.

2. Background

JSON Objects and JSONPath. JavaScript Object Notation (JSON) is a lightweight textual format for representing structured data through objects, arrays, and primitive values. A JSON object is delimited by curly braces and contains a set of key–value pairs, where keys are strings and values may be strings, numbers, booleans, arrays, or other objects. This makes JSON suitable for representing nested and heterogeneous information. Let $\mathcal{J}$ denote the set of JSON objects. JSONPath is a query notation for selecting values inside a JSON document by following paths through objects and arrays. In a path expression, object properties are reached through dot notation, while array elements can be selected through wildcard expressions. This notation is useful when the information needed by an application is not stored at the top level of an object, but inside nested structures.

Example 1 (Selecting nested values with JSONPath)**.** Consider the following JSON fragment, where the field `roles` contains an array of professional roles:

```
{
  "data": {
    "roles": [
      {
        "profession": "psychologist",
        "group": "Psychology"
      },
      {
        "profession": "psychiatrist",
        "group": "Medicine"
      }
    ]
  }
}
```

The JSONPath expression `$.data.roles[*].group` selects all values of the field `group` inside the `roles` array. In this case, it returns `Psychology` and `Medicine`.

Knowledge Graphs. In our setting, a Knowledge Graph (KG) is a graph whose nodes represent *entities* within a domain of interest, and whose edges represent relations among them. More formally, we represent a KG as a *JSON-labelled graph*, that is, as a pair (G, λ), where $G = (V, E)$ is a graph with nodes V and edges $E \subseteq V \times V$, and $\lambda : V \to \mathcal{J}$ is a node labelling function. For each node $v \in V$, the object $\lambda(v)$ describes the corresponding entity; we assume that $\lambda(v)$ contains the key `type`, whose value specifies the semantic type of the node (e.g., `person` or `place`), and possibly other data regarding domain-specific properties, in particular a temporal interval and a group of belonging. An *induced subgraph* is the graph obtained after selecting a subset of nodes according to the current analytical context. Such a context may depend on node types, temporal intervals, or groups. Formally, given a subset of vertices $V' \subseteq V$, the induced subgraph is denoted as $G' = (V', E')$, where $E' = \{\{u, v\} \in E \mid u, v \in V'\}$. The reasoning tasks considered in this paper, including clique detection, are performed over such induced subgraphs.

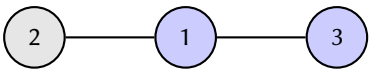

Figure 1: The induced structural graph from Example 2. Vertices 1 and 3 (blue) belong to the selected group g, while vertex 2 (gray) does not.

Example 2 (A JSON-labelled graph and an induced subgraph)**.** Consider the following JSON-labelled graph:

```
{
  "nodes": [
    {"label": "1", "type": "event",
      "data": {"active_from": 2018,"active_to": 2024}},
    {"label": "2", "type": "event",
      "data": {"active_from": 2019,"active_to": 2023}},
    {"label": "3", "type": "event",
      "data": {"active_from": 2020,"active_to": 2025}}
  ],
  "edges": [
    {"source": "1", "target": "2" },
    {"source": "1", "target": "3" }
  ]
}
```

The document represents a graph $G = (V, E)$, where $V = \{1, 2, 3\}$ and $E = \{\{1, 2\}, \{1, 3\}\}$. The labelling function λ associates each vertex with the corresponding JSON object in the `nodes` array. For instance, $\lambda(1)$ is the object with `label` equal to `"1"`, `type` equal to `event`, and temporal interval 2018–2024.

Now consider an analytical context that selects nodes active in the interval 2024–2025. Since a node is selected when its temporal interval overlaps the selected interval, vertices 1 and 3 are selected, while vertex 2 is excluded. Hence, $V' = \{1, 3\}$. The induced subgraph keeps only the edges whose endpoints both belong to V', therefore $E' = \{\{1, 3\}\}$. The resulting induced JSON-labelled subgraph is:

```
{
  "nodes": [
    {"label": "1", "type": "event",
      "data": {"active_from": 2018,"active_to": 2024}},
    {"label": "3", "type": "event",
      "data": {"active_from": 2020,"active_to": 2025}}
  ],
  "edges": [
    {"source": "1", "target": "3" }
  ]
}
```

Maximum Clique Problem and Answer Set Programming. In an undirected graph $G = (V, E)$, a clique is a subset of vertices $C \subseteq V$ such that every pair of distinct vertices in C is connected by an edge in E. The Maximum Clique Problem (MCP) consists of finding a clique with maximum cardinality. In this work, MCP is addressed by means of Answer Set Programming (ASP). (For details on ASP syntax and semantics we refer to the ASP-Core-2 standard format [31].) Specifically, an induced subgraph $G' = (V', E')$ is translated into a set $facts(G')$ of ASP facts: nodes in V' are represented by facts with predicate `node/1`, while undirected edges are represented by facts with predicate `edge/2`. If the graph must be further restricted to nodes within a given group g, we use the notation $facts_g(G')$ and extend the set $facts(G')$ with additional facts with predicate `group/2`. The facts representing the graph G' (and possibly the group g) are coupled with the following program:

```
% guess a subset of nodes
{ in(V) : node(V) } :- not group(_,_).  % either among all nodes
{ in(V) : group(V,_) } :- group(_,_).   % or among those in the specified group

:- in(U), in(V), U < V, not edge(U,V).  % all guessed nodes must be connected
#maximize { 1,V : in(V) }.              % maximize the number of guessed nodes
#show in/1.                             % show only the guessed nodes
```

Example 3 (ASP facts and answer sets)**.** Consider again the graph introduced in Example 2, where $V = \{1, 2, 3\}$ and $E = \{\{1, 2\}, \{1, 3\}\}$. The corresponding ASP facts are:

```
node(1).  edge(1,2).  edge(2,1).
node(2).  edge(1,3).  edge(3,1).
node(3).
```

When no group filter is selected, no `group/2` facts are generated. Therefore, the first choice rule of the encoding is active and the clique search ranges over all vertices of the current graph. Since vertex 1 is connected to both 2 and 3, but vertices 2 and 3 are not connected to each other, the maximum clique size is 2. The optimal answer sets are {`in(1)`, `in(2)`} and {`in(1)`, `in(3)`}.

Now assume that the current analytical context restricts the search to a selected group g, and that only vertices 1 and 3 belong to this group. This restriction is represented by adding the corresponding `group/2` facts `group(1,`g`)` and `group(3,`g`)`. When at least one `group/2` fact is present, the second choice rule of the encoding is active and the clique search ranges only over vertices occurring in `group/2` facts. Thus, vertex 2 is not considered as a candidate, even though it belongs to the graph. Since vertices 1 and 3 are connected, the only optimal answer set is {`in(1)`, `in(3)`}.

3. Data-Driven Architecture and Graph Model

The framework operates on a JSON-labelled graph described by two separate resources: a data JSON and a configuration JSON. The data JSON defines the graph structure, whereas the configuration JSON specifies how selected fields of node labels must be interpreted by the application. The configuration does not modify the graph; rather, it declares which metadata are used for temporal projection, semantic filtering, and ASP fact generation.

3.1. Graph Data Representation

The data JSON contains two main collections: `nodes` and `edges`. Each node must provide at least a human-readable `label` and a mandatory `type`. The optional field `id`, when present, is used as the basis for the internal node identifier; otherwise, the identifier is derived from the node label. In both cases, the value is normalized before being used internally. For example, the label `Alice Rossi` is converted into the identifier `alice_rossi`. The same normalization is applied to edge endpoints, so that `source` and `target` may refer either to explicit identifiers or to labels, provided that they normalize to the same internal form.

```
{
  "nodes": [
    {
      "id": "alice_rossi",
      "label": "Alice Rossi",
      "type": "actor",
      "data": {
        "active_from": 2018,
        "active_to": 2024,
        "affiliations": [
          {"group": "Network Analysis" }
        ]
      }
    },
    {
      "id": "bruno_verdi",
      "label": "Bruno Verdi",
      "type": "actor",
      "data": {
        "active_from": 2019,
        "active_to": 2023,
        "affiliations": [
          {"group": "Network Analysis" },
          {"group": "Cybersecurity" }
        ]
      }
    },
    {
      "id": "carla_bianchi",
      "label": "Carla Bianchi",
      "type": "actor",
      "data": {
        "active_from": 2020,
        "active_to": 2025,
        "affiliations": [
          {"group": "Social Analysis" },
          {"group": "Network Analysis" }
        ]
      }
```

```
    }
  ],
  "edges": [
    {"source": "alice_rossi", "target": "bruno_verdi" },
    {"source": "alice_rossi", "target": "carla_bianchi" },
    {"source": "bruno_verdi", "target": "carla_bianchi" }
  ]
}
```

Listing 1: An example data JSON

Example 4. Listing 1 shows a compatible data JSON. The graph contains three actor nodes, each annotated with a temporal interval and one or more group values. The three edges make the actors pairwise adjacent, forming a triangle in the structural graph.

3.2. Configuration and Metadata Mapping

The field `type` determines how a node is interpreted. More precisely, its value is used as a key into the `nodeTypes` object of the configuration:

$$\texttt{node.type} \mapsto \texttt{nodeTypes[node.type]}.$$

Thus, a node with `"type": "actor"` is interpreted according to the entry `nodeTypes.actor`. This entry declares the JSONPath expressions used to extract the metadata relevant to the application. The nested `data` object remains domain-specific: the framework does not impose a fixed schema, but relies on the configuration to identify the relevant fields.

```
{
  "nodeTypes": {
    "actor": {
      "timeline": {
        "startField": "$.data.active_from",
        "endField": "$.data.active_to"
      },
      "group": "$.data.affiliations[*].group"
    }
  }
}
```

Listing 2: An example configuration JSON for nodes of type actor

Example 5. Listing 2 shows a minimal configuration for nodes of type `actor`. The entry specifies that temporal information is extracted from `$.data.active_from` and `$.data.active_to`, while group values are extracted from `$.data.affiliations[*].group`. The key `affiliations` is not required by the framework; it is only part of this example dataset. Another dataset could use a different structure, provided that the configuration points to it through the corresponding JSONPath expressions.

3.3. Generating the Graph Projection and ASP Facts

For each node, the JSONPath expressions declared in the selected configuration entry are evaluated on the node object. In the example, all nodes have type `actor`; therefore, all of them are interpreted using `nodeTypes.actor`. The extracted temporal intervals are used to compute the graph projection shown by the timeline, while the extracted group values are used to derive semantic filters and, when selected by the user, to generate `group/2` facts for the ASP encoding. Nodes whose type has no temporal mapping are not interpreted as temporal nodes by the timeline mechanism.

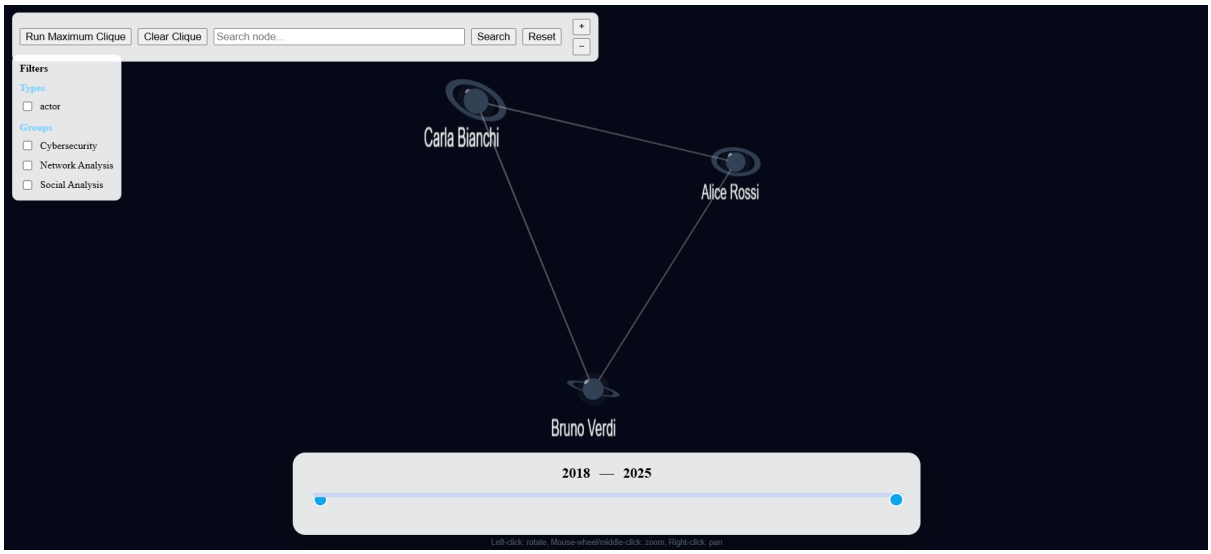

Figure 2: Visualization of the minimal JSON-labelled graph defined in Listing 1. The three actor nodes form a triangle in the structural graph and can be inspected through the interactive web interface.

Figure 3: Complete visualization of the Italian psychology Knowledge Graph.

Edges are interpreted structurally. The clique encoding uses only adjacency information: each visible edge contributes to the ASP instance independently of any domain-specific relation label. Visible nodes are encoded as `node/1` facts, and visible edges are encoded as symmetric `edge/2` facts. Since the three actors in Listing 1 are pairwise connected, they may be selected together by the maximum clique encoding after normalization and projection.

Figure 2 shows how the minimal graph is rendered by the web application. The visualization reflects the triangular topology declared in the data JSON, while the timeline and semantic filters are derived from the configuration-dependent metadata described above. In particular, the values of `type` are used to populate the type filters, whereas the group values extracted through `$.data.affiliations[*].group` are used to populate the semantic filters shown in the interface.

4. Case Study: The Italian Psychology Knowledge Graph

We evaluated the framework on a heterogeneous Knowledge Graph concerning the history of Italian psychology and its scientific, medical, institutional, and cultural context. The graph consists of 335 nodes and 1,365 edges. Its entities are organized into two main types: 275 nodes of type `person` and 60 nodes of type `place`. Person nodes encode historical actors together with temporal and semantic metadata, while place nodes represent contextual entities such as cities, institutions, scientific societies, universities, hospitals, and research centres. Figure 3 shows the complete graph as rendered by the interactive visualization interface.

4.1. Domain Modeling: Data and Configuration

Listing 3 reports a simplified excerpt of the data JSON. The full version includes long textual descriptions, multiple professional roles, temporal attributes, geographical information, and dense relations among people and institutions.

```
{
  "nodes": [
    {
      "id": "mario_ponzo",
      "label": "Mario Ponzo",
      "type": "person",
      "data": {
        "anno_nascita": 1882,
        "anno_morte": 1960,
        "periodo": "Nascita psicologia scientifica",
        "luogo_nascita": "Milano",
        "roles": [
          {
            "professione": "psicologo",
            "group": "Psychology",
            "sotto_area": "Psicologia generale"
          }
        ]
      }
    },
    {
      "id": "societ_italiana_di_psicologia",
      "label": "Societa italiana di psicologia",
      "type": "place",
      "data": {
        "citta": "Italia",
        "latitudine": 41.030454,
        "longitudine": 14.742209
      }
    }
  ],
  "edges": [
    {"source": "mario_ponzo", "target": "societ_italiana_di_psicologia" }
  ]
}
```

Listing 3: Simplified excerpt of the Italian psychology Knowledge Graph

To process this dataset, the configuration maps the `person` and `place` types to application logic (Listing 4). Nodes of type `person` are interpreted as temporal nodes: their active interval is extracted from the birth and death year fields, while their group metadata are extracted from the nested roles

array via the JSONPath `$.data.roles[*].group`. Nodes of type `place` are interpreted as spatial nodes and are kept available as graph context whenever connected to visible temporal nodes.

```
{
  "nodeTypes": {
    "person": {
      "category": "temporal",
      "timeline": {
        "startField": "$.data.anno_nascita",
        "endField": "$.data.anno_morte"
      },
      "group": "$.data.roles[*].group"
    },
    "place": {
      "category": "spatial",
      "alwaysVisible": true
    }
  }
}
```

Listing 4: Relevant configuration excerpt for the Italian psychology case study

4.2. Clique Detection Scenarios

This case study combines structural, temporal, and semantic heterogeneity. For this reason, we evaluated clique detection under different graph projections and semantic filters. In particular, we considered two structural projections: the subgraph induced by nodes of type `person`, and the complete graph including both `person` and `place` nodes. The first projection focuses on relations among historical actors, whereas the second also includes contextual entities such as institutions, universities, hospitals, scientific societies, and research centres. Semantic filters are derived from the group metadata extracted from the `roles` array of person nodes. The most frequent groups include *Medicine*, *Psychology*, *Academia*, *Arts & Culture*, *Social Sciences*, and *Philosophy*.

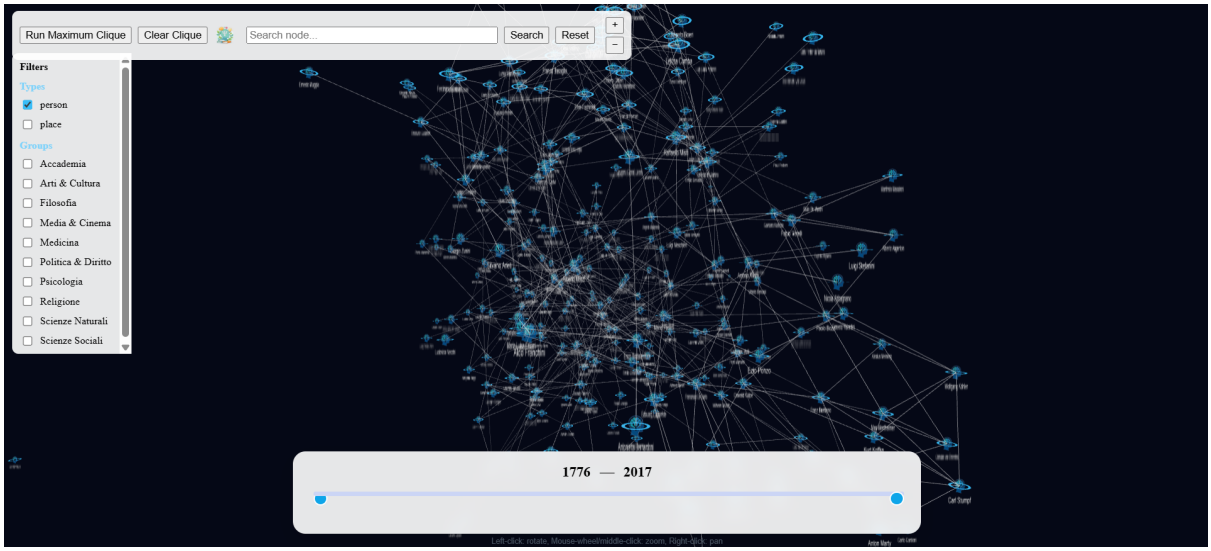

Figure 4: Visualization of the subgraph induced by nodes of type `person`.

Before running the ASP encoding, we computed basic graph statistics for each projection in order to characterize the induced graphs obtained through type and semantic filtering. Table 1 reports the number of nodes, edges, and connected components for each person-only and person–place projection.

Table 1
Graph statistics for person-only induced graphs and complete person–place graphs.

Scenario	P nodes	P edges	P comp.	P+L nodes	P+L edges	P+L comp.
All	275	1047	6	335	1365	1
Academia	67	115	18	69	117	19
Arts & Culture	34	27	20	34	27	20
Philosophy	29	57	5	29	57	5
Medicine	131	357	12	182	530	16
Psychology	88	172	25	95	191	25
Social Sciences	33	29	17	33	29	17

Table 2
Clique detection results over person-only induced graphs and complete person–place graphs.

Scenario	P cliques	P max	P+L cliques	P+L max	Δ cliques
All	192	4	505	5	+313
Academia	20	4	21	4	+1
Arts & Culture	3	3	3	3	0
Philosophy	4	3	4	3	0
Medicine	45	3	146	4	+101
Psychology	29	4	41	4	+12
Social Sciences	2	3	2	3	0

For each group, the framework generates the corresponding ASP instance from the currently selected graph projection and computes the clique structures exported by the solver. Table 2 summarizes both the clique statistics and the execution times. The reported times correspond to the client-side interval between the submission of the generated ASP program to the *clingo-wasm* Web Worker and the reception of the resulting answer sets by the application.

In Table 2, **P** denotes the projection induced by nodes of type `person`, while **P+L** denotes the complete projection including both `person` and `place` nodes. Figure 4 shows the person-only induced subgraph used as one of the structural projections in the comparison. The comparison shows that contextual nodes substantially affect the density of the resulting clique structures. On the complete graph, the framework exports 505 cliques, with maximum size 5, whereas the person-only projection produces 192 cliques, with maximum size 4. This indicates that contextual entities do not merely add peripheral information, but may participate in dense relational structures together with historical actors.

The graph statistics in Table 1 help explain the clique results in Table 2. For instance, the complete graph is connected, whereas the person-only projection contains six connected components. This shows that contextual `place` nodes contribute to the global connectivity of the Knowledge Graph.

The effect is particularly visible in the *Medicine* group. In the person-only projection, the framework exports 45 cliques, all of size 3. When `place` nodes are included, the number of exported cliques increases to 146 and the maximum size becomes 4. This suggests that medical institutions, hospitals, clinics, and university structures play an important role in connecting actors within this part of the graph. Figure 5 shows one maximum clique in the complete person–place projection for this group. The example illustrates how the framework can expose dense actor–institution configurations that would not appear in the same form in the person-only projection.

The *Psychology* group provides a second relevant scenario. The person-only projection produces 29 cliques, with maximum size 4, while the complete person–place projection produces 41 cliques, still with maximum size 4. In this case, adding contextual nodes does not increase the maximum clique size, but it reveals additional dense configurations involving scientific societies, schools, and institutional contexts. For instance, the complete projection includes cliques involving entities such as `societ_italiana_di_psicologia` and `scuola_magistrale_ortofrenica_di_firenze`, together with

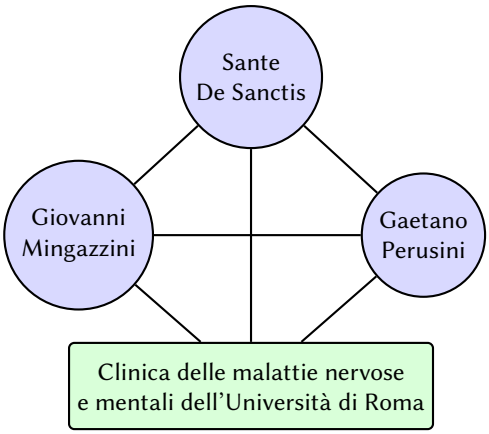

Figure 5: A maximum clique in the complete person–place projection for the *Medicine* group. The clique includes three person nodes and one place node.

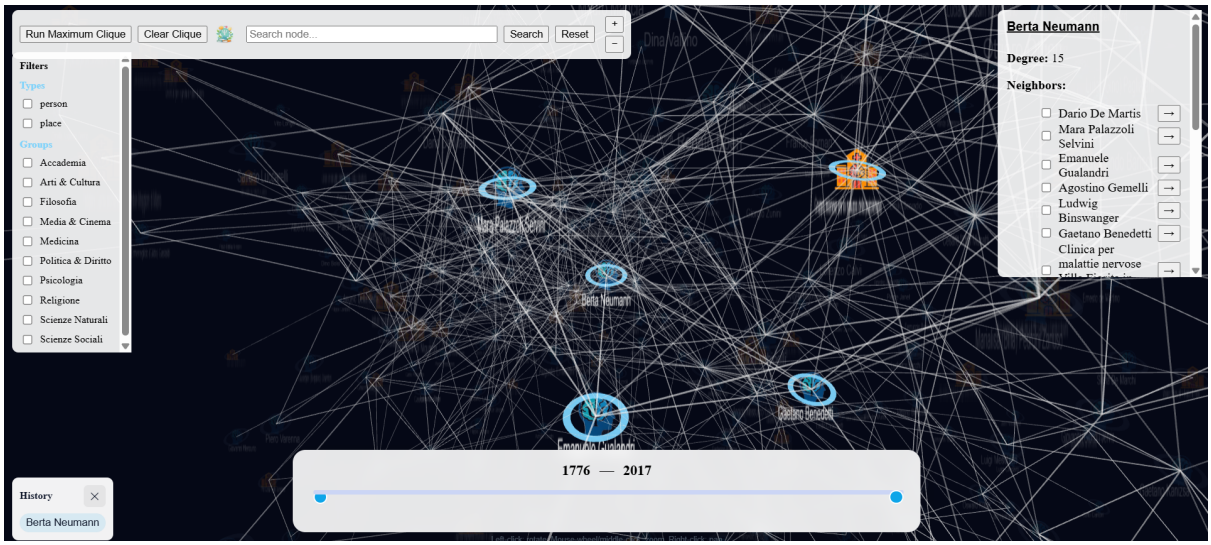

Figure 6: Visualization of a selected clique mapped back to the 3D graph and inspected through the information panel and neighborhood navigation tools.

person nodes associated with the development of Italian psychology.

Overall, these scenarios show how the same ASP encoding can be reused over different graph projections generated by the interface. By changing the selected node types and semantic groups, the user obtains different induced graphs and different clique statistics, while the declarative reasoning layer remains unchanged. This supports the intended use of the framework as an exploratory environment where structural and semantic hypotheses can be tested interactively over heterogeneous Knowledge Graphs.

4.3. Interactive Exploration and Reproducibility

The purpose of this case study is not to claim that each detected clique directly corresponds to a historically defined school. Rather, clique detection acts as an exploratory reasoning task to identify dense relational structures. The framework maps solver results back to the 3D graph, where the selected clique can be visually highlighted and analyzed through the information panel and neighborhood navigation tools (Figure 6).

For reproducibility, the complete interactive case study is publicly available online at http://pierpaolosestito-dev.github.io/KliqueASP/PsychologyHistoryGraph. This live version allows read-

ers to reproduce the visualization workflow and examine the clique detection results discussed here. The underlying data JSON, clique exports, and additional synthetic scalability tests are also available in the companion repository.[1]

Finally, the web application exposes debugging utilities through the browser console. The command *dumpAsp()* prints the ASP facts generated from the currently loaded graph, while *dumpAspFile()* exports a complete `.asp` file containing the generated facts together with the clique-detection encoding. These utilities allow users to inspect the exact ASP instance submitted to *clingo-wasm*, verify the node-index mapping, and reproduce the reasoning task outside the visual interface if needed.

5. Related Works

Knowledge Graph visualization has been widely studied as a way to support the exploration of heterogeneous and relational data. Existing works address both general visualization challenges and domain-specific interfaces for making graph-structured knowledge accessible to users. Nararatwong et al. [32] discuss challenges and framework-level requirements for Knowledge Graph visualization, while Li et al. [25] analyze practical user needs and identify visualization opportunities related to exploration, analysis, domain-specific interfaces, and temporal views. Several systems focus on specific domains, including education [33, 34], digital heritage [35], genealogy [36], biomedical and health-related collections [37, 38, 39], and scientific literature [40, 41]. These works show the importance of visual interfaces and Knowledge Graph infrastructures for inspecting, integrating, and contextualizing heterogeneous data, but they mainly focus on representation, navigation, querying, or domain-specific exploration.

A related line of work concerns the construction, manipulation, and validation of Knowledge Graphs through reusable data-processing infrastructures. Agocs and Le Goff [42] propose a RESTful service based on JSON Schema and JSON Meta Schema to support the construction and validation of Knowledge Graphs. KGTK [43] provides a data-science-oriented toolkit for large-scale Knowledge Graph manipulation and analysis, representing graph data in tabular form and supporting operations over resources such as Wikidata, DBpedia, and ConceptNet. These works are relevant to the data-driven perspective of our framework, since they highlight the need for practical representations and processing pipelines for Knowledge Graphs. However, their focus is primarily on construction, validation, transformation, and large-scale manipulation, rather than on embedding declarative reasoning into an interactive visual analysis loop. Other approaches connect Knowledge Graphs with visual analytics workflows. QueDI [44] supports the transition from Knowledge Graph querying to data visualization, with the goal of making Linked Open Data more accessible to users who are not necessarily experts in SPARQL. KG4Vis [45] uses Knowledge Graphs to model relations between data features and visualization choices, supporting visualization recommendation. Wiens et al. [46] investigate the use of Knowledge Graphs for customizable chart visualizations of tabular data. VisCARS [47] uses Knowledge Graphs for context-aware visualization and monitoring dashboards. CEPV [48] addresses information extraction and visualization for large Knowledge Graphs, while GeoGraphViz [49] proposes a geographically constrained 3D force-directed layout for spatially grounded Knowledge Graphs. These systems are close to our work in their attention to interactive exploration and visualization-oriented analysis, but they do not use a declarative logic program as an explicit reasoning layer over the current graph projection.

Several tools specifically target the exploration and visualization of Knowledge Graphs and ontological structures. KG-Visual [50] provides a tool for visualizing RDF Knowledge Graphs, while Luo et al. [51] present a Knowledge Graph visualization and retrieval system based on a joint extraction model. KG Explorer [52] focuses on customizable exploration of Knowledge Graphs. The Whyis Knowledge Explorer [53] also addresses customizable Knowledge Graph visualization, emphasizing the need to adapt the visual representation to the user and to the structure of the graph. Cox et al. [54] present an interactive biomedical query language and web application interface for federated Knowledge Graphs.

[1] https://github.com/pierpaolosestito-dev/KliqueASP/tree/e92caedf328166e96ccf382d15123673f60b9b10/static/PsychologyHistoryGraph/results

These systems provide important examples of user-oriented Knowledge Graph exploration, but their interaction models are mainly centered on browsing, querying, customization, or explanation. In contrast, our framework treats the currently selected graph projection as the input of a declarative reasoning task whose results are mapped back into the visual environment.

Answer Set Programming has been successfully used for reasoning and optimization over structured and relational knowledge. In industrial settings, Chu et al. [28] combine Knowledge Graphs and Answer Set Programming for supplier optimization. Mofatteh et al. [55] integrate Answer Set Programming and Knowledge Graphs in an adaptive transportation planning model based on real-time driver knowledge. More generally, ASP is well suited to encode combinatorial search problems and domain constraints in a compact and inspectable form. These works demonstrate the value of ASP as a reasoning technology over knowledge-rich data, but the reasoning process is typically presented as a problem-solving component rather than as part of an interactive visual exploration loop.

Our work is positioned between these lines of research. It does not propose a new Knowledge Graph model, a new visualization recommendation method, or a new clique-detection algorithm; instead, its novelty is the operational integration of these dimensions into a browser-based declarative visual analytics loop. Instead, it introduces a browser-based framework in which a JSON-labelled graph and a separate configuration JSON drive visual exploration, temporal projection, semantic filtering, ASP fact generation, and client-side solving. The ASP instance is generated from the graph projection currently selected by the user, and the resulting answer sets are mapped back into the visual environment. In this sense, ASP is not used as an external offline solver, but as a declarative reasoning layer embedded into the interaction loop. Clique detection is used as a concrete combinatorial task to evaluate this integration. Previous works have studied cliques and dense structures in Knowledge Graphs and scientific communities [56], as well as clique-based and community-oriented methods in network analysis. However, the contribution of this paper is not the optimization of clique enumeration itself. The maximum clique problem is used as a representative ASP task to show how structural, temporal, and semantic constraints can be expressed declaratively over an interactively generated graph projection and immediately reintegrated into the visual analysis workflow.

6. Conclusions and Future Work

This paper presented a browser-based, data-driven framework for the interactive exploration of heterogeneous Knowledge Graphs through ASP-based reasoning. The main contribution lies in embedding declarative reasoning directly within an interactive visual analytics loop. Instead of treating graph analysis as a separate offline computation, our framework allows users to define an analytical context through visual, temporal, and semantic filters, execute reasoning tasks on the resulting graph projection, and inspect the detected structures directly within the 3D visualization. The case study on the Italian psychology Knowledge Graph demonstrates the feasibility of this approach on heterogeneous historical datasets. We showed that combining structural logic with semantic filtering (such as isolating the *Psychology* group) narrows the reasoning task, making the resulting combinatorial structures more relevant and easier to inspect. While the current implementation is intentionally limited to maximum clique detection to evaluate the end-to-end pipeline (graph projection, ASP fact generation, client-side solving via *clingo-wasm*, and visual feedback), the results suggest that declarative reasoning provides a flexible layer for exploratory graph analysis.

Future work will focus on generalizing the framework beyond the specific maximum-clique encoding considered in this paper, by allowing alternative ASP encodings for graph patterns based on user-defined structural constraints, node attributes, edge attributes, and relation-specific properties. The next step is to formalize the system as a reusable graph-oriented component that can be ported to ASP-Chef [57] as a dedicated ingredient. This integration will create a broader recipe-based workflow where graph data can be dynamically loaded, transformed, reasoned upon with custom ASP encodings, and visually inspected entirely within the ASP-Chef environment.

Acknowledgments

This work was supported by the Italian Ministry of Health (MSAL) under POS projects CAL.HUB.RIA (CUP H53C22000800006) and RADIOAMICA (CUP H53C22000650006); by the Italian Ministry of Enterprises and Made in Italy under project STROKE 5.0 (CUP B29J23000430005); under PN RIC project ASVIN "Assistente Virtuale Intelligente di Negozio" (CUP B29J24000200005); by Regione Calabria under project NextGenGuides "Innovazione Tecnologica per le Guide Turistiche del Futuro tramite l'ausilio di tecnologie innovative AI e sistemi di geolocalizzazione avanzata" (CUP J39I24002040005); under project MOZART "Modelli e tecniche di mOdernizzazione di applicaZioni e data silos a supporto di clienti esterni, field force e impiegati di backoffice basati su AI GeneRativa e apprendimento auTomatico" (CUP J49I24001740005); by the LAIA lab (part of the SILA labs). Mario Alviano is a member of Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT for grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (2011) 92–103. doi:10.1145/2043174.2043195.

[2] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Mag. 37 (2016) 53–68.

[3] V. Lifschitz, Answer Set Programming, Springer, 2019.

[4] R. Kaminski, J. Romero, T. Schaub, P. Wanko, How to build your own asp-based system?!, Theory Pract. Log. Program. 23 (2023) 299–361.

[5] M. Alviano, C. Dodaro, S. Fiorentino, A. Previti, F. Ricca, ASP and subset minimality: Enumeration, cautious reasoning and muses, Artif. Intell. 320 (2023) 103931.

[6] F. Wotawa, On the use of answer set programming for model-based diagnosis, in: H. Fujita, P. Fournier-Viger, M. Ali, J. Sasaki (Eds.), IEA/AIE 2020, Kitakyushu, Japan, September 22-25, 2020, Proceedings, volume 12144 of *LNCS*, Springer, 2020, pp. 518–529. doi:10.1007/978-3-030-55789-8_45.

[7] R. Taupe, G. Friedrich, K. Schekotihin, A. Weinzierl, Solving configuration problems with ASP and declarative domain specific heuristics, in: M. Aldanondo, A. A. Falkner, A. Felfernig, M. Stettinger (Eds.), Proceedings of the 23rd International Configuration Workshop (CWS/ConfWS 2021), Vienna, Austria, 16-17 September, 2021, volume 2945 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 13–20. URL: https://ceur-ws.org/Vol-2945/21-RT-ConfWS21_paper_4.pdf.

[8] F. Wotawa, D. Kaufmann, Model-based reasoning using answer set programming, Appl. Intell. 52 (2022) 16993–17011. URL: https://doi.org/10.1007/s10489-022-03272-2. doi:10.1007/S10489-022-03272-2.

[9] L. M. Prikler, F. Wotawa, Faster diagnosis with answer set programming (short paper), in: I. Pill, A. Natan, F. Wotawa (Eds.), 35th International Conference on Principles of Diagnosis and Resilient Systems, DX 2024, Vienna, Austria, November 4-7, 2024, OASIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 24:1–24:13. URL: https://doi.org/10.4230/OASIcs.DX.2024.24. doi:10.4230/OASICS.DX.2024.24.

[10] J. Pujara, H. Miao, L. Getoor, W. Cohen, Knowledge graph identification, in: International semantic web conference, Springer, 2013, pp. 542–557.

[11] S. Rezayi, H. Zhao, S. Kim, R. Rossi, N. Lipka, S. Li, Edge: Enriching knowledge graph embeddings

with external text, in: Proceedings of the 2021 conference of the North American Chapter of the association for computational linguistics: human language technologies, 2021, pp. 2767–2776.
[12] E. Iglesias, S. Jozashoori, M.-E. Vidal, Scaling up knowledge graph creation to large and heterogeneous data sources, Journal of Web Semantics 75 (2023) 100755.
[13] Z. Li, H. Liu, Z. Zhang, T. Liu, N. N. Xiong, Learning knowledge graph embedding with heterogeneous relation attention networks, IEEE Transactions on Neural Networks and Learning Systems 33 (2021) 3961–3973.
[14] S. Singh, M. Siwach, Handling heterogeneous data in knowledge graphs: a survey, Journal of Web Engineering 21 (2022) 1145–1186.
[15] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Computing Surveys (Csur) 54 (2021) 1–37.
[16] D. Fensel, U. Şimşek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, A. Wahler, Introduction: what is a knowledge graph?, in: Knowledge graphs: Methodology, tools and selected use cases, Springer, 2020, pp. 1–10.
[17] C.-M. Chen, B. Witt, C.-Y. Lin, A knowledge graph analysis tool of people and organizations to facilitate digital humanities research, Data Technologies and Applications 59 (2025) 82–110.
[18] M. Koho, H. Rantala, E. Hyvönen, Digital humanities and military history: analyzing casualties of the warsampo knowledge graph, in: CEUR Workshop Proceedings, volume 3232, CEUR, 2022, pp. 308–316.
[19] X. Chen, S. Jia, Y. Xiang, A review: Knowledge reasoning over knowledge graph, Expert systems with applications 141 (2020) 112948.
[20] J. L. Szwarcfiter, A survey on clique graphs, in: Recent advances in algorithms and combinatorics, Springer, 2003, pp. 109–136.
[21] F. S. Roberts, J. H. Spencer, A characterization of clique graphs, Journal of Combinatorial Theory, Series B 10 (1971) 102–108.
[22] T. S. Evans, Clique graphs and overlapping communities, Journal of Statistical Mechanics: Theory and Experiment 2010 (2010) P12037.
[23] L. Falzon, Determining groups from the clique structure in large social networks, Social networks 22 (2000) 159–172.
[24] M. G. Everett, S. P. Borgatti, Analyzing clique overlap, Connections 21 (1998) 49–61.
[25] H. Li, G. Appleby, C. D. Brumar, R. Chang, A. Suh, Knowledge graphs in practice: characterizing their users, challenges, and visualization opportunities, IEEE Transactions on Visualization and Computer Graphics 30 (2023) 584–594.
[26] D. R. Aloysius, Bron-kerbosch algorithm, 2012.
[27] Y.-W. Wei, W.-M. Chen, H.-H. Tsai, Accelerating the bron-kerbosch algorithm for maximal clique enumeration using gpus, IEEE Transactions on Parallel and Distributed Systems 32 (2021) 2352–2366.
[28] C. X. Chu, M. H. Gad-Elrab, T.-K. Tran, M. Schiller, E. Kharlamov, D. Stepanova, Supplier optimization at bosch with knowledge graphs and answer set programming, in: European Semantic Web Conference, Springer, 2023, pp. 200–204.
[29] A.-S. Himmel, H. Molter, R. Niedermeier, M. Sorge, Adapting the bron–kerbosch algorithm for enumerating maximal cliques in temporal graphs, Social Network Analysis and Mining 7 (2017) 35.
[30] G. Brewka, T. Eiter, M. Truszczyński, Answer set programming at a glance, Communications of the ACM 54 (2011) 92–103.
[31] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. Maratea, F. Ricca, T. Schaub, ASP-Core-2 input language format, Theory Pract. Log. Program. 20 (2020) 294–309. URL: https://doi.org/10.1017/S1471068419000450. doi:10.1017/S1471068419000450.
[32] R. Nararatwong, N. Kertkeidkachorn, R. Ichise, Knowledge graph visualization: Challenges, framework, and implementation, in: 2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), IEEE, 2020, pp. 174–178.
[33] K. Sun, Y. Liu, Z. Guo, C. Wang, Visualization for knowledge graph based on education data.,

International Journal of Software & Informatics 10 (2016).
[34] K. Sun, Y. Liu, Z. Guo, C. Wang, Eduvis: Visualization for education knowledge graph based on web data, in: Proceedings of the 9th International Symposium on Visual Information Communication and Interaction, 2016, pp. 138–139.
[35] C. S. Khoo, E. A. Tan, S.-G. Ng, C.-F. Chan, M. Stanley-Baker, W.-N. Cheng, Knowledge graph visualization interface for digital heritage collections, Information Technology and Libraries (2024).
[36] R. Wang, J. Deng, X. Guan, Y. He, A framework of genealogy knowledge reasoning and visualization based on a knowledge graph, Library Hi Tech 42 (2024) 1977–1999.
[37] X. He, R. Zhang, R. Rizvi, J. Vasilakes, X. Yang, Y. Guo, Z. He, M. Prosperi, J. Bian, Prototyping an interactive visualization of dietary supplement knowledge graph, in: 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), IEEE, 2018, pp. 1649–1652.
[38] X. He, R. Zhang, R. Rizvi, J. Vasilakes, X. Yang, Y. Guo, Z. He, M. Prosperi, J. Huo, J. Alpert, et al., Aloha: developing an interactive graph-based visualization for dietary supplement knowledge graph through user-centered design, BMC medical informatics and decision making 19 (2019) 150.
[39] R. B. Correia, J. C. Rozum, L. Cross, J. Felag, M. Gallant, Z. Guo, B. W. Herr, A. Min, J. Sanchez-Valle, D. Stungis Rocha, et al., myaura: a personalized health library for epilepsy management via knowledge graph sparsification and visualization, Journal of the American Medical Informatics Association 33 (2026) 167–181.
[40] R. Lin, Q. Fang, B. Wu, Hydro-graph: A knowledge graph for hydrogen research trends and relations, International Journal of Hydrogen Energy 45 (2020) 20408–20418.
[41] H. Li, L. Li, Q.-Q. Huang, S.-Y. Yang, J.-J. Zou, F. Xiao, Q. Xiang, X. Liu, R. Yu, Global status and trends of metabolomics in diabetes: A literature visualization knowledge graph study, World Journal of Diabetes 15 (2024) 1021.
[42] A. Agocs, J.-M. L. Goff, A web service based on restful api and json schema/json meta schema to construct knowledge graphs, in: 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), 2018, pp. 1–5. doi:10.1109/CITS.2018.8440193.
[43] F. Ilievski, D. Garijo, H. Chalupsky, N. T. Divvala, Y. Yao, C. Rogers, R. Li, J. Liu, A. Singh, D. Schwabe, P. Szekely, Kgtk: A toolkit for large knowledge graph manipulation and analysis, in: J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, L. Kagal (Eds.), The Semantic Web – ISWC 2020, Springer International Publishing, Cham, 2020, pp. 278–293.
[44] R. De Donato, M. Garofalo, D. Malandrino, M. A. Pellegrino, A. Petta, V. Scarano, Quedi: From knowledge graph querying to data visualization, in: E. Blomqvist, P. Groth, V. de Boer, T. Pellegrini, M. Alam, T. Käfer, P. Kieseberg, S. Kirrane, A. Meroño-Peñuela, H. J. Pandit (Eds.), Semantic Systems. In the Era of Knowledge Graphs, Springer International Publishing, Cham, 2020, pp. 70–86.
[45] H. Li, Y. Wang, S. Zhang, Y. Song, H. Qu, Kg4vis: A knowledge graph-based approach for visualization recommendation, IEEE Transactions on Visualization and Computer Graphics 28 (2021) 195–205.
[46] V. Wiens, M. Stocker, S. Auer, Towards customizable chart visualizations of tabular data using knowledge graphs, in: International conference on asian digital libraries, Springer, 2020, pp. 71–80.
[47] P. Moens, B. Volckaert, S. Van Hoecke, Viscars: Knowledge graph-based context-aware recommender system for time-series data visualization and monitoring dashboards, IEEE Transactions on Visualization and Computer Graphics (2024).
[48] S. Sheng, P. Zhou, X. Wu, Cepv: A tree structure information extraction and visualization tool for big knowledge graph, in: 2019 IEEE International Conference on Big Knowledge (ICBK), IEEE, 2019, pp. 221–228.
[49] S. Wang, W. Li, Z. Gu, Geographviz: Geographically constrained 3d force-directed graph for knowledge graph visualization, Transactions in GIS 27 (2023) 931–948.
[50] D. Ghosh, E. Rajabi, Kg-visual: a tool for visualizing rdf knowledge graphs, in: Research Conference on Metadata and Semantics Research, Springer, 2021, pp. 126–136.
[51] J. Luo, S. Yao, K. Miao, S. Zhou, Y. Feng, J. Xu, A knowledge graph visualization and retrieval

system based on joint extraction model, in: 5th International Conference on Computer Information Science and Application Technology (CISAT 2022), volume 12451, SPIE, 2022, pp. 481–485.
[52] T. Ehrhart, P. Lisena, R. Troncy, Kg explorer: a customisable exploration tool for knowledge graphs, in: VOILA 2021, International Workshop on the Visualization and Interaction for Ontologies and Linked Data, 2021.
[53] J. P. McCusker, Customizable knowledge graph visualization using the whyis knowledge explorer., in: VOILA@ ISWC, 2024, pp. 24–34.
[54] S. Cox, S. C. Ahalt, J. Balhoff, C. Bizon, K. Fecho, Y. Kebede, K. Morton, A. Tropsha, P. Wang, H. Xu, Visualization environment for federated knowledge graphs: development of an interactive biomedical query language and web application interface, JMIR Medical Informatics 8 (2020) e17964.
[55] M. Y. Mofatteh, R. Abbas, N. D. Seddoh, S. Sedhumadhavan, M. Thunyaluck, O. F. Valilai, A smart adaptive transportation planning model including real-time drivers' knowledge using answer set programming and knowledge graphs, Procedia Computer Science 253 (2025) 2358–2368.
[56] R. Fabre, O. Azeroual, P. Bellot, J. Schöpfel, D. Egret, Retrieving adversarial cliques in cognitive communities: a new conceptual framework for scientific knowledge graphs, Future internet 14 (2022) 262.
[57] M. Alviano, L. A. R. Reiners, W. Faber, ASP chef grows mustache to look better, Theory Pract. Log. Program. 25 (2025) 417–436. URL: https://doi.org/10.1017/s1471068425100094. doi:10.1017/S1471068425100094.