KnowledgePearls: Provenance-Based Visualization Retrieval

Holger Stitz, Samuel Gratzl, Harald Piringer, Thomas Zichner, and Marc Streit

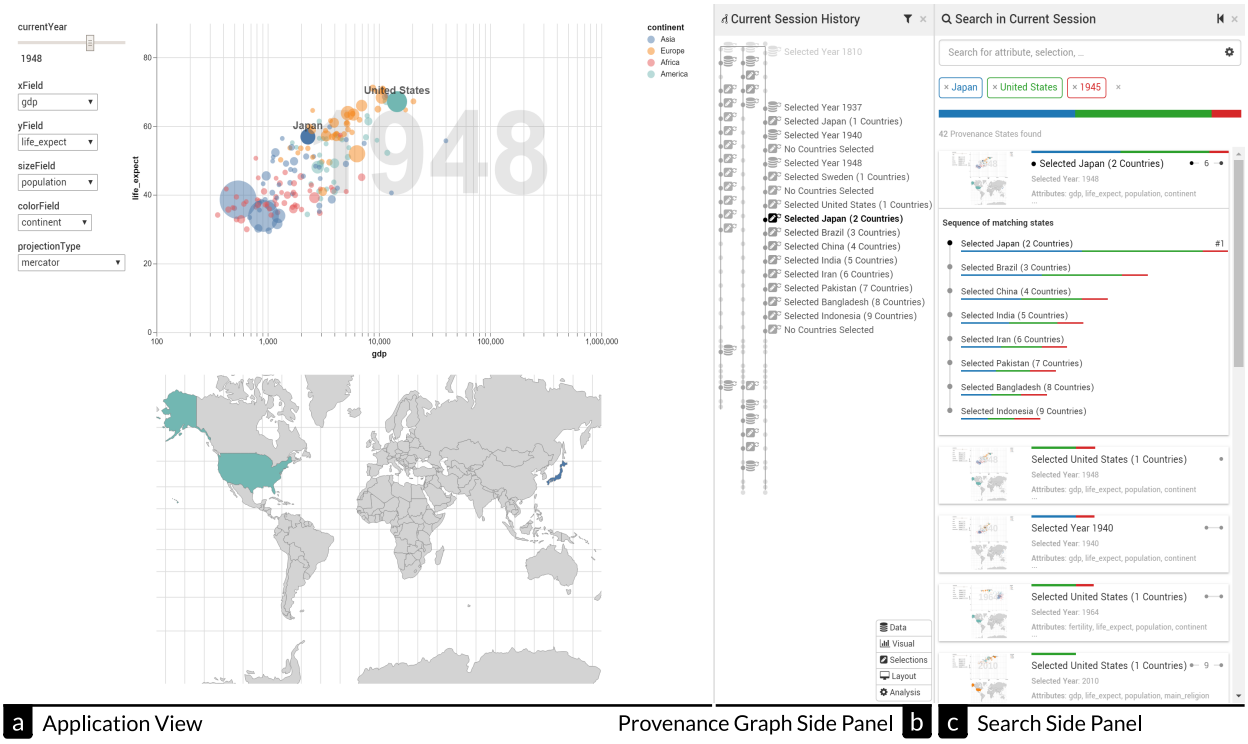

Fig. 1. Our *Gapminder*-inspired prototype implemented in *Vega* (a) with a captured provenance graph (b) consisting of three stories from Hans Rosling's presentations. Users can query the provenance graph for visualization states using the search side panel (c).

Abstract— Storing analytical provenance generates a knowledge base with a large potential for recalling previous results and guiding users in future analyses. However, without extensive manual creation of meta information and annotations by the users, search and retrieval of analysis states can become tedious. We present *KnowledgePearls*, a solution for efficient retrieval of analysis states that are structured as provenance graphs containing automatically recorded user interactions and visualizations. As a core component, we describe a visual interface for querying and exploring analysis states based on their similarity to a partial definition of a requested analysis state. Depending on the use case, this definition may be provided explicitly by the user by formulating a search query or inferred from given reference states. We explain our approach using the example of efficient retrieval of demographic analyses by Hans Rosling and discuss our implementation for a fast look-up of previous states. Our approach is independent of the underlying visualization framework. We discuss the applicability for visualizations which are based on the declarative grammar *Vega* and we use a Vega-based implementation of *Gapminder* as guiding example. We additionally present a biomedical case study to illustrate how *KnowledgePearls* facilitates the exploration process by recalling states from earlier analyses.

Index Terms—Visualization provenance, interaction provenance, retrieval.

1 Introduction

Visual exploration is a time-consuming and complex process. A single analysis session can consist of hundreds of individual steps. Over the past two years, we developed—together with our collaboration partners at a pharmaceutical company—a data-driven visual analysis system for identifying and prioritizing drug targets. All interactions within a session are tracked in a provenance graph. Capturing sessions in provenance graphs not only ensures reproducibility of findings, but also provides a growing knowledge base of the overall analysis state of the explored datasets. Further, visual exploration is usually not performed by a single domain expert alone but by a team in which each expert analyzes the dataset individually. To avoid repetitive analyses that do not lead to new findings and to increase the confidence in potential findings when identified by multiple experts independently, effective recall of captured provenance information becomes crucial. In addition to retrieval purposes, the growing knowledge base can be used to guide future analysis sessions by identifying unexplored areas of the exploration landscape. In this paper, we present a novel search and retrieval concept for visualization states stored in a provenance graph.

- *Holger Stitz, Samuel Gratzl, and Marc Streit are with Johannes Kepler University Linz, Austria. E-mail: {firstname.lastname}@jku.at.*
- *Samuel Gratzl is with datavisyn GmbH, Austria. E-mail: samuel.gratzl@datavisyn.io.*
- *Harald Piringer is with VRVis Research Center, Austria. E-mail: hp@vrvis.at.*
- *Thomas Zichner is with Boehringer Ingelheim RCV GmbH & Co KG, Austria. E-mail: thomas.zichner@boehringer-ingelheim.com.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

A visualization state is a snapshot of properties, including data properties, visual properties, and interaction properties as attribute-value pairs, at a certain juncture during analysis. The composition of properties depends on application, visualization types used, and interactions supported. Changing one or multiple properties creates a new visualization state that is pushed into the provenance graph.

Ragan et al. [29] characterize provenance information by *type* and *purpose*. According to their organizational framework, our approach operates on *interaction provenance*, which contains the history of user interactions with the system, and the tightly coupled *visualization provenance*, which comprises the history of visualization states. *KnowledgePearls* is designed to allow users to *recall* states visited in the analysis history. In addition, we see the purpose of our work extending to user guidance.

Our primary contribution is a solution for efficient retrieval of visual analysis states that are structured as provenance graphs of automatically recorded user interactions and visualizations. As a secondary contribution, we propose guidelines and requirements for existing visualization systems to support provenance tracking and retrieval and explain the integration of *KnowledgePearls* into a dashboard defined in the *Vega* visualization grammar [32].

At a glance, *KnowledgePearls* allows users to specify a search query either by definition (i.e., data attributes, selected items) or by example (i.e., from an active visualization state). All query aspects can be weighted based on user interest. To provide more meaningful search results, matching subsequent visualization states are grouped into state sequences (resembling a pearl necklace). We argue that this approach has the potential to minimize redundant analyses and to accelerate the process of analyzing the data.

We introduce *KnowledgePearls* and its integration using a *Gapminder*-inspired prototype implemented in *Vega* as guiding example. The loaded provenance graph contains all interactions Hans Rosling performed with the original *Gapminder* software in three different presentations [1][2][3]. Additionally, we demonstrate the scalability and effectiveness of our solution in a case study carried out by collaborators working in the field of cancer drug discovery.

2 Design Objectives

Based on literature reviews, interviews with our domain experts, and our own experience with provenance information in the context of visual analysis, we elicited a set of design objectives that an effective provenance-based retrieval approach should support.

O1 Support Heterogeneous Properties. A visualization state can contain properties with different data types, for instance, numerical or categorical. The retrieval approach must take this heterogeneity of data types into account when looking for matching visualization states.

O2 Support Fuzzy Search. Most provenance retrieval approaches support binary search that considers exact matches only. This limits the number of search results tremendously, but has the disadvantage of displaying only results of equal importance. However, users might be interested in search results that match the formulated query only partially. The output of such a fuzzy search is a collection of search results with varying importance.

O3 Consider Inherent Temporal Coherence. Each interaction leads to a new visualization state in the provenance graph. Due to temporal coherence, adjacent states of the same interaction sequence are typically very similar. In the case of a plain retrieval, all these similar states would be listed as separate search results, cluttering the result list. To alleviate this scalability problem, the retrieval approach should support appropriate aggregation and clutter reduction techniques.

With these objectives in mind we developed *KnowledgePearls*, a novel provenance-based retrieval process.

[1]Presentation by Hans Rosling, The Best Stats You've Ever Seen, 2007.
[2]Presentation by Hans Rosling, 200 Countries, 200 Years, 2010.
[3]Presentation by Hans Rosling, Religions and Babies, 2012.

3 Related Work

Information retrieval (IR) can be divided into two main parts: (1) building the search index from the input data, and (2) the retrieval process, which is targeted at finding items based on a search query. Depending on the data structure and the number of items, a broad variety of search capabilities and visual interfaces are available.

3.1 Index and Provenance

During the indexing phase, the IR system identifies and extracts features from the input data, and stores them for later retrieval. Features are, for instance, representative keywords describing a text document or tags and labels describing the content of an image or a video. Features of visualizations can be extracted in many ways, e.g., based on data metrics, image metrics, and parameters of the visualization pipeline (see Section 4.1). In general, indexing becomes easier and achieves better results for highly structured input data. Visualization grammars, as proposed by Wilkinson [45] or systems such as *Vega* [32, 33]/*Vega Lite* [31], or *Polaris/Tableau* [24, 42, 43], describe the visualization in a standardized and structured way. Hence, they are well suited to index and retrieval of visualizations. However, these grammars can represent only a snapshot of a visualization and offer limited interactions during the visual analysis, which form a valuable knowledge base for future exploration and recall [37]. Storing changes and interactions over time creates an interaction history [12, 19]. Capturing changes on additional levels results various types of provenance information [29] (e.g., data, visualization, and interaction provenance) that can also be used for retrieval purposes.

Provenance tracking, storage, and retrieval has also been an active research topic in the database community. Recent surveys by Herschel et al. [16] and Pérez et al. [28] summarize the state-of-the-art. However, most of the works focus on efficient algorithms to store and retrieve information, while the visualization of and interaction with the search results play a tangential role.

Khan et al. [18] propose a framework that records users search activities in an enterprise file repository. Users can explore the evolving search provenance graph by setting a time range and exploring the collection of matching files in a detail view. Their approach demonstrates how provenance can be used to improve search queries. However, the approach is not directly applicable to visualization retrieval, since files contain large amounts of unstructured text that require building and maintaining ontologies—which do not yet exist for visualization. Further, we focus on utilizing the inherently incremental characteristics of visualization states (**O3**).

Another exception in provenance retrieval is the well-studied field of workflow provenance graphs [41]. Due to workflow modifications or multiple executions with different parameters or input datasets, this type of provenance graph can rapidly grow very complex. Index and retrieval of workflow provenance have been discussed in work by Frew et al. [11] and Biton et al. [5]. Workflow provenance graphs, however, are very different from visual exploration provenance graphs. Workflows are pipelines of tools and files forming a graph structure. While in workflow provenance graphs multiple tools or input files are modified between two pipeline executions, the properties of visualization states are changing incrementally (**O3**).

3.2 Retrieval

Once the input data is available in the search index, users can start formulating search queries and sending them to the IR system for comparison with the index. We identified three strategies for formulating a search query:

1. **Query by Definition.** Queries can be created by selecting facets or classifications, by formulating a statement in system language, or via natural language.
2. **Query by Example.** Queries are implicitly defined by the user by selecting or creating an example. The query is then derived from the example.

3. **Query by Perception.** Users have a mental model of the query and interactively explore the data for possible results. Hence, the user knows which parts of the data and visualizations are of interest and how to proceed with the analysis.

Query by Definition

Writing a search query at the system level can be difficult for users. The query languages strongly depend on the underlying storage format (e.g., *XQuery*, *Prolog*, *SQL*, and *SPARQL*) and offer great versatility and flexibility [10, 11]. Furthermore, the syntax of these query languages is hard to memorize and tends to produce long statements. Easier access can be achieved by using custom perspectives (similar to database views) on the provenance data [5] or carefully designed query builder and search interfaces that abstract the complexity of a query language.

A common interface for filtering a large collection of items are facets, which are generated from an object's metadata or text extraction and can form a hierarchy or a list of classifications. Users can select one child element of the hierarchy at a time and thus narrow down the scope (i.e., limit the number of search results) [7, 38]. With this approach, however, only one item can be selected at a time, and including other parts from a different sub-tree is impossible. In contrast, using facet classification, users can select multiple classes and thus construct a binary search query [1, 36]. A limitation is that numerical values must be binned to derive categories. Further, objects are considered only if they match exactly. Another example of a Boolean search interface was presented by Heer et. al. [15] for *Polaris/Tableau* [42]. The selected chart types and data fields are stored as states in the provenance graph ("worksheet history") and can be employed by the user for filter operations. These approaches, however, allow users to search for data fields only and the results are restricted to exact matches. In *KnowledgePearls* we provide related search terms that enable users to narrow down the scope, provide a fuzzy search for properties, and let users weight the search terms based on their interests [14, 23].

With recent advances in natural language processing, formulating queries in natural language is becoming increasingly popular. Users can speak or write queries using their native language to interact with the visualization [39] or search in datasets, such as integrated in *IBM Watson Analytics*[4]. The queries, however, often need to follow a predefined structure or contain commands that can be translated into system language.

Query by Example

In contrast to writing the definition of a query, users can create or re-use existing parts of an entity as search query. The eponymous high-level database management language *Query-by-Example* by Zloof [47] is one of the earliest approaches that allows users to query, update, and control the database with little knowledge of the query language. Numerous example-based language have emerged since then [22].

A notable example of provenance retrieval in the visualization domain is *VisTrails* [3], which encourages users to re-use existing workflows [35]. In order to find these workflows, users can interactively build parts of a workflow as search query and thus avoiding having to learn a new query language. Matching workflow versions are displayed along with the highlighted part. Similarly, users can interactively construct visual graph queries in *Visage* [25]. Search results must match the structure of query graph and additional attributes attached to the nodes. Both approaches work well for a relatively small provenance graph and a limited number of search results. However, with a larger number of matching results, the presentation becomes cluttered.

Query by Perception

Both query by definition and query by example result in a query that serves as the input for retrieval. In contrast, query by perception relies on the user's visual system to identify contextually relevant states in a visualization of the provenance graph that encodes its properties. Examples are *AVOCADO* [41] and the taxonomy-based glyph design [20], which both visualize workflows of biological experiments. *InProv* [6] utilizes time-based hierarchical grouping for filesystem provenance and lets users explore file changes in an interactive radial-based tree layout.

Shrinivasan and van Wijk [37] propose a technique to capture visualization states and store them in an interaction history (i.e., provenance graph). Users can annotate and revisit states.

In the work on *CLUE*, Gratzl et al. [13] extend this approach by allowing users to assemble states of interest into a story that can then be used for presentation and recall.

As the explicit visualization of a large and quickly growing provenance graph is a limiting factor in terms of scalability, we decided to focus on a query by definition and query by example strategy in *KnowledgePearls*.

[4] https://www.ibm.com/watson-analytics

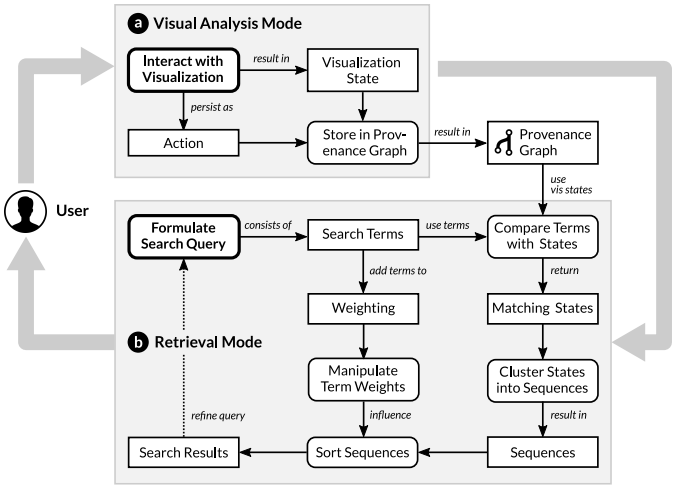

Fig. 2. The user can rapidly switch between two different modes: (a) User interactions change the visualization, and result in new visualization states, which are added to the provenance graph. (b) The user formulates a search query that is compared with the stored visualization states. Matching states are grouped into sequences and sorted according to the weighting of the search terms.

4 Provenance-Based Retrieval Approach

KnowledgePearls is designed as an extension to visual analysis tools that record user interactions in a provenance graph, with the goal to allow users to effectively query previous exploration states. With our approach, users can rapidly switch between two different modes, as illustrated in Figure 2: (1) **visual analysis mode** and (2) **retrieval mode**. In visual analysis mode (Figure 2a), new visualization states are continuously created with every user interaction and stored in a provenance graph. In retrieval mode (see Figure 2b), users can formulate search queries that are compared with all visualization states stored in the provenance graph. Matching states are grouped, ranked, and presented in a result list.

Users can switch seamlessly between these modes, that is, start a retrieval during an analysis and then continue the visual exploration from a selected search result.

4.1 Provenance Graph and Visualization States

When interacting with a visual analysis system, users trigger actions such as attribute changes, updates of filter settings, or item selections. Possible interactions in our *Gapminder*-inspired prototype are, for instance, choosing the data attributes mapped to the axes of the scatterplot, determining the projection type of the map, selecting countries shown as items in the plot, or determining the time point for which the scatterplot shows data.

The structure of the provenance graph as used in *KnowledgePearls* is based on the definition presented by Gratzl et. al [13]. Each user interaction triggered in the visualization results in a transformation of the visual representation. After updating the visualization, a snapshot of the selected attributes, filters, and items is automatically captured

as a visualization state. Both entities—the visualization state and the actions—are stored in a provenance graph that consists of visualization states as nodes and the corresponding actions as edges (see Figure 2a).

The information stored in the provenance graph can be used to restore any previously seen visualization state by applying all actions from the root of the provenance graph to the desired state. Branches in the graph emerge when users jump back to a previous state in order to continue the exploration from there. This back-tracking strategy is typical for exploratory data analysis scenarios.

Visualization State Properties

A visualization state consists of a title, metadata (e.g., creator and creation date), and multiple properties describing the visualization state (**O1**).

Choosing visualization properties to be stored is a non-trivial task, as they need to be tailored to the visualization (system) at hand. While the set of tracked properties will typically vary between domains, tasks, and visualization techniques, we provide a high-level semantic characterization of property types for guiding the process of defining meaningful properties for a specific system. Properties of the visualization state can coarsely be classified as data-related or visualization-related.

Data-related properties refer to aspects of the inspected data which are independent of the visual encoding. Important examples include:

- Data attributes which are mapped to a visual variable, e.g., color or an axis. For example, users may search for states where the attribute *GDP* is displayed.
- Data items which are represented by the visualization so that distinct items can be discriminated from each other. This may include data categories for data sets with nominal data attributes. For systems supporting data selection, an important specific type of property is the set of data items in focus. For example, users could search for states where the country *United States* is selected.

Visualization-related properties are derived from the current visual encoding and thus depend on the visualization technique. Important sub-types include:

- The visualization technique itself. For example, in systems which offer different visual encoding options (e.g., *Tableau*), users could search for states that contain a map or stacked bars.
- Visualization parameters which are set by the user. For example, searching for *logarithmic* could retrieve states where data is visualized by a logarithmic scale.
- Visualization metrics which quantify aspects of the visual encoding of the particular data. Important examples include *Scagnostics* [46] in case of scatterplots or *Pargnostics* [8] for parallel coordinates. For more information about quality metrics, we refer to Bertini et al. [4], who provide an overview and a systematization of their use in high-dimensional data visualization. In general, the selection of supported quality metrics depends on the visualization techniques, the data, and the task. From the perspective of our approach, the search based on quality metrics requires the visualization system to compute these metrics for (some of) the views of the current analysis state. The provenance graph needs to store quality metrics much like other numerical properties. In the Gapminder example, a user could, e.g., search for states with high values for the property *skinniness*.

Our *Gapminder* example includes data attributes and selected data items as data-related properties as well as visualization parameters and *Scagnostics* as visualization-related properties.

Relationships Between Properties

In our concept, each property of a visualization state is treated independently. However, depending on the visualization type, relationships between properties may exist, for instance, that both a data attribute and an axis scale define an axis in a scatterplot. In this version of our concept, we ignore potential relationships in favor of an easy-to-use search interface that can cover most of the use cases.

In early stages of our concept, we experimented with the definition of relationships between properties. An ontology is a formal way to describe the relationships between possible terms (e.g., that an axis contains attributes and a scale). Creating such an ontology that can be used for visualization states is an open research challenge, which is beyond the scope of this work. In response to early feedback from our target users to first versions of our prototype implementation, we deliberately decided to favor simplicity over being able to express more complex relationships between query terms.

For a detailed discussion of relationships between properties and hierarchical property structures, see Section 8.

4.2 Retrieval

In retrieval mode (Figure 2b), we utilize the captured provenance data for finding similar visualization states for a given search query. *KnowledgePearls* supports users in their current analysis in recalling earlier states, which facilitates collaboration between users. A retrieval can be performed either by the user who did the analysis in an earlier session or by a different user who searches, for instance, for similar visualization states in a provenance graph created by a colleague.

Retrieval starts with formulation of a search query that consists of one or multiple search terms. A search term can be either a string (e.g., *France*) or a property identifier followed by a numerical value (e.g., *monotonic = 0.3*) (**O1**).

Once the user has entered the search query, each search term is compared with the whole collection of visualization states stored in the provenance graph. The comparison step determines the relevance of a visualization state compared to the given search query. We use different comparison mechanisms based on the properties' data types: categorical, numerical, and set-typed. The result is a fuzzy search that presents the visualization states found according to a continuous spectrum from high to low relevance (**O2**).

Categorical properties are, for instance, displayed data attributes (e.g., *GDP*, *population*) and categorical visualization settings, such as the map project type (e.g., *mercator*, *orthographic*). To calculate the similarity between categorical properties, we index the property values and apply the *term frequency–inverse document frequency* (*tf–idf*) [30]. This measure is widely used in information retrieval and reflects the importance of a word in a document (in our case in a visualization state) with respect to the whole collection of documents (i.e., provenance graph). A word is more important when the term frequency (in the given visualization state) is high and the document frequency of the term in the whole collection of documents is low. Thus, we decrease the importance of commonly used values of categorical properties. For instance, in all recorded stories, Hans Rosling used the *population* of a country as the size of a visual mark in *Gapminder*. In contrast, he used the attribute *child mortality* in a single session only. Consequently searching for *child mortality* results in a higher similarity score in matching states than the search term *population*.

For **numerical properties**, such as the selected *year* in *Gapminder* and derived visualization metrics, we calculate the absolute difference between the query value and the state value. The smaller the difference between a numerical input value and the actual state value, the greater the importance of this property. For example, given two scatterplots with the selected years *2006* and *2003*, a search for *2005* would result in greater similarity to states from the *2006* set than to those from the *2003* set because the difference is only one year in the former case and two years in the latter.

Set-typed properties typically refer to selections of data items, for example, a set of brushed countries. Depending on the visualization type, multiple set-typed properties can exist in cases in which users can select multiple different entity types, such as countries or continents. We use the *Jaccard Index* to determine the similarity between two sets, that is, all set-typed input values and the set-typed properties of the visualization state. The higher the overlap between the compared sets, the more similar they are. In our *Gapminder* example, Hans Rosling selected *United States* along with *Vietnam* to demonstrate the economic

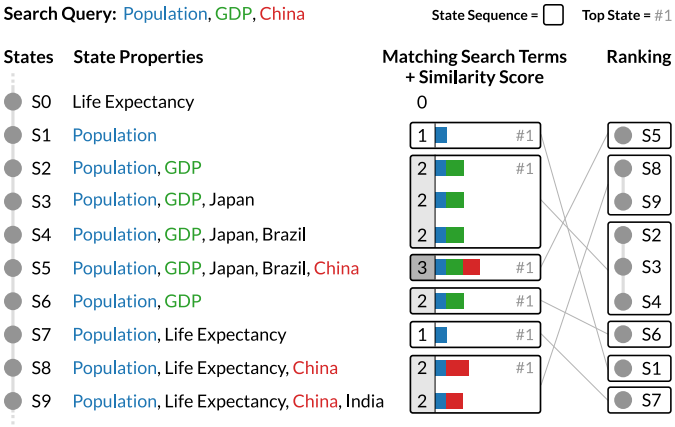

Fig. 3. A list of states, including their properties, is compared to a search query. We calculate the matching search terms for each state, and group subsequent states into state sequences. The ranking is defined by the number of matching terms and the weighted similarity score.

differences in 1964. In another presentation, he selected *United States* together with eight other countries. A search for *United States* would rank the first visualization state with the selection of only two selected countries higher than the second one.

The comparison step results in a *relevance score* within the range $[0,1]$ for each search term, where 1 denotes an exact match and 0 a non-matching search term. We normalize the tf–idf value for categorical search terms in that range since the tf–idf value can go beyond 1. Subsequently, the relevance of the state is calculated as the weighted sum of all relevance scores. The applied weights can be changed interactively by users according to their interest in each search term. This ensures that states of high interest also result in a higher rank. States that are equal to a value of 0 are considered to be non-matching and are excluded from further processing steps. Setting a higher threshold value to exclude less relevant states is also possible.

4.3 Grouping of Search Results

Since several visualization properties remain unchanged between multiple subsequent visualization states, displaying all matching ones individually results in a long list of highly similar search results (**O3**). To address this issue, we group visualization states into *state sequences*.

Our grouping algorithm determines matching terms, i.e., terms with a *relevance score* > 0, for every visualization state, and groups subsequent visualization states with an equal number of matching terms into sequences. States that have no matching search terms are excluded. This approach tends to create multiple short sequences, as shown in Figure 3. For instance, searching for *population* only results in one sequence containing states between *S1* and *S9* . Adding *GDP* as a second search term splits the sequence into three sequences. Adding *China* as a third search term results in six sequences, four of which contain only a single state.

In addition to the grouping, we identify a *top state* for each sequence. Although the number of matching search terms remains constant for all states of a sequence, the similarity score for each state may vary within a sequence because of the different comparison mechanisms that depend, for instance, on the remaining property values in the state (see Section 4.2). We consider the state with the highest similarity score, or in the case of multiple candidates the first candidate (see *S2* to *S4* in Figure 3), in a sequence as the top state. The top state is used as a representative of the sequence within the search result.

We utilize the top results to order the sequences by the weighted similarity score. This yields a ranking in which relevant sequences—those that match multiple search terms—are ranked more highly. For instance, in Figure 3, state *S5*, which matches the whole search query, is ranked the highest, followed by the sequence with states *S8* and *S9*, which have slightly better similarity scores than other sequences matching two search terms.

In summary, the grouping algorithm provides a trade-off between presenting individual relevant states (i.e., maximizing the number of search results) and longer sequences (i.e. minimizing the number of search results).

5 Visualization and User Interaction

Our prototype implementation consists of three views, as shown in Figure 1: The **application view** (a), the **provenance graph side panel** (b), and the **search side panel** (c). The application view contains the visualization system (e.g., *Gapminder*), with which users interact in visual analysis mode (see Figure 2). On the right side, users can open the provenance graph and the search side panel on demand. The provenance graph side panel (labeled "Current Session History" in our prototype) provides a visualization of all recorded states [13] (see Figure 1b). Interactions from the application view, which are added to the provenance graph, instantly appear in this side panel (Figure 1c). The user can jump back to previous states and continue the analysis from there. The search side panel is linked with the provenance graph side panel and contains a search field (Figure 4a), selected search terms as query (d), a weighting editor (e), and a list of search results (f).

Below, we explain individual views and their functionality in more detail. As a guiding example, we use a provenance graph that is based on selected presentations given by Hans Rosling. We focus on the parts of the presentations in which he mentioned the development of *Japan* and the *United States* at the end of World War II in *1945*.

5.1 Search Field and Weighting Editor

Users can formulate the search query by entering search terms in the input field shown at the top of the side panel (see Figure 4a). Upon entering the first term (e.g., *Japan*), a drop-down list of suggestions is presented below the input field. The list contains only property values that occur in the provenance graph and thus have been involved in the exploration. Suggestions are grouped by property names (e.g., *data attributes*, *Scagnostics*) and their possible values (e.g., *GDP*, *China*, *density = ?*) (**O1**). We allow users to search for property names and values, and highlight the matching part of the string. In addition, users can search for metadata, such as author and timestamp of creation of a visualization state. If property values require further input, for instance, the reference value for the *Scagnostics* measures, we indicate the input type (e.g., numerical) and validate the input before accepting the search term (**O2**).

Taking the provenance graph created from Hans Rosling's presentations as basis and entering the string "Ja" for the search term *Japan* will show a drop-down list with only three possible countries—*Japan*, *Azerbaijan*, and *Jamaica*. From this filtered list, *Japan* is the only country that is active and can be selected. The other two countries do not feature in the provenance graph, and are therefore disabled to prevent a worthless, empty search result list. By default, we only list countries that are contained in the provenance graph at least once. Countries that are defined in the dataset but are not in the provenance graph can be optionally added to the suggestions as context.

For the string "United" entered as part of the second search term *United States*, all three suggestions—*United States*, *United Kingdom*, and *United Arab Emirates*—are active and can be selected. The countries are sorted in descending order by their frequency (i.e., the number of occurrences in all recorded visualization states). We indicate the frequency by the length of a bar shown next to the property value (see Figure 4c). In our guiding example, the retrieval returns 24 visualization states for *United States*, 2 states for *United Kingdom*, and 2 states for *United Arab Emirates*.

Derived Properties

We show the ten most frequent terms and property values in a *Top 10* group at the top of the result list. Without the need for a specific search term to be entered, the *Top 10* group provides an immediate summary of the most used properties in the provenance graph. In our guiding example, the most frequent items are presented in decreasing order of their frequency.

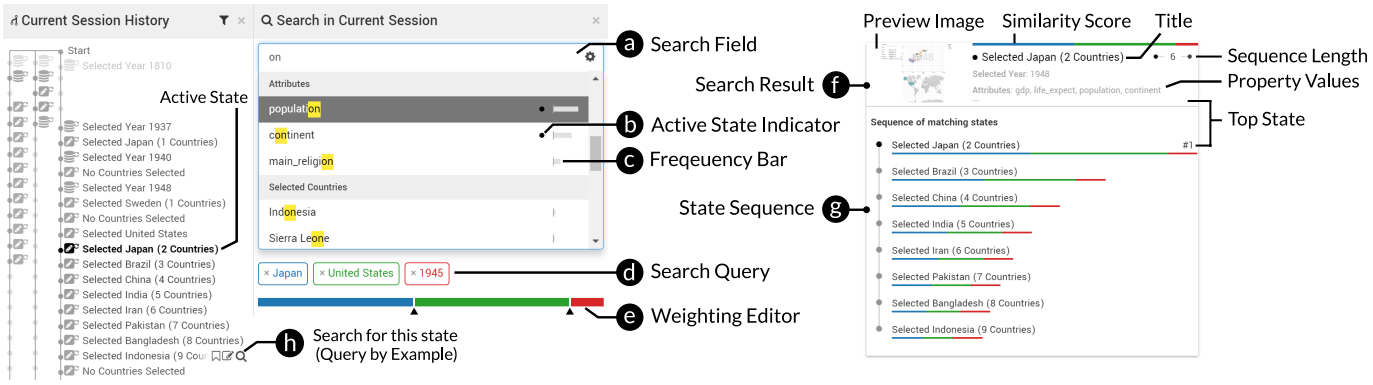

Fig. 4. Components of the retrieval interface. The search field (a) suggests visualization properties while typing. Properties of the active state are indicated by a black circle (b) and provide a frequency bar (c). The selected properties are added to the search query (d) and can be weighted based on the user's interest (e). The search results (f) are ranked by the state's similarity score and can be expanded to access the state sequence (g). The top state is used as search result cover and indicated in the state sequence. The active state is indicated in the sequence length glyph and in the state sequence. Users can also formulate a query with parts of an existing state (h).

Similar to the *Top 10* results that are derived from the whole provenance graph, we also use all properties from the current list of search results as a subset, and assemble a list of related property values that can be used to refine the search query and narrow down the scope.

Query by Example

Aside from entering explicit search terms, users can use their current analysis state in the application view to identify similar states, which is referred to as query by example. To simplify the identification of items that are present in the currently shown visualization state, for example, when analyzing the country development in *1945*, they are marked with a black circle in the suggestion list (see Figure 4b). However, these properties might be scattered throughout the whole list. As a solution, we provide a setting that filters the list for marked properties only.

The three search terms (*Japan*, *United States*, and *1945*) that have been added to the search query are displayed as color-coded words and as a stacked bar below the search field (see Figure 4d). The user can remove individual search terms from the query or clear the entire query. Modifying the search query triggers comparison and updates the search result list (see Figure 2).

Weighting Editor

The selected search terms, which can be seen as multiple attributes of a ranking of visualization states, can be weighted according to user interest by using a dedicated weight editing interface (see Figure 4e). By default, the weights are equally distributed across all search terms. In our example, the user wants to emphasize the two countries *Japan* and *United States* and reduce the impact of the year *1945*. To achieve this, the user can distribute the weight by dragging the sliders of the stacked bar, as in *LineUp* [14] and *ThermalPlot* [40]. Changing the weights triggers a recalculation of the similarity scores and updates the order of the search result. For the remainder of this example, we set the following weights: 45% for *Japan*, 45% for *United States*, and 10% for *1945*.

5.2 Search Results

We present matching visualization states in decreasing order according to their similarity scores (see Figure 1). As explained in Section 4.3, individual states are grouped into state sequences to guarantee temporal coherence across states (**O3**). Searching for the terms *Japan*, *United States*, and *1945* in our example, 41 states with a significance score greater than zero, which will be grouped into 16 state sequences. To further increase the readability of the results, only the top state of each sequence is displayed as a representative. However, users can reveal the whole sequence on demand (see Figure 4g).

Each search result box follows the same structure as shown in Figure 4f. The preview image provides a quick overview of all search results and support users in interpreting the somewhat abstract visualization state definition. This may be particularly helpful when comparing different search results.

The stacked bar at the top of a search result encodes how much each search term (i.e., the weighted relevance score) contributes to the state's similarity score. In our guiding example, the first search result matches all three search terms. To find out why some percents are missing, users can hover over the stacked bar to see more details about the distribution of the similarity score in a tool tip.

We show the property values grouped by property name as comma-separated list. In our example, the search terms *Japan* and *United States* are not visible immediately since the list exceeds the given space of the search result box. By hovering over the list, users can expand the box to see all property values and check that *Japan* and *United States* are exact matches, since they are highlighted. The selected year *1948* is not highlighted, because it does not match the search term *1945* by three years and thus explains the few missing percentages in the similarity score of the first result.

In addition to gaining an overview of the contained property values, users can explore the state sequence in detail. In the upper right corner of a search result, a glyph indicates the length of the state sequence. In the case of the search result that best matches our given search terms, the sequence represents eight states. Depending on the sequence length, the visual representation features one to three connected circles. For sequences with more than three states, we show the first and last states and indicate the remaining number of states in the center (see Figure 4f). If a state sequence contains the active visualization state, the circle or number is highlighted in black.

Clicking the glyph reveals the whole state sequence. All matching states are aligned from top to bottom and contain the state title and the weighted relevance score of the matching search terms as a stacked bar. We add an additional *#1* indicator to the top state of the sequence to point out that this state has the highest similarity with respect to the search query in the sequence and is already presented in greater detail above the sequence list as the representative state of this state sequence.

In our guiding example, the first sequence item was captured when *Japan* was selected as second country. The *United States* must have been selected as first country, since the title of subsequent states indicates that several other countries were selected. Moreover, the sequence shows that the similarity score for *Japan* and *United States* decreases for further selected countries in subsequent states (see Section 4.2). Hence, the first state of the sequence is indicated as the top state.

When the user hovers over a sequence item, the corresponding state is highlighted in the provenance graph view. Selecting an item loads the state in the visualization view. Likewise, when the user hovers over a search result box, the sequence of states is highlighted in the provenance view. Selecting a search result box loads the top state of the sequence. For the users' convenience, *KnowledgePearls*

remembers the last active visualization state before loading a selected search result in the visualization view. This allows users to easily restore the visualization state in which they started their search.

5.3 Provenance Graph

Next to the search side panel the user can optionally open the provenance graph as a second side panel (see Figure 1b) containing a visualization of all recorded states as described in [13]. Both side panels are linked, that is, the active visualization state and mouse hover are highlighted in both panels.

In order to provide an alternative to querying a state by a given example, we place a search button next to the state in the provenance view (see Figure 4h). Clicking the search button adds a dynamic property group that contains all values of the selected state to the top of the suggestion list of the search input field. Users can select properties of this state or filter the suggestions as previously explained (see Section 5.1).

We further support the user's search by enhancing the visibility of matching states in the graph visualization. For matching states, we increase the degree of interest, which causes expansion of the corresponding representations and collapses non-matching ones (see Figure 1b). We encode the similarity score in the opacity of a state representation, which makes it easier for users to focus on relevant states.

In our example, all matching states are enhanced when hovering over the first sequence, which matched all three search terms. The highlighted states confirm that the first country selected was *United States* and that *Japan* was selected as the second country.

6 Implementation

To demonstrate how existing visualization applications can be extended with *KnowledgePearls* retrieval capabilities, we implemented two prototypes: the first one is a *Gapminder*-inspired visualization implemented in *Vega* (see Figure 1) and the second one is our *Ordino* drug target discovery tool [44] that our collaborator used for the case study (see Section 7 and Figure 6).

Both prototype systems consist of three main building blocks (see Figure 5): (a) the application view containing the actual visualization(s), (b) the provenance tracking and provenance visualization component, and (c) the retrieval component.

The provenance and retrieval components are implemented in TypeScript using the *Phovea* platform [5]. The code is open source and available on Github [6]. The *Vega Gapminder* prototype is deployed at `https://vega-gapminder.caleydoapp.org` and the *Ordino* application is available at `https://ordino-retrieval.caleydoapp.org`.

The provenance tracking and visualization component uses the *CLUE* provenance graph implementation by Gratzl et. al. [13]. In the course of this work, we extended the *CLUE* approach by capturing and storing visualization states in the provenance graph, since the original approach captures only the action leading to a state but not the state itself.

We build the initial search index when loading the provenance graph into the browser and update it incrementally with every new user interaction. The retrieval for the currently loaded provenance graph is performed on the client side.

6.1 Integration Guidelines

While our two prototypes demonstrate how *KnowledgePearls* can be connected to existing visualization systems, we also want to provide concrete guidelines how this can be achieved for other visualizations.

A system needs to fulfill two important requirements to be compatible with *KnowledgePearls* (see Figure 5). First, it needs to be able to store and restore a visualization state. Second, the system needs to provide a description that instructs *KnowledgePearls* which retrieval-relevant properties need to be extracted from the visualization state and stored in the provenance graph. This includes, among other aspects,

[5] `http://phovea.caleydo.org`

[6] `https://github.com/Caleydo/knowledge-pearls`

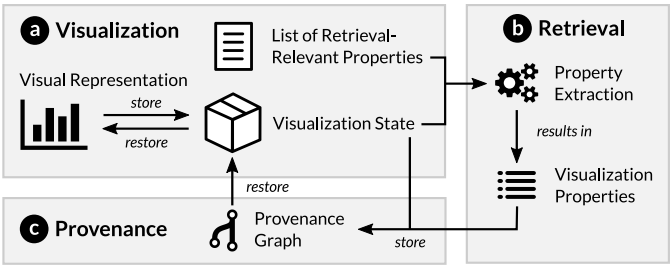

Fig. 5. Integration overview. The visualization component (a) can persist the representation as visualization state and provides a list of retrieval-relevant properties. The retrieval component (b) extracts the visualization properties from the state. The visualization properties and visualization state are then stored as new node in the provenance graph (c). When a user selects a node, the visualization state is pushed backed into the visualization and the representation is restored.

the computation of visualization quality metrics. Transferring this step to *KnowledgePearls* is in general not possible because it depends on the type and purpose of the visualization, and requires access to the data at a level that is opaque to *KnowledgePearls*. An interesting idea for future work is to incorporate approaches for computing quality metrics solely on the image results generated by the visualization system. However, this is a non-trivial topic and beyond the scope of this paper.

6.2 Vega Integration

The visualization in our *Gapminder*-inspired prototype is declared as *Vega JSON specification* and rendered using the *Vega* library [7].

As *Vega* comes with the built-in functionality to store and restore visualization states, it fulfills the first requirement. To meet the second requirement, we minimally extend the *Vega* specification with the two properties `track` and `search` that mark the declaration parts that are relevant for the retrieval (see Supplementary Listing 1). As these extensions operate on the interaction level, no extensions on the dataset, or individual data item level are necessary.

Further, we need to add event listeners to the *Gapminder* visualization to receive notifications when users change an attribute, select items, or choose a different year. In *Vega*, events are declared in the form of signals, which can release further actions, as for instance, update dependent signals or the visual representation itself. We employ this mechanism to check if the `track` property is contained in the declaration of the triggered signal. If this is the case, we start the property extraction in the retrieval component (see Figure 5b). To be able to distinguish between different signal sources, the developer needs to declare a title and icon for nodes, which will be displayed in the provenance graph view (see Figure 1).

The `search` property configures the creation of the visualization properties that will be offered to the user in the search side panel (see Figure 5b). The declaration must contain the property data type (categorical, numerical, or set), a custom title that is used as label for the search terms and suggestions, and a group label for the suggestions (see Section 5.1).

7 Case Study

We demonstrate the value and utility of the *KnowledgePearls* approach by integrating it into *Ordino* [44], a web-based discovery tool that allows users to flexibly rank and explore genes, cell lines, and tissue samples (see Figure 6).

In *Ordino*, the user starts by selecting or defining a list of items (e.g., genes) that will be opened in a multi-attribute ranking view based on *LineUp* [14]. Once the user has selected one or multiple items in the ranked list, a collection of possible follow-up detail views for exploring the current selection is displayed. This detail view can be either another ranking, a view with additional information about the selection, or a

[7] `https://vega.github.io/vega/`

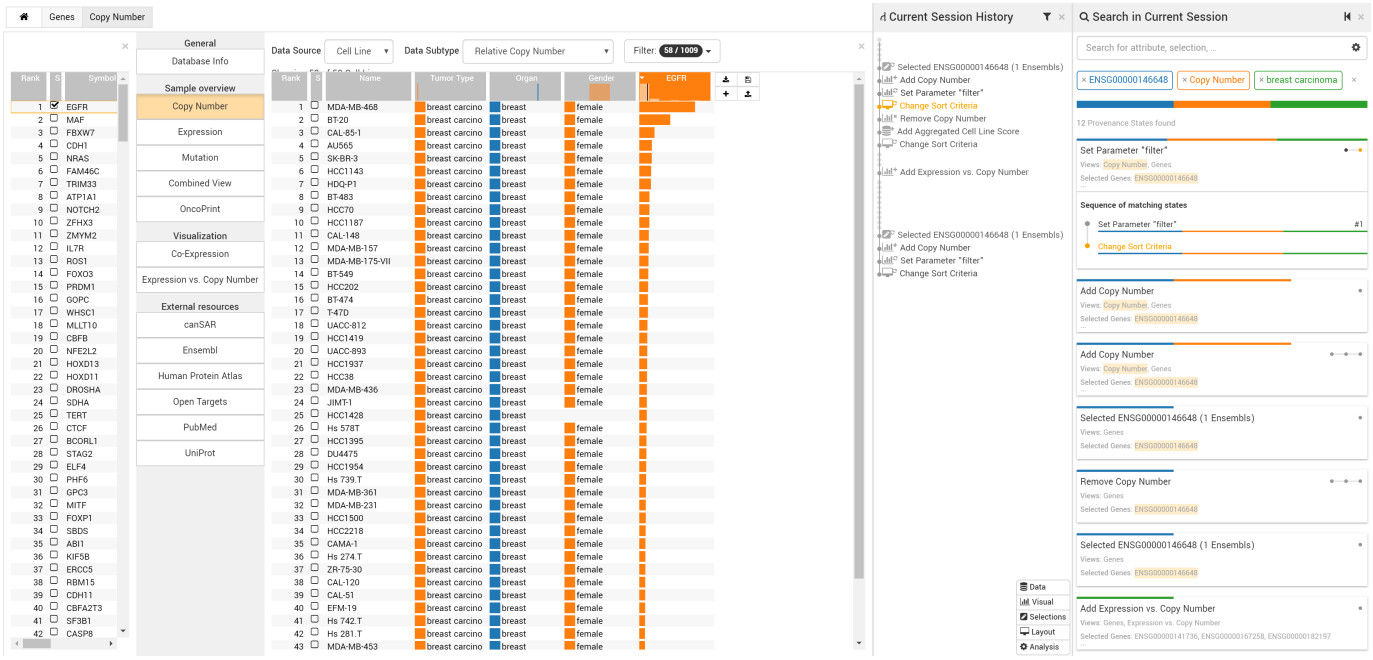

Fig. 6. The user has entered three search terms to find similar state sequences and to recall a previously recorded analysis. The search query results in seven state sequences, with the first sequence matching all search terms. Jumping to this state shows a ranking of breast cancer cell lines with copy number information for the gene *EGFR*. The user continues the analysis for *MDA-MB-468*, the cell line with the highest copy number.

visualization based on the selected items (e.g., a scatterplot matrix). *Ordino* follows a focus+context approach where the focus view is shown on the right and the previous focus view is shown as context on the left. New views are pushed from the right to the list of open views. Users are able to close the view and horizontally scroll back to previous views at any time. We extended *Ordino* with *KnowledgePearls* capabilities by tracking the shown views, parameter settings, and item selections as visualization states and store them in the built-in provenance graph. The set of properties that define a visualization state has been conjointly determined with our collaborators, based on what they consider as relevant for recalling previous analysis states and results.

The case study summarizes analysis sessions carried out by a collaborator working in the field of cancer research. Some time ago, the scientist performed several analyses regarding various cancer cell lines—cultured cells that are derived from tumors and that can proliferate indefinitely in the laboratory—and cancer genes (see Figures S1-S9 in supplementary material [8]). Now, the user comes back to that analysis session in order to find certain analysis steps and to continue with them. First, the user wants to know which previous analysis states are similar to the very last one from the session.

Ordino opens the last state at which the user left off as active state and shows a list of lung cancer cell lines ranked by the copy number of the gene *EGFR*. Using *EGFR* as search term results in two state sequences (see Figure S10). When hovering with the mouse over the search results, the corresponding state chains are highlighted in the provenance graph view.

The user refines the search by using *Copy Number* from the related suggestions in the drop-down menu as second search term, because he remembers having inspected this attribute in detail in the previous analysis. This time, the search returns five sequences, where the first two sequences match both search terms (see Figure S11).

The user knows that the initial analysis was not about lung cancer, as currently visible in the focus view, but he cannot remember the other tumor type. Entering "tum" for tumor type shows exactly two suggestions: *non-small-cell lung cancer* and *breast carcinoma*. Selecting *breast carcinoma* as the third search term results in seven state sequences. Exactly one sequence matches all search terms (see Figure S12). Jumping to the last state of this sequence shows a ranking for breast cancer cell lines with an additional column for *EGFR* copy number (see Figure 6).

The user has a new idea and wants to know whether the cell line *NCI-H2170* has already been used at some earlier point in the analysis. Searching for this cell line returns exactly one state sequence (see Figure S13). Jumping to the top result of the sequence reveals that the cell line was selected in a scatterplot in the *Expression vs. Copy Number* detail view. The user can seamlessly continue the analysis (see Figure 2) by opening a detail view, which provides further information about this cell line (see Figure S14). In conclusion, the *KnowledgePearls* integration supported the user in recalling previous work and making use of this knowledge in the context of a new analysis.

Informal User Feedback

We evaluated the *KnowledgePearls* integration into the *Ordino* platform with our collaborators by means of two thinking aloud sessions, in which we observed them while using the system.

Our collaborator pointed out multiple times that simplicity of the interface is more important than being able to express more complicated queries. We used this feedback as a guiding principle throughout all phases of the design process.

Furthermore, the collaborator valued the seamless switch between analysis and retrieval mode, and the flexibility in combining and weighting search terms to find specific analysis steps within a large provenance graph. He stated that the current solution with the search field and search term suggestions is highly useful and makes it easier to find contextually relevant previous states.

Our collaborator reported that he was slightly confused by the output of the grouping algorithm (see Section 4.3), which splits long sequences into smaller sequences for multiple search terms. We plan to address this feedback in the next iteration of our prototype.

The *Ordino* drug discovery system extended with *KnowledgePearls* retrieval capabilities will soon be in productive use by a few dozen domain experts form multiple disciplines—including biology, cancer genomics, and bioinformatics. The active use of the tool will result in a quickly growing provenance graph. In the case study described above, our collaborator was operating on the provenance graph that he created

[8] `http://knowledge-pearls.caleydo.org`

as a single user in multiple analysis sessions over time. However, he stressed that *KnowledgePearls* will be particularly valuable in the future when the provenance graph contains visualization states from exploration sessions done by his colleagues working on various projects. To use the full potential of *KnowledgePearls* for collaborative scenarios, we plan to extend *KnowledgePearls* with additional annotation and filtering capabilities.

8 Discussion and Limitations

8.1 Generalizability

Many applications do not provide a visualization state and hence, do not fulfill the integration requirements (see Section 6.1). In this case an image of the visualization can be captured subsequent to every interaction. The images can be processed using computer vision and machine learning algorithms to extract the properties, such as axis labels or data items [17, 26, 27, 34]. The extracted properties can form a visualization state and/or list of retrieval-relevant properties that are served as input for our retrieval approach (see Figure 5b).

In case the application cannot restore a visualization state, users are unable to switch back from retrieval mode into visual analysis mode (see Figure 2) and must manually recover the state based on the given property information. Recovering visualizations automatically for a given visualization state (e.g., using reinforcement learning) remains an open research topic.

8.2 Relationship of Visualization Properties

Besides the relationships between properties of a visualization, multiple coordinated view (MCV) setups add relationships across visualizations. In our *Gapminder*-inspired prototype a world map is linked to a scatterplot, i.e., country selections are updated in both visualizations accordingly. For the retrieval we consider property values from both visualizations. As discussed above, by treating the search terms independently users cannot specify in a query that a property needs to be present in a particular view that is part of the MCV setup.

In early stages of our concept, we explored a tree-like structure for organizing and structuring visualization properties such that the root node represents the MCV setup itself and its children represent the individual visualizations. However, increasing the expressiveness of the visualization property setup increases the complexity of the query formulation. Besides MCV setups in which different visualization techniques show the same data, setups that use the same visualization technique for different data are challenging. Typical examples are scatterplot matrices and parallel coordinate plots. In both, the number of instances (scatterplot or axis) is variable, as it depends on the number of attributes in the explored dataset. Managing such a variable set of visualizations and querying specific subsets remain open topics for future research.

Further, the current concept does not consider logical operators other than the *AND* combination, which we use by default to combine the individual query terms. A *NOT* operator, for example, could be valuable for expressing that a certain state should not contain a given search term. Integrating advanced logical operators could lead to new kinds of retrieval goals in which not the presence but the absence of certain visualization properties is desired. For instance, users could penalize the presence of a well-known and frequently used gene to filter out commonly explored cases.

8.3 Graph Retrieval

We designed *KnowledgePearls* for retrieving similar states from a recorded provenance graph based on a user-defined search query. However, due to the graph structure and its inherently contained metadata, such as date of creation of individual states (**O3**), more advanced queries and retrieval techniques could be applied to the graph itself. Motif-based search to query specific action sequences could be a valuable addition. Applications include cases in which users remember states that led to a certain insight rather than the state that contains the insight itself.

In addition, motif-based search and graph retrieval could be used to extract knowledge about the analysis process itself. Identifying and retrieving repetitive patterns in the provenance graph can be useful for various purposes, such as detecting common analysis patterns across users. These identified patterns could then be used to provide guidance systems that support the user throughout the analysis by suggesting common action sequences based on the current one. However, how to guide users without restricting them in the exploration process remains an open research question.

8.4 Scalability

Real world analysis sessions can quickly result in a provenance graph with many nodes and branches. A branch is introduced when a user jumps back to a previous state and continues the analysis from that point. In *KnowledgePearls* we improve the scalability by identifying similar visualization states and grouping them into sequences (**O3**). However, if the same or a highly similar state sequence is found in multiple branches, each sequence is displayed as separate search result. As part of future work, we plan to apply semantics-based, motif-based, and hierarchical aggregation strategies [2, 9, 21, 41].

Besides the visual scalability of the provenance graph visualization and retrieval interface, also the computational and storage scalability play a major role for an effective provenance retrieval solution. Both are influenced by two factors: (1) the number of states in the graph and (2) the number of attributes stored in each visualization state (**O1**). In the current prototype the whole graph containing all attributes for each state are stored in a database on the server and transferred to the client. This approach works for small provenance graphs with up to a few hundred states, which are typically generated by a single user in a consecutive exploration session. In such scenarios, computing the similarities between the search query and individual states can be executed on the client side, due to the negligible computational overhead. However, when additionally integrating provenance graphs from exploration sessions created by other users, different measures need to be taken, such as computing the similarity score on the server and transferring only the relevant states to the client. However, a trade-off exists between computing the scores on the server and the flexibility to let the user weight individual components, as the latter would require a re-computation of each score upon change. To address this issue, we plan to investigate hashing strategies.

9 Conclusion and Future Work

We have presented *KnowledgePearls*, an approach to searching effectively for visualization states in provenance graphs. An intuitive visual interface enables users to query and explore previous analysis states based on a definition that can be explicitly formulated or implicitly inferred from a given reference state. As a key aspect of our work, the visualization and the used metrics support a quantitative notion of similarity. This allows for a gradual ordering of states by their relevance and enables users to express interest by assigning weights to different elements of the search definition. A case study carried out by collaborators in the field of cancer drug discovery illustrated how *KnowledgePearls* facilitates the exploration process by recalling states from earlier analyses.

As part of future work, we intend to formally evaluate our retrieval approach in a user study with domain experts from multiple domains. Another direction for future work is to extend our approach with comprehensive meta-analyses capabilities, which might allow visualization designers to better understand the non-linearity and backtracking nature of visual analysis.

Acknowledgments

We are grateful to Suzie Lee Hoops for proof-reading our work. This work was supported in part by Boehringer Ingelheim Regional Center Vienna, the Austrian Science Fund (FWF P27975-NBL), and the State of Upper Austria (FFG 851460). The VRVis Forschungs-GmbH is funded by COMET – Competence Centers for Excellent Technologies (854174) by BMVIT, BMWFW, Styria, Styrian Business Promotion Agency – SFG and Vienna Business Agency. The COMET Programme is managed by FFG.

REFERENCES

[1] C. Ahlberg. Spotfire: An Information Exploration Environment. *ACM SIGMOD Record*, 25(4):25–29, 1996. doi: 10.1145/245882.245893

[2] T. Aittokallio and B. Schwikowski. Graph-based Methods for Analysing Networks in Cell Biology. *Briefings in Bioinformatics*, 7(3):243–255, 2006. doi: 10.1093/bib/bbl022

[3] L. Bavoil, S. P. Callahan, C. Scheidegger, H. T. Vo, P. Crossno, C. T. Silva, and J. Freire. VisTrails: Enabling Interactive Multiple-View Visualizations. In *Proceedings of the IEEE Conference on Visualization (VIS '05)*, pp. 135–142. IEEE, 2005. doi: 10.1109/VISUAL.2005.1532788

[4] E. Bertini, A. Tatu, and D. Keim. Quality Metrics in High-Dimensional Data Visualization: An Overview and Systematization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2203–2212, 2011. doi: 10.1109/TVCG.2011.229

[5] O. Biton, S. Cohen-Boulakia, S. B. Davidson, and C. S. Hara. Querying and Managing Provenance Through User Views in Scientific Workflows. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE '08)*, pp. 1072–1081. IEEE, 2008. doi: 10.1109/ICDE.2008.4497516

[6] M. A. Borkin, C. S. Yeh, M. Boyd, P. Macko, K. Z. Gajos, M. Seltzer, and H. Pfister. Evaluation of Filesystem Provenance Visualization Tools. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2476–2485, 2013. doi: 10.1109/TVCG.2013.155

[7] E. Clarkson, K. Desai, and J. Foley. ResultMaps: Visualization for Search Interfaces. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1057–1064, 2009. doi: 10.1109/TVCG.2009.176

[8] A. Dasgupta and R. Kosara. Pargnostics: Screen-Space Metrics for Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10)*, 16(6):1017–1026, 2010. doi: 10.1109/TVCG.2010.184

[9] N. Elmqvist and J.-D. Fekete. Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2010. doi: 10.1109/TVCG.2009.84

[10] J. Freire, C. T. Silva, D. Koop, and E. Santos. Provenance for Computational Tasks: A Survey. *Computing in Science & Engineering*, 10:11–21, 2008. doi: 10.1109/MCSE.2008.79

[11] J. Frew, D. Metzger, and P. Slaughter. Automatic Capture and Reconstruction of Computational Provenance. *Concurrency and Computation: Practice and Experience*, 20(5):485–496, 2008. doi: 10.1002/cpe.1247

[12] D. Gotz and M. X. Zhou. Characterizing Users' Visual Analytic Activity for Insight Provenance. *Information Visualization*, 8(1):42–55, 2009. doi: 10.1057/ivs.2008.31

[13] S. Gratzl, A. Lex, N. Gehlenborg, N. Cosgrove, and M. Streit. From Visual Exploration to Storytelling and Back Again. *Computer Graphics Forum*, 35(3):491–500, 2016. doi: 10.1111/cgf.12925

[14] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. LineUp: Visual Analysis of Multi-Attribute Rankings. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '13)*, 19(12):2277–2286, 2013. doi: 10.1109/TVCG.2013.173

[15] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '08)*, 14(6):1189–1196, 2008. doi: 10.1109/TVCG.2008.137

[16] M. Herschel, R. Diestelkmper, and H. Ben Lahmar. A survey on provenance: What for? What form? What from? *The VLDB Journal*, 26(6):881–906, Dec. 2017. doi: 10.1007/s00778-017-0486-1

[17] D. Jung, W. Kim, H. Song, J.-i. Hwang, B. Lee, B. Kim, and J. Seo. ChartSense: Interactive Data Extraction from Chart Images. pp. 6706–6717. ACM Press, 2017. doi: 10.1145/3025453.3025957

[18] S. Khan, U. Kanturska, T. Waters, J. Eaton, R. Baares-Alcntara, and M. Chen. Ontology-assisted provenance visualization for supporting enterprise search of engineering and business files. *Advanced Engineering Informatics*, 30(2):244–257, 2016. doi: 10.1016/j.aei.2016.04.003

[19] M. Kreuseler, T. Nocke, and H. Schumann. A History Mechanism for Visual Data Mining. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '04)*, pp. 49–56. IEEE, 2004. doi: 10.1109/INFVIS.2004.2

[20] E. Maguire, P. Rocca-Serra, S.-A. Sansone, J. Davies, and M. Chen. Taxonomy-Based Glyph Design with a Case Study on Visualizing Workflows of Biological Experiments. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '12)*, 18(12):2603–2612, 2012. doi: 10.1109/TVCG.2012.271

[21] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, 298(5594):824–827, 2002. doi: 10.1126/science.298.5594.824

[22] G. Ozsoyoglu and H. Wang. Example-based graphical database query languages. *Computer*, 26(5):25–38, May 1993. doi: 10.1109/2.211893

[23] S. Pajer, M. Streit, T. Torsney-Weir, F. Spechtenhauser, T. Mller, and H. Piringer. WeightLifter: Visual Weight Space Exploration for Multi-Criteria Decision Making. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '16)*, 23(1):611–620, 2017. doi: 10.1109/TVCG.2016.2598589

[24] Pat Hanrahan, Chris Stolte, and Jock Mackinlay. Tableau: Visual Analysis for Everyone, 2007.

[25] R. Pienta, F. Hohman, A. Tamersoy, A. Endert, S. Navathe, H. Tong, and D. H. Chau. Visual Graph Query Construction and Refinement. pp. 1587–1590. ACM Press, 2017. doi: 10.1145/3035918.3056418

[26] J. Poco and J. Heer. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. *Computer Graphics Forum*, 36(3):353–363, June 2017. doi: 10.1111/cgf.13193

[27] J. Poco, A. Mayhua, and J. Heer. Extracting and Retargeting Color Mappings from Bitmap Images of Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):637–646, Jan. 2018. doi: 10.1109/TVCG.2017.2744320

[28] B. Prez, J. Rubio, and C. Senz-Adn. A systematic review of provenance systems. *Knowledge and Information Systems*, Feb. 2018. doi: 10.1007/s10115-018-1164-3

[29] E. Ragan, A. Endert, J. Sanyal, and J. Chen. Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purposes. *IEEE Transactions on Visualization and Computer Graphics (VAST '15)*, 22(1):31–40, 2016. doi: 10.1109/TVCG.2015.2467551

[30] G. Salton, A. Wong, and C.-S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975. doi: 10.1145/361219.361220

[31] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2017. doi: 10.1109/TVCG.2016.2599030

[32] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):659–668, Jan. 2016. doi: 10.1109/TVCG.2015.2467091

[33] A. Satyanarayan, K. Wongsuphasawat, and J. Heer. Declarative Interaction Design for Data Visualization. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '14)*, pp. 669–678. ACM Press, 2014. doi: 10.1145/2642918.2647360

[34] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer. ReVision: automated classification, analysis and redesign of chart images. p. 393. ACM Press, 2011. doi: 10.1145/2047196.2047247

[35] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva. Querying and re-using workflows with VisTrails. In *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD '08)*, pp. 1251–1254. ACM, 2008. doi: 10.1145/1376616.1376747

[36] B. Shneiderman, C. Williamson, and C. Ahlberg. Dynamic Queries: Database Searching by Direct Manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*, pp. 669–670. ACM, 1992. doi: 10.1145/142750.143082

[37] Y. B. Shrinivasan and J. J. van Wijk. Supporting the analytical reasoning process in information visualization. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1237–1246. ACM, 2008. doi: 10.1145/1357054.1357247

[38] G. Smith, M. Czerwinski, B. R. Meyers, G. Robertson, and D. S. Tan. FacetMap: A Scalable Search and Browse Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5), 2006. doi: 10.1109/TVCG.2006.142

[39] A. Srinivasan and J. Stasko. Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '17)*, PP(99):1–1, 2017. doi: 10.1109/TVCG.2017.2745219

[40] H. Stitz, S. Gratzl, W. Aigner, and M. Streit. ThermalPlot: Visualizing Multi-Attribute Time-Series Data Using a Thermal Metaphor. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2594–2607, 2016. doi: 10.1109/TVCG.2015.2513389

[41] H. Stitz, S. Luger, M. Streit, and N. Gehlenborg. AVOCADO: Visualiza-

tion of WorkflowDerived Data Provenance for Reproducible Biomedical Research. *Computer Graphics Forum*, 35(3):481–490, 2016. doi: 10.1111/cgf.12924

[42] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002. doi: 10.1109/2945.981851

[43] C. Stolte, D. Tang, and P. Hanrahan. Query, Analysis, and Visualization of Hierarchically Structured Data Using Polaris. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '02)*, pp. 112–122. ACM, 2002. doi: 10.1145/775047.775064

[44] M. Streit, S. Gratzl, H. Stitz, A. Wernitznig, T. Zichner, and C. Haslinger. Ordino: visual analysis tool for ranking and exploring genes, cell lines, and tissue samples. pp. –, 2018. doi: 10.1101/277848

[45] L. Wilkinson. *The Grammar of Graphics*. Springer, 2nd ed., 2005.

[46] L. Wilkinson, A. Anand, and R. Grossman. Graph-Theoretic Scagnostics. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '05)*, pp. 157–164, 2005. doi: 10.1109/INFVIS.2005.1532142

[47] M. M. Zloof. Query-by-Example: A data base language. *IBM Systems Journal*, 16(4):324–343, 1977. doi: 10.1147/sj.164.0324