Compression in Visualization

Eamonn J. Maguire
St. Anne's College

Michaelmas Term 2014

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Department of Computer Science
University of Oxford

Contents

Chapter 1

Introduction

Chapter 2

Related Work

Chapter 3

Taxonomy-Based Glyph Design—with a Case Study on Visualizing Workflows of Biological Experiments

3.1 Introduction

Glyph-based visualization is a class of visual representations where a collection of small visual objects (referred to as *glyphs*) are used to encode different attribute dimensions of an input data space. A good glyph design can enable users to conduct visual search more efficiently during interactive visualization, and facilitate effective learning, memorizing and using the visual encoding scheme. A less effective visual design may suffer from various shortcomings such as being perceptually confusing, semantically ambiguous, difficult to learn and remember, or unable to accommodate low-resolution display devices. Most accepted designs have undergone an enduring process of evolution, refinement and standardization. One could not and should not remove the necessity for such processes in glyph design. Meanwhile, the design process for a glyph set usually relies on an assortment of creativity, artistic skill, domain knowledge, intuition about users, and sometimes personal preference. While most of these qualities are absolutely helpful and some are inevitable, it is highly desirable to introduce "systematicism" and objectivity into the design process.

To demonstrate our approach, we elected the field of experimental biology as a test bed for developing a systematic methodology to create a glyph-based visual mapping for visualizing experimental design and experimental processes. While experimental design is at the heart of biological data evidence gathering, representation of such information is confined to either verbose description or ineffective representations. This claim is backed by a survey of public biodata repositories, devised to ensure data perennity, scientific scrutiny and reproducibility. As illustrated in Fig. **??**(a), workflows are traditionally drawn as text-based diagrams. Text labels are indices of concepts, and usually they do not encode multivariate information directly. Text-labeled boxes are costly in terms of display space usage as well as time required for visual search since parsing and interpretation of text is a slow post-attentive process [**?**]. They are particularly ineffective when one needs to compare between different workflows or to identify unusual or missing components in a workflow.

Therefore, exploring an alternative in the form of glyph based representations, which are at the core of many successful schematic diagrams in the history of sciences, engineering and business management, offers a potentially more effective means for depicting these workflows. This presents us with an interesting case study where other types of design approaches would not be appropriate, especially in dealing with several hundreds of conceptual names.

It is important to note that domain experts are in general more willing and able to learn and memorize an encoding scheme in order to improve the accuracy and efficiency of routine tasks. At the same time, it is also necessary for an encoding scheme to facilitate effective learning and remembering through appropriate abstraction and metaphors.

Motivated by this case study, we made an observation that when a large number of concepts (or taxa) are organized into a taxonomic tree, the hierarchy typically represents an ordering of different categorization schemes. The schemes that are higher-up in the taxonomy (closer to the root) are usually considered to be more important, which reflect their conceptual coverage, frequency of usage, domain-specific convention, and some other measurable factors. In terms of visual encoding, higher up schemes should ideally be mapped to visual channels that have more discriminative capacity. Hence, we can explore the parallel between the taxonomic hierarchy and the ordering of visual channels based on discriminative capacity to formulate a systematic and relatively objective process for designing a glyph set. This approach addresses one of the most common criticisms of data glyphs: the data to visual attribute mapping bias [**?**]. Given a large data repository that encompasses many concepts, our design process is composed of four major steps: (i) gathering and processing raw metadata for obtaining a set of taxa (names); (ii) formulating a taxonomy based on a set of categorization schemes (Section **??**); (iii) carrying out visual design, which includes the sub-process of determining the ordering of visual channels, proposing optional visual mappings, and identifying metaphoric abstractions and associations (Section **??**); and (iv) implementing a glyph-based visualization system, in our case, for depicting workflows of biological experiments (Section **??**).

Similar to most design processes, it is helpful to conduct all stages in a progressive and iterative manner. As glyph-based visualization is normally deployed in a specific application domain, it is important to involve domain experts at every stage of the design process. During this work, we met with domain specialists on a weekly basis.

3.2 Related Work

In this section, we give a brief overview of two most relevant areas in visualization, *glyph-based visualization* and *workflow visualization*. The remainder background information includes the biological data management, perceptual guidance, and categorization algorithms, which will be described in the following sections where the relevant technical details are discussed.

3.2.1 Glyph-based Visualization

There are many examples of glyph usage in the literature spanning many disciplines, especially in conjunction with many schematic diagrams. Bertin considered a range of simple glyphs in the context of geo-information visualization [**?**]. Ward surveyed the use of glyphs in visualization, and discussed a number of bias issues, design approaches, and layout options [**?**]. Ropinksi *et al.* conducted a survey on glyph-based techniques in medical visualizations [**?**]. In visualization, a number of interesting glyph designs have been presented with noticeable impact on a large range of applications (e.g., medicine [**?**], software [**?**], text [**?**], and scientific computation [**?**]). Furthermore, Ribarsky *et al.* developed an editing system, Glyphmaker [**?**], to assist the process of designing glyphs. Post *et al.* proposed a language, Icon Modeling Language, for creating glyphs and glyph-style contours [**?**].

There have been some recent efforts to use glyph-based visualization in biological sciences. One noticeable attempt is the *Systems Biology Graphical Notation* (SBGN) [**?**]. The design of SBGN makes use of the notional representations of UML with some modifications to describe the biological entities and their interactions within a biological system. GenoCAD [**?**] provides a grammar-based language for representing and searching DNA sequences and for building genetic constructs from DNA sequences, facilitating the use of conventional link diagrams. Both systems rely heavily on text labels. In handling a large collection of workflows, they exhibit a few shortcomings, such as inefficiency in using display space and ineffectiveness in supporting some visualization tasks, such as comparisons and novelty or anomalies detection in workflows.

In the literature, several authors encouraged glyph designers to consider visual perception when constructing glyphs [**?**, **?**]. Others examined the design space of icons, which normally encode less information than glyphs (e.g., [**?**, **?**]). There were perceptual studies showing the merits of icons over text labels (e.g., [**?**, **?**]), as well as those showing the contrary (e.g., [?]). Building on such discussions, this work aims to address a methodological need for a systematic approach to glyph design for applications where large collections of concepts need to be visually encoded using glyphs.

3.2.2 Workflow Visualization

The need for *workflow visualization* is pervasive across many different domains. For example, in business and management the Gantt chart is a form of text based workflow visualization, while BPMN (Business Process Model and Notation) and EPC (Event-driven Process Chain) make use of icons to enrich text labels. In engineering disciplines, various schematic diagrams, such as UML (Unified Modeling Language), Petri-net and circuit diagrams, are used to convey data flow and process interaction.

In this work, we consider workflows used to describe biological experiments. This class of workflow visualization renders the processes enacted on biological materials in an experiment (e.g., removal of blood sample from patient). There has been little effort to develop tools that address the domain-specific needs. Scientists usually use generic text-based graph drawing tools such as GraphViz [?].

One related aspect is *pathway visualization*, which is concerned with the rendering of cellular biological processes. An array of pathway visualization tools have been developed, some of which incorporate glyphs. For example, VANTED [?] overlays glyphs representing gene expression, enzyme and metabolite profile data on top of pathway diagrams from KEGG [?]. GENeVis [?], which also employs glyphs to represent multivariate data, is a comprehensive visualization toolkit for exploration of pathways in conjunction with temporal data. GenoCAD has at its core a workflow generation system and relies on their glyph library for rendering of the different biological processes within a cell. SBGN-ED [?], is an add-on for the VANTED software and performs pathway creation using the aforementioned SBGN [?] visual language.

3.3 Motivation and Overview

The advent of massively parallel techniques such as DNA microarray, mass-spectrometry based proteomics and metabolomics and next generation sequencing enables molecular biologists to collect, process and manipulate biological signals on a new scale. Big data in biology is a reality. There has been serious effort in biology to ensure quality of experimental records by providing data archiving infrastructure and defining standards for adequate annotation and sufficient details for recapitulation of results. A number of molecular biology signature databases have been or are being established to cover the main molecular dimensions: transcript, protein and metabolites. The captured metadata mostly revolves around a similar configuration where sets of samples corresponding to different conditions are processed, measurements produced and analysis performed, delivering data that needs to be handled and interpreted. While the availability of such metadata facilitates comparison across datasets and enables meta-analysis, providing means to serve an overview of experimental design can assist analysts and data managers alike, from data selection according to relevancy, to error detection such as imbalances, irregularities or inconsistencies in records.

Task Analysis. In the context of meta-databases for experimental records, there are two main groups of users. The majority of users are those who create data for their biological experiments or retrieve data relevant to their scientific interests. Their tasks include: (a) entering and editing their own experimental records; (b) transforming workflow records to schematic diagrams for comparison, external memorization, publication and education; (c) uploading and downloading datasets; (d) querying and searching for relevant experiments; (e) understanding workflows of existing experimental designs; (f) identifying similarity and difference between workflows for a common task; (g) understand pooling events and sample relations (derivation).

The second group of users are curators who manage data archives. As biology or bioinformatics scientists themselves, they perform the above-mentioned tasks (d)-(g) frequently, and (b)-(c) occasionally. In addition, they also carry tasks for: (h) checking syntactic correctness of submitted workflows; (i) checking semantic correctness of submitted workflows which usually involves reading the associated publications then comparing the submitted workflows with those described in the publications; (j) interacting with authors of submitted

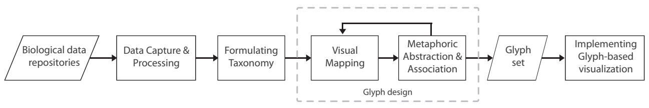

Biological data repositories
Data Capture & Processing
Formulating Taxonomy
Visual Mapping
Metaphoric Abstraction & Association
Glyph design
Glyph set
Implementing Glyph-based visualization

Figure 3.1: A systematic process for creating glyph based representations.

workflows to clarify any inconsistency and misunderstanding; (k) making appropriate corrections in submitted records where necessary; (l) augmenting with appropriate annotation based on additional information found in the associated publications; (m) providing authors with submission feedback. (n) forming an up-to-date overview about the experiments in the data archives, and maintaining an insight about the provenance of the archives; (n) analyzing the grouping, replication, distribution and trend of experiments; (o) analyzing ambiguities, errors and uncertainty in experimental recording, and identifying the need to enrich and refine the relevant standards for ontology, labelling and annotation. (p) providing tools to assist users in creating meta-data from raw data.

It is not difficult to observe that effective workflow visualization can significantly improve users' capability in performing tasks b, e, f, g, h, i, j, l, n, o and p.

While it is necessary to evolve a glyph-based design for workflow visualization over a period, it is also important not to make the first step in an ad hoc manner. Such a process does not scale well with the requirements of the application concerned where a large number of concepts are to be encoded using glyphs. We thus adopt a new systematic process for glyph design by exploring the parallel between the hierarchy of concept categorization and the ordering of discriminative capacity of visual channels. Fig. **??** depicts this process.

Data Capture and Processing. We first retrieved workflow metadata from a biological repository (content of the ArrayExpress archive), through use of a multi-threaded harvesting operation. All experimental workflows were then converted using a MAGE-Tab to ISA-Tab converter, the latter format being more general than the former and can be used to carry metadata payloads for many types of experiments, making the data processing program more "future-proof".

From the 21,000 ISA-Tab files, we extracted all names of protocols (processes) and biological materials, chemical materials, devices and data used in annotation. We also computed the occurrence of each material and process found in the entire set of ISA-Tab files, which have been used as one of the metrics in the next stage (see Section 5 for details). This step results in 61 process names and 3492 names of inputs and outputs (e.g., biological and chemical materials, device measurements and data).

Taxonomy Formulation. Since a taxonomy for such a large collection of terms is absent, we, for starters, established a set of categorization schemes with the help of domain experts. For example, one of the schemes for categorizing processes may be based on different biological inputs to a process (e.g., molecule, cell, organism, etc.). Another scheme may be based on processing methods (e.g., perturbation, combination, etc.). We computed the quality measures of each scheme based a set of generic metrics (see Section **??**) then created a taxonomic tree by recursively selecting the best categorization scheme based on the quality measures. We finalized the organization of the taxonomy by allowing domain experts (2 co-authors) to make adjustments according to domain-specific conventions. All leaf nodes of the resulting taxonomy tree are names extracted from the workflow metadata. All non-leaf nodes represent a categorization scheme.

Glyph Design. First, we established an ordering of commonly-used visual channels based on the literature focused on perception and visualization (see Section **??**). This allows us to systematically compare the order of categorization schemes in the taxonomy with the order of different visual channels. Ideally, schemes at the upper level of the tree can be mapped to visual channels which are more prominent to our visual system.

We then proceeded to the glyph design process involving two intertwined sub-processes for **visual mapping** and **metaphoric abstraction and association**. We proposed various options of visual channels for each categorization scheme featured in the taxonomic tree. We considered the merits of these visual channels and identify potential conflicts with the visual channels that have been assigned to other schemes. With the direct help from domain experts, we tried to identify a metaphoric abstraction or association for each design option proposed. We evaluated and recorded the intuitiveness and suitability of the abstraction and metaphor association.

Informed by the results the two sub-processes, we finalized a glyph set by selecting a design option for each scheme, while maintaining the order of discriminative capacity of visual channels, minimizing conflicts between different channels, and maximizing the use of metaphoric abstraction and association.

Implementation of Glyph-based Visualization. We then integrated the glyph set with a layout algorithm to form a prototype system for workflow visualization (see Section **??**). For demonstration purposes, we tested the workflow visualization in conjunction with ISAcreator, a popular domain agnostic biological experiment metadata capture tool.

3.4 Taxonomy Generation

Given a large collection of concepts, we should ideally make use of a standard taxonomy, where non-leaf nodes represent different categorization schemes and each scheme provides subclasses that lead to different sub-trees. In many circumstances, however, there is no agreed taxonomic tree as establishing a standard categorization requires non-trivial scientific effort that often spans over a few decades.

The algorithm described below is not intended to establish a semantic-rich taxonomic tree for classifying concepts. It is designed purely for addressing the needs for ordering various categorization schemes (when there is no such an agreed order) to aid glyph design, and for devising a useful tree data structure in implementing the mapping between concepts and glyphs in workflow visualization.

Hence, the criteria used for structuring the tree are based on usage of the concepts and the structural quality of the tree to be constructed.

Taxonomy is a long standing concept that can be traced back to 3000BC [**?**]. Automatic taxonomy generation has been an active field in computer science and computational biology with existing work largely focusing on clustering algorithms (e.g., [**?**]). Many such algorithms assume the availability of similarity measures for ordering entities rather than relying on the existence of individual categorization schemes. In these algorithms, there is usually no attempt towards application of a metric for creating meaningful non-leaf nodes in the resulting tree. In this work, it is essential to keep each categorization scheme as a non-leaf node unless it is redundant.

3.4.1 Ordering Classification Schemes

Let $\mathscr{X} = \{x_1, x_2, \ldots, x_n\}$ be a set of concepts to be classified. In our application, this is the set of all valid names of biological processes and IOs (inputs and outputs) in the database. Each concept x_i is associated with a scalar value, $\mu_i \in [0,1]$, indicating its frequency of usage in relation to the total occurrence of all concepts in the database. There are a number of categorization schemes, $\mathscr{S} = \{S_1, S_2, \ldots, S_m\}$. Each scheme, S_k divides concepts into several classes, $c^k_1, c^k_2, \ldots, c^k_{l_k}$. The relationship between the concept set $\mathscr{X}$ and different classification schemes can thus be represented by a Boolean matrix, $\mathbf{A}$, where each element $\alpha[i,j,k] = 1$ if concept x_i belongs to the j^{th} class of scheme S_k; otherwise $\alpha[i,j,k] = 0$. In the context of feature-based categorization, one can also view each scheme S_k as a feature, and each class under S_k as a particular type of this feature. Without losing generality, we assume that the classes under the same S_k are disjoint. It is also possible that a concept does not possess the k^{th} feature, and hence does not belong to any class under S_k.

Given the above categorical information about the concept set $\mathscr{X}$, one can choose a categorization scheme $S_k \in \mathscr{S}$, which will divide $\mathscr{X}$ into a number of disjoint subsets corresponding to classes $c^k_1, c^k_2, \ldots, c^k_{l_k}$ and c^k_0. The subset c^k_0 contains those concepts which S_k is unable to classify. The partitioning process can be repeated recursively by applying one of the remaining categorization schemes in $\mathscr{S}$ to each subset. This results in a hierarchical categorization tree, which defines a taxonomy for the concepts set $\mathscr{X}$.

The ordering of the schemes in $\mathscr{S}$ thus determines the taxonomic structure of $\mathscr{X}$. It is not difficult to anticipate that many criteria can be used to determine the ordering for a given concept set. Some criteria will no doubt encode application-specific semantics, and some may be subjective or debatable. However, there are also some common-sense criteria that are generic to most applications. These include:

Coverage. The number of concepts that can be classified by scheme S_k is a capacity measure of S_k. The more concepts that S_k can classify (i.e., the fewer in c_0^k), the better. The measure, which is normalized by the set size $|\mathscr{X}| = n$, can be defined as:

$$M_1(S_k) = \frac{\sum_{i=1}^{n} \max_{1\le j\le l_k}(\alpha_{i,j,k})}{n}. \tag{3.1}$$

Potential Usage. A categorization scheme that is higher up in the taxonomical tree is expected to be used more often in the application concerned. The occurrence frequencies of concepts, μ_i, enable us to estimate the potential usage of a classification scheme as follows:

$$M_2(S_k) = \frac{\sum_{i=1}^{n} \mu_i \max_{1\le j\le l_k}(\alpha[i,j,k])}{\sum_{i=1}^{n} \mu_i}. \tag{3.2}$$

Subtree Balance. Having a balanced node distribution in a tree is a desirable property of a tree structure. It prevents a tree from having an excessive height, which corresponds to the need for more visual channels. Let l_k denote the number of subclasses in categorization scheme S_k, τ_j be the number of concepts in each subclass $c_j^k, j = 1,2,\ldots,l_k$ and σ_τ and $\bar{\tau}$ be the standard deviation and mean respectively of $\tau_1,\ldots,\tau_{l_k}$. We can measure the level of balance as follows:

$$M_3(S_k) = \begin{cases} 0 & \sum_{i=1}^{l_k} \tau_{l_i} = 0 \\ 1 & \sigma_\tau < \varepsilon(\varepsilon > 0) \\ P(\bar{\tau} \pm 1 \mid N_{\bar{\tau},\sigma_\tau}) & \bar{\tau} > 0 \,\&\, \sigma_\tau > \varepsilon \end{cases} \tag{3.3}$$

where P is the probability that a value within $[\bar{\tau}-1, \bar{\tau}+1]$ falls under the curve given by the normal distribution N with $(\bar{\tau}, \sigma_\tau)$ [**?**]. There are two special cases. When all values of τ_j are 0, S_k cannot classify any concept; hence the metric returns a zero score. When $\sigma_\tau = 0$, the subtree is totally balanced; hence the metric returns one. As σ_τ approaches zero, the function P becomes numerically unstable, we use a cut-off value ε to prevent this. In this work, we set $\varepsilon = 0.00001$. The normal distribution is preferred over a χ-test, as χ may not be reliable when l_k is a small number.

Number of Subclasses. All visual channels used in glyphs have limited discriminative capacity. A higher number of subclasses in a scheme would require visual encoding to have more codewords (e.g., more colors or more shape types), which will increase users' cognitive load in learning, remembering, and recognizing the codewords. It is thus more desirable to have a smaller number of subclasses, except that a categorization scheme with fewer than 2 sub-classes is useless. Let $\eta^\top$ be an up-limit for the number of codewords, which is set to 10 in this work. We introduce the following metric to measure the discriminative capacity of a scheme:

$$M_4(S_k) = \begin{cases} 0 & \eta_k < 2 \\ \frac{\eta^\top - \eta_k + 2}{\eta^\top} & 2 \le \eta_k \le \eta^\top \\ \frac{1}{\eta^\top} & \eta_k > \eta^\top \end{cases} \tag{3.4}$$

Although the above four metrics have been normalized to ensure their functional values within the $[0,1]$ domain, the distribution of the values for different schemes can still be rather application-specific and may vary substantially between different metrics. This may lead to inconsistency in combining these metrics.

We thus provide an optional linearization filter for these metrics by mapping the values obtained using a each metric M_i into fractional ranking numbers, which are then normalized into the $[0,1]$ domain with 1 being the best and 0 the worst. For example, consider a set of six schemes with metric values:

$$(S_1, 0.5), (S_2, 0.3), (S_3, 0.54), (S_4, 0.8), (S_5, 0.85), (S_6, 0.6).$$

We first sort the set:

$$(S_2, 0.3), (S_1, 0.5), (S_3, 0.54), (S_6, 0.6), (S_4, 0.8), (S_5, 0.85)$$

Table 3.1: A fragment of the input document passed to the taxonomy generation algorithm. Schemes are grouped by common names preceding the semi-colon, e.g. *S1:On Material* refers to schema 1 and *On Material* is the classification.

Process Name	**Occurrences**	**S1:Material**	**S1:Data**	**...**	**S6:in vitro**	**S6:in vivo**	**S6:in silico**
labeling	390811	1		...	1		
nucleic acid extr.	350267	1		...	1		
hybridization	345671	1		...	1		
feature extr.	267044		1	...			1
bioassay data trans.	176347		1	...			1
grow	116194	1		...		1	
pool	68004	1		...	1		
...	...	...	...	...	...	...	...

We then obtain the normalized ranking values via the *distance for ordinal variables* function $\sigma = \frac{r-1}{R-1}$ where R is the top rank and r is the rank position for each schema.

$$(S_2,0),(S_1,\frac{1}{5}),(S_3,\frac{2}{5}),(S_6,\frac{3}{5}),(S_4,\frac{4}{5}),(S_5,\frac{5}{5}).$$

We denote this filter as a function $R(S_k,M_i,\mathscr{S})$. Using the above set of metrics in conjunction with the filter R, we can derive a combined metric as

$$M(S_k) = \frac{\sum_1^4 \omega_t R(S_k,M_t(S_k),\mathscr{S})}{\sum_1^4 \omega_t}.$$

where $\omega_t, t = 1,2,3,4$ are user-adjustable weights for the four individual metrics. Similar to weights in clustering algorithms, these weights need to be used with care as they introduce additional semantics into the ordering algorithm.

Equipped with the combined metric M, the algorithm for establishing an order of different schemes in $\mathscr{S}$ can be described as follows:

3.4.2 Application to the Biological Case Study

The biological community has built over the years a comprehensive collection of resources to archive experimental data. As molecular biology became a more data intensive field with the advent of DNA microarrays, came the need to store not only measurements but also ancillary annotation describing experimental conditions and set up, thus ensuring a metadata core always shipped with the data set. The sizes of microarray databases (GEO, AE) constitute prime resources for evaluating and testing our approach [**?**, **?**]. The content of ArrayExpress was therefore accessed obtaining data via parallel calls, converting 21000 experiments and associated experimental metadata to ISA-Tab [**?**, **?**]. This step provided a harmonized format in which to represent not just transcriptomic data, but also genomic, proteomic, metabolomic and other classical experiment types.

Given the data sets, the code analyzes the content of these directories to determine the processes and IOs which exist within the experiments, and the number of times they occur. The analysis revealed 61 processes (a small number resulting from the homogeneity of the database where analysis techniques are targeted primarily towards DNA microarrays) with 1,845,089 occurrences and 8,223 properties of the sample of which 3492 were deemed to be IOs with 486,353 occurrences.

Several distinct, empirical but meaningful, categorizations were devised encompassing a number of facets defining the properties of the experimental process, either in terms of its participants or in terms of key process properties. Classifications, based on features such as the nature of process participants, the granularity scale, the nature of experiments and common types a treatments applied in biological experiments were developed. Some classifications were somewhat informed by the overall assumptions ISA model relies upon, where nodes can be either material or data files and where edges are processes acting upon those nodes. Others resulted from applying a small number of axioms discovered through consultations with domain experts. There were 6 classification schemes in all with a total of 23 sub-classifications, these are detailed in figure **??**. Table **??** shows a snapshot of the input file used in the classification algorithm.

The schemes are not tied to any existing taxonomic tree, may not be orthogonal and/or may be redundant with respect to others. The development of the classifications was not the result of application of any knowledge engineering methods and there is no ontological commitment, although in theory the classification names used could be derived from an ontological framework. The reason for not relying an ontology in the first place was risk mitigation, where use would almost certainly result in an unbalanced tree as occurrence counts for the terms would not be considered, resulting in creation of many glyphs that would never be used.

Figure 3.2: The classification algorithm arranged the classification schemes in the above order.

The creation of the tree shown in Fig. **??** was a largely iterative quest with continuous involvement of domain experts checking to ensure that the classifications assigned were meaningful. From early versions of the classification matrix creation, classifications were removed, added or merged in consultation with those who knew the data domain. For example, in *S2* there were sub-classifications in *Genetic Modification & Labeling* which are types of *Material Combination*. Therefore, these sub-classifications were removed since it would be technically and semantically incorrect to keep them. These early interactions emphasized the importance of having domain experts in the loop, otherwise classifications and subsequent glyphs would not be as well created as they could be.

The algorithm managed to correctly place the classifications where they were expected (checks were made by hand to ensure that the algorithm performed well). Schema *S6* (*in vitro*, *in silico* and *in vivo*) was the best top level classification due to its overall fitness metric of 3.08 with individual metrics found to be: M1 = 1.0; M2 = 1.0; M3 = 0.9; and M4 = 0.18. *S6* was closely followed by *S1* (on material and on data), however after the top level separation, *S1* became redundant as a result of the top level implicitly making a split on data (in silico) and materials (in vitro and in vivo). The algorithm successfully detected this redundancy; hence $S1$ is absent from Fig. **??**.

Table 3.2: Perceptual strength of different visual channels based on levels of organization studied by Bertin [**?**] and Green [**?**], 'popout' effect studied extensively in psychology [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**], and hierarchy effect studied by Navon [**?**] and Love [**?**]. The summary strength of each channel is estimated and it should be considered in conjunction with application-specific conventions and metaphors.

	Visual	**Levels of Organization**				**Popout**	**Hierarchy**	**Summary**
Regions	**Channels**	**Associative**	**Selective**	**Ordered**	**Quantitative**	**Effect**	**Effect**	**Strength**
a) Main	1. Color	Yes	Yes	Yes		*****	Strong	*****
	2. Size		Yes	Yes	Yes	***		****
	3. Shape	Yes	Yes			***		***
	4. Orientation	Yes	Yes			***		***
	5.Texture	Yes	Yes	Yes		**		**
b) Supplementary	6-10 (same as 1-5)						Medium	
	11. Planar	Yes	Yes		Yes	***		***
c) Interior	Contains (a+b))						Low	

Although the algorithm performed as expected, a possible criticism of the algorithm could be highlighted in the placement of *S5* below *S4*. *S5* could have been placed above *S4* in line with the observation that, for the majority of cases, *S5* was selected over *S4*. The reason *S5* wasn't used to sub-classify C1 of *S4* is due to its number of classifications (7) being greater than that of *S4* (3), therefore *S4* was selected primarily on this metric even though the sub-tree balance of *S5* (0.8) was slightly better than that of *S4* (0.68). To modify this behavior, it was necessary to involve domain experts in refining the tree based on their domain-specific knowledge and case-specific requirements.

3.5 Visual Encoding

3.5.1 Perceptual Guidance

A *glyph* is a small visual object composed of a number of *visual channels* which can be used independently as well as constructively to depict attributes of a data record. Glyphs are of a type of visual signs that can make use of visual features of other types of signs, such as icons, indices and symbols.

Although there are several books on sign design (e.g., [**?**, **?**]), they focus on signage in public space, and offer empirical guidance on a large number of issues including standardization, size, location, illumination and so on. Many of these issues are not quite relevant to the need for visualizing the workflows of biological experiments. Here, we draw our design principles mainly from findings in perception, especially in the area of visual search [**?**, **?**].

While the extensive use of signs, icons and pictograms in everyday life reflects their usefulness and effectiveness, several perceptual studies also directly or indirectly confirmed their perceptual and cognitive merits. For example, Franks and Bransford's study on transformation of prototypes [**?**] suggested that humans can learn to recognize glyphs by rules consciously as well as unconsciously. The presence of iconic memory [**?**] may facilitate rapid comparison between glyphs in the same display, whereas it is less so for texts.

Guideline on Semantic Relevance. Bertin [**?**] classified visual channels (which he referred to as retinal variables) into two categories, planar (location) and retinal (size, color, shape, orientation, texture and brightness). Bertin proposed four semantic criteria for determining the suitability of different channels in representing certain types of information. These semantic criteria are: *associative*, *selective*, *ordered* and *quantitative*. These criteria are important guidelines, though there has been disagreement in the literature as to how individual visual channels are judged. For example, Bertin considered shape as a non-selective variable. Research has shown that shapes such as filled rectangles, circles and triangles do not allow the human visual system to identify one shape from another effectively in a rapid action when they form some global structures [**?**, **?**], (they have poor "pop-out" effect). However, the omission of all shapes as a selective visual channel has been challenged, for example, by [**?**, **?**, **?**], who show practice and familiarity can support selectivity with almost any shape.

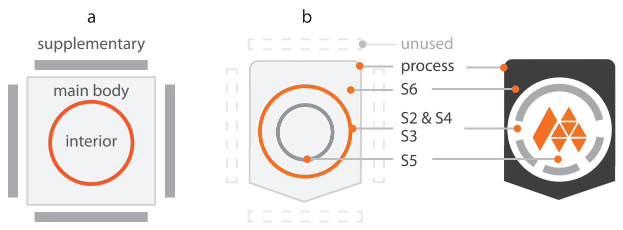


Figure 3.3: (a) The glyph template with a main body, an interior region and an exterior region (consisting of 4 sections). (b) Our final design makes use of the main body and interior region. The exterior is reserved for future extension.

Guideline on Channel Composition. As a glyph is likely to feature a number of visual channels, the constructive composition may affect how individual channels are perceived. A rich collection of literature on integral and separable dimensions shows that the combined dissimilarity of closely integrated visual channels exhibits Euclidean distance $\sqrt{d_a^2 + d_b^2}$ [**?**, **?**], whereas that of separable visual channels exhibits city-block distance $d_a + d_b$ [**?**, **?**]. The latter is more cost-effective than the former in rule-based encoding of multi-faceted concepts, therefore effective glyph design should encompass a non-conflicting set of separable retinal variables.

Guideline on Pop-out Effects. Many classic studies in perception also established the "power" of different visual channels in terms of *pop-out effect* (pre-attentive search), and fixation (during attentive search) [**?**]. The *pop-out effect* is one which allows identification of a target within a few nanoseconds of initial exposure to the visual search space. A result of several milestone studies focusing on observed response times, the ordering of the four commonly used visual channels follows the consensus: color $\prec$ size $\prec$ shape $\prec$ orientation (e.g., [**?**, **?**, **?**]). The symbol $\prec$ reads as *precedes*. However, the strength of color over the other three channels is generally much more noticeable.

Guideline on Visual Hierarchy. *Visual hierarchy*, with which the environment and objects around us are arranged is a well documented theoretical framework [**?**, **?**, **?**, **?**, **?**]. However, the literature contains a debate over the ways in which the visual system traverses this hierarchy. There are four possible ways: top-down (also called global processing) [**?**]; bottom-up (also called local processing); middle-out [**?**]; and salient features (*e.g., edges, points, colors*) [**?**]. Because glyphs are relatively small in comparison with a whole visualization, we consider at such a "localized level", the top-down and salient features may play more significant roles. The top-down assumption suggests that when consider a glyph in isolation, its global feature will affect visual search more than its local features. Salient features are partly addressed by the pop-out effects.

Based on the above-mentioned perceptual guidance, we considered a generic glyph design template for this work as shown in Fig. **??**(a). The template divides a glyph into three regions, namely *main body*, *exterior* and *interior*. In practice, each region can be divided into 2 or more sub-regions (e.g., a twin body glyph or 4 exterior sections), it is convenient to consider the three regions in abstraction. The separation of these regions facilitates the basic separation of visual channels based on the composition guideline, while allowing us to consider them individually according to the hierarchy guideline.

In theory, the exterior and interior regions may also be divided into sub-regions in a recursive fashion though in practice this is tightly constrained or discouraged by the very limited display resolution typically available for glyphs. Similar to the design convention for icons, pictograms are normally featured in the interior region. The exterior region may be further divided in four sections (top, bottom, left and right). If glyphs will be connected to form a network or graph (as in this work), the use of these four sections has to take into account the possible incoming and outgoing connections.

Table **??** summarizes relative merits of some of the most commonly-used visual channels in different regions according to Bertin's categorization, pop-out effects and hierarchy effects. We estimated the overall discriminating capacity of each channel by using a summary rating in the penultimate column. We also recognized the importance of the conventions and metaphors in an application. We will discuss visual metaphors further in Section **??**. We added the last column to highlight the necessity to consider these in a design process.

It is difficult to establish an accurate ranking order of different visual channels by taking all perceptual effects into account in a quantitative manner. The state of the art in perception research is yet to provide all evidence needed for a full and conclusive analysis. Nevertheless, Table **??** can serve a qualitative guidance to glyph designs in this work, providing an ordering of visual channels in parallel with the ordering of categorization schemes discussed in Section **??**.

3.5.2 Mapping Taxonomy to Visual Channels

For each scheme in the taxonomic tree as shown in Fig. **??**, we propose a number of design options. Fig. **??** shows some examples of the proposed designs. For example, the first column shows the use of colors to encode the classes of a categorization scheme. The second column shows the use abstract shapes. Some options convey an abstraction from pictorial representations of classes, and in other cases, we try to establish a metaphoric association between a visual channel and a biological categorization.

Metaphoric visual representations enable domain-specific encoding using "natural mapping" [**?**, **?**]. This natural mapping can make it easier for users to infer meaning from the glyph with less effort required to learn and remember them [**?**]. A recent study showed that visual metaphors can aid memorization of the information depicted in a visualization [**?**]. However, the same study also showed that visually realistic metaphors (those with a lot of detail) may have a negative impact on performance in visual search. Moreover, realistic visual metaphors require a higher pixel resolution, and would lose their discriminating capacity in low resolution conditions.

Based on the design options shown in Fig. **??**, we followed the taxonomic tree in Fig. **??** and identified the best option for each scheme in a hierarchical manner. The evaluation criteria include:

- the discriminating capacities of different channels (Table **??**);
- metaphoric capacity for aiding learning and remembering;
- potential conflicts, including spatial, perceptual and metaphoric conflicts, with visual channels that have already been assigned to other schemes in the tree;
- encoding costs in terms of requirement for pixel resolution.

This process normally takes a few iterations, during which new design options and new metaphoric abstractions and associations are sometimes proposed.

Based on the hierarchy in Fig. **??**, we considered to use color and shape options for S0 (IOs vs. processes) and S7 (four classes of IOs). As introducing 4 main body shapes to encode IOs is not an effective mapping for learning and remembering, we decided to assign outline color of the main body to encode S7, and use two basic shapes, circle and pentagon (a rectangle with a pointer to show workflow direction) to encode S0. Since the main body or the pentagon will only be colored in black or white, S0 also implicitly encoded in using color symbolism, that is, color for IOs and black/white for processes. In effect, S0 is encoded using two visual channels. This redundancy can serve as an error detection in visualization [**?**].

Fig. **??** shows each of the five schemes for the process subtree, and the design option chosen from Fig. **??**. Below we discuss our reasoning for selecting each of the design options for each scheme in the taxonomy.

S6: Process environment. The taxonomic tree suggested a high priority for visual mapping, which is consistent with the domain experts' intuition. We took advantage that black color was not used by S7 for IOs, we assigned a white background to *in silico/in computer* (related to computational processes), and black to *in vivo/in living)* and *in vitro/in glass* (related to materials). Further more, we made use of a shape-based metaphor, fully-filled background for *in vivo* (whole organism), and black background with white cut out for *in vitro* (component of an organism). Together, this was given in Fig. **??** as design option 6. We maintained the overall appearance of the main body of process glyphs in black and white to avoid potential clash with material glyphs.

S2: Types of Material Manipulation. S2 has 5 classes, and we adopted design option 7, which employs visual metaphors that encapsulate strong domain-specific meanings. For example, visual symbol for the *material amplification* class depicts a small segment becoming a large segment.

Figure 3.4: Experimenting with visual channels: an overview of the various design options available for use in representing the different classifications. Most schemes for IO categorization are not shown due to space restriction.

S4: Type of Experimental Perturbation. S4 defines 3 further sub-classes of the *material perturbation* class of S2. Due to the low number of subclasses, we made use of line styles to modify the diamond shape of the *material perturbation*. We made metaphoric association of three line styles as: “solid” line for physically induced perturbation; dash line, a common metaphor for uncertainty and unpredictability, for behaviorally induced perturbation; and wavy line, which is closer to a circle (a visual signature for IOs), for material induced perturbation.

S5: Levels of Material Granularity. This scheme has 7 subclasses (molecule, cellular part, cell, tissue, organ, organism and population), and finding a suitable visual channel was not straightforward. A simplest approach would be to use colors to fill the interior of the glyph, or to use some shape-based encoding. After

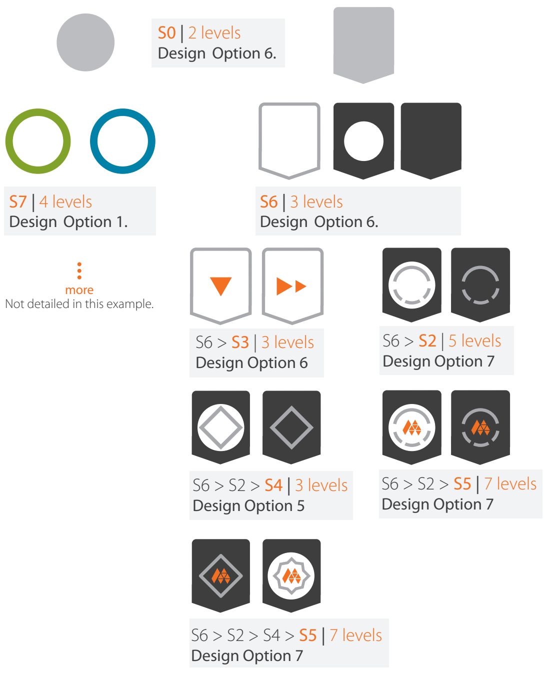


Figure 3.5: Formation of the final glyph design. Top level items require the greatest visual power. It is important to be able to distinguish each of the processes based on their parent level in the taxonomy. Distinguishable global shapes provide the difference between IOs (circles) and processes (squares with pointed bottom to indicate directionality).

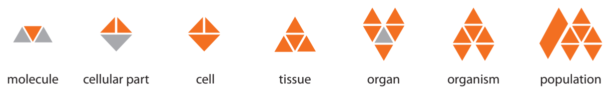


Figure 3.6: Visual representations of the 7 classes in *S5*.

consulting the domain experts, these two options were ruled out. In fact, domain experts preferred some pictograms to represent the 7 subclasses. After considering a number of more realistic drawings, we found that it was not easy to create realistic representations that can differentiate all subclasses (e.g., cellular part vs. cell; organ vs. organism). We designed a special set of icons as shown in Fig. **??** with three visual channels to aid learning, memorization and recognition. The first visual channel is the overall shape and orientation. The second visual channel is metaphoric abstraction and association. For example, the shape of cell part indicates a portion of a cell. Tissue is associated with a patch, organ with an abstract heart shape, organism with an abstract human, population with two abstract humans. The third visual channel is the number of orange sub-shapes, representing the levels 1-7.

S3: Types of Data Manipulation. S3 defines a 3 further sub-classes of the *in silico* class of S6. We use three abstract pictograms to represent data capture, processing and analysis. The encoding makes use of the difference in overall shape, orientation, and number of triangles to aid learning, memorization and recognition. Note that these shapes will not be confused with those for S5 as the white background of the main body provides a distinct context of computing rather than IOs.

Following mapping of all schemes to design options, creation of all glyphs is a straightforward process. Fig. **??** shows all variations of the glyphs associated with the process sub-tree.

We introduced a "crush" test for the designed glyphs by scaling it to different pixel resolutions. Fig. **??** shows some example glyphs display at varying resolutions. One can comfortably see all details at the 40×40 level. At the 10×10 level, one can observe the visual signature of *in vivo* and *in vitro*. Even at the lowest level (5×5), one can still differentiate the two glyphs.

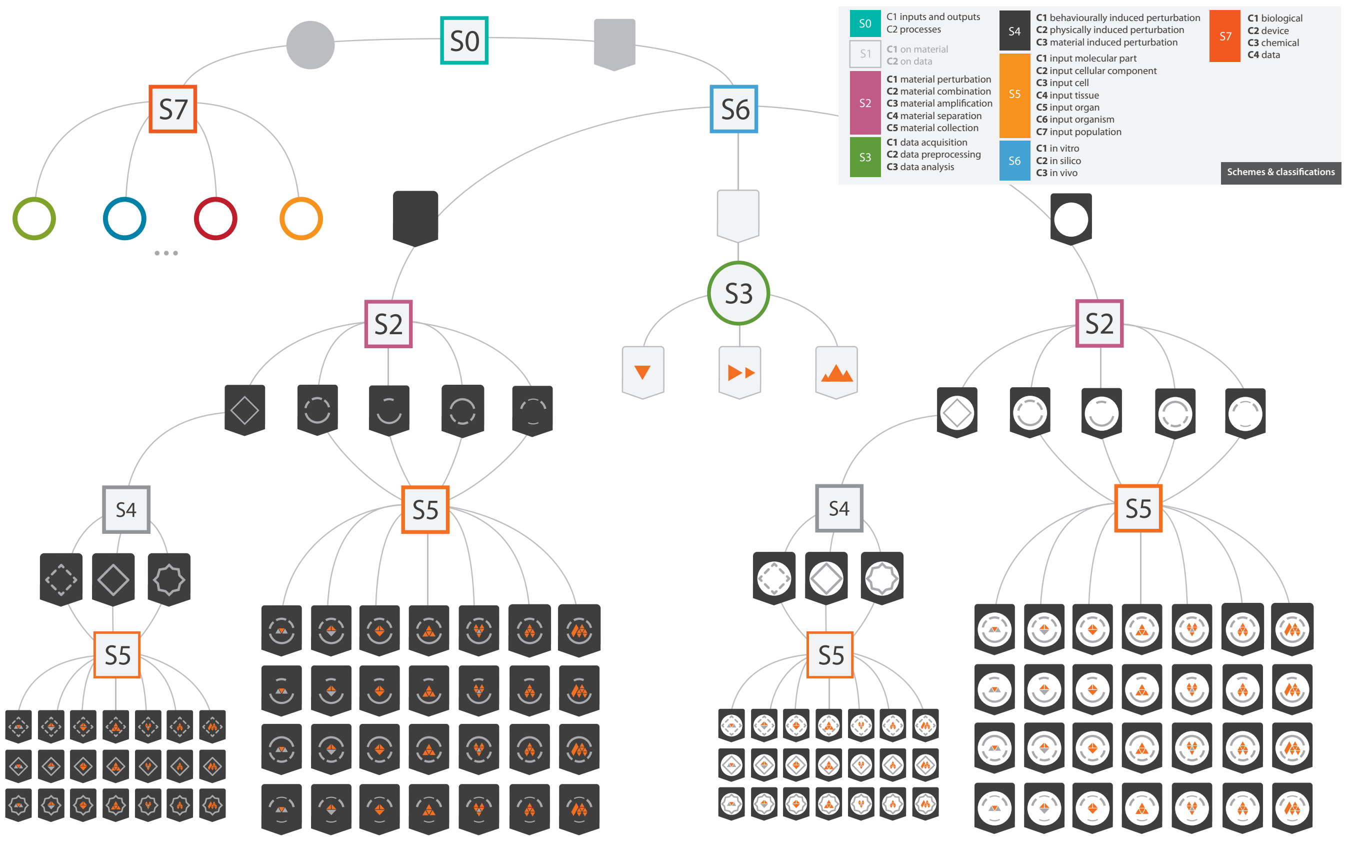


Figure 3.7: Overview of the taxonomy based glyph designs in the context of biodomain.

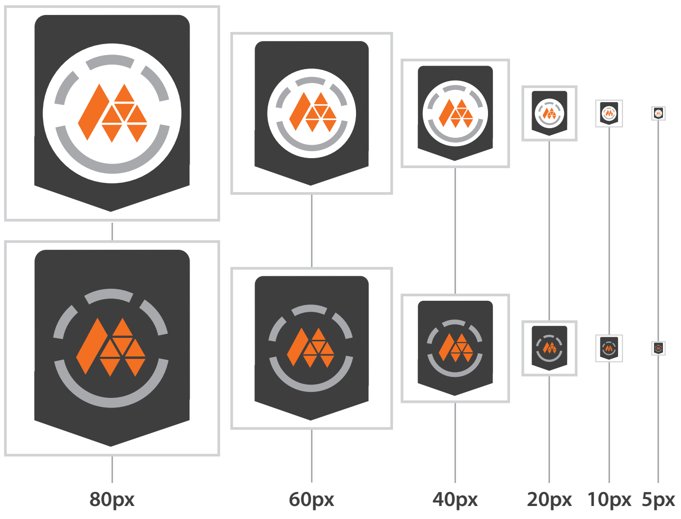
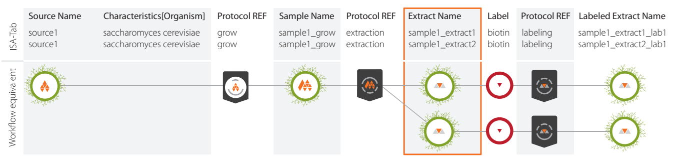


Figure 3.8: A “crush” test of glyphs to evaluate the discriminating capacity of various visual channels at different resolutions.



Source Name	Characteristics[Organism]	Protocol REF	Sample Name	Protocol REF	Extract Name	Label	Protocol REF	Labeled Extract Name
source1	saccharomyces cerevisiae	grow	sample1_grow	extraction	sample1_extract1	biotin	labeling	sample1_extract1_lab1
source1	saccharomyces cerevisiae	grow	sample1_grow	extraction	sample1_extract2	biotin	labeling	sample1_extract2_lab1

Figure 3.9: A workflow is recorded in text form within the ISA-Tab format. Our software translates it to glyph-based visualization. A branching event is automatically detected during the translation.

3.6 Workflow Visualization and ISA Integration

In order to visualize workflows of biological experiments, we had to address the following technical issues: 1) mapping from a name in the metadata to a glyph; 2) creating a workflow visualization with both node placement and connection display; 3) developing a prototype tool for a practical environment such as the ISA tools framework.

The mapping from concept name is achieved by a look-up table which is built from the matrix used in formulating the taxonomic tree and implemented as a tab delimited file. Each concept name is mapped to a text tag encoding the traversal path from the root of the tree to the leaf node corresponding to the name. The tag includes identifiers of the schemes and classes encountered. With the path, a glyph can be constructed dynamically from the pre-defined visual mapping as described in the previous section. The look-up table also enables storage of pre-rendered glyphs in an image format.

To generate the workflow, we made use of the layout algorithm available within the Prefuse visualization framework [**?**]. This framework also brings with it functionality such as panning, zooming and filtering which bring more interactivity to the user and making navigation through large collections of workflows easier. The only requirement for use of Prefuse was a minimal amount of Java code to create the user interface coupled with creation of an XML file format native to Prefuse for representation of the tree structure. This XML format is configurable, we have configured it to contain: *node type* (e.g., process), *node name* (e.g., labeling) and an image to be rendered, which is assigned using a look-up operation through the above mentioned tab delimited mapping file. The order of the XML elements within this file has direct implications on the order these elements are displayed in. Within the ISA-Tab format, there is an implicit time element found in the ordering of the columns in the study sample and assay files. This can be used to construct the XML elements through near direct mappings of processes and IOs, a workflow which is illustrated in Fig. **??**. Recognizing branching events is an important part of workflow visualization as illustrated in **??**. In software, these branch events can be identified when the preceding nodes of a process have the same names while the succeeding output nodes are different. Fig. **??** highlights one such case, where branching occurs after extraction of 3 materials from one sample.

Our software reads text-based ISA-Tab files as illustrated to create the XML notations required by Prefuse for rendering the experiment workflow illustrated in Fig. **??**(b).

The software is implemented as a Java application capable of processing ISA-Tab files to create experimental workflows using glyphs. For the purposes of broad dissemination in the near future, we have integrated the workflow visualization directly inside ISAcreator, a Java desktop application for creating and editing ISA-Tab files. Furthermore, the standalone workflow visualization shown in figures **??**a and b makes use of the same ISA-Tab parser as available within ISAcreator. The integration involved the development of user interface elements for ISAcreator software to access the workflow visualization. From the spreadsheet view, users may highlight rows (assays), right click and select an option named “View workflow for assays” to visualize the entire processing pipeline for the corresponding samples as shown in Fig. **??**(b).

3.7 Conclusions and Future Work

Although it was established in psychology that people can learn rules of glyph encoding consciously as well as unconsciously [**?**], we do not expect users to learn and remember these glyphs without help. We thus allow users to find detailed text descriptions in pop-up windows interactively. As shown in Fig. **??**, not only do

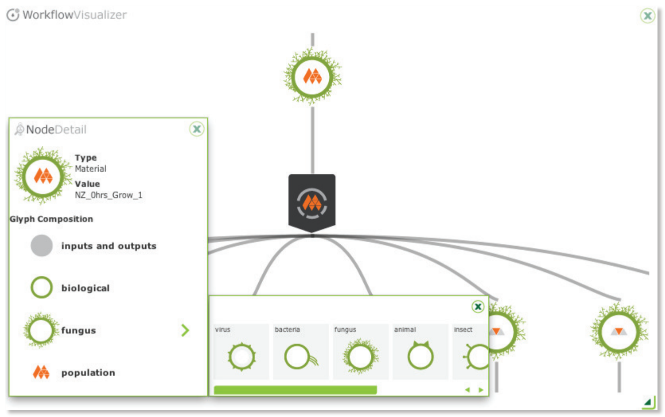


Figure 3.10: User can interact with a workflow to view the text descriptions of individual glyphs in pop-up windows, which are also dynamic legends showing how the concept is categorized and how the corresponding glyph is decomposed into different elementary visual representations. Users can also use such a legend to find out other classes in a categorization scheme.

these windows provide names of processes and IOs, they also serve as dynamic legends, showing how each glyph is decomposed into different elements corresponding to categorization schemes. One can also select a component to view all classes in a scheme.

In this manuscript, we presented a systematic approach for glyph design by using taxonomy as a guide. We demonstrated our approach by analyzing the content of a biological database and showed how our tree-building algorithm could be used to take a large collection of concepts and generate a taxonomic tree, which provides an ordering of a set of categorization schemes (i.e., attribute dimensions). This enabled us to draw a parallel between the ordering of attribute dimensions with the ordering of visual channels compiled from the perception and visualization literature. We involved domain experts in refining the tree as well as in creating metaphoric abstraction and association for glyphs. This work was followed by development of a prototype tool for glyph-based visualization of experimental design workflows as found in biology. The prototype is integrated in the ISA suite of tools to be disseminated to users.

We plan to further the present work to include the provision of "macros" in workflows in order to make graphs more compact and create dynamic web-based rendering of the workflows using vector graphics. We also intend to carry out field-based evaluation through ISAcommons, the user community of ISA-tools framework. Like all potential diagrammatic schemes, we expect that it will take many iterations before it can become a standard. Nevertheless, the development of an online visualization tool will make such a process more efficient.

Chapter 4

Visual Compression of Workflow Visualizations with Automated Detection of Macro Motifs

4.1 Introduction

The term 'macro' is derived from the Greek word *makro* meaning *big* or *far*. In computer science, the term is defined as "a single instruction that expands automatically into a set of instructions" [**?**]. It is commonly used as a noun (*e.g.*, a macro), or an adjective (a macro command). In schematic diagrams, such as electronic circuit diagrams, data flow diagrams, and control engineering block diagrams, *macros* are commonly used to provide hierarchical concept abstraction as well as visual compression. Not only can macros facilitate "overview first, details on demand" in visualization [**?**], but they can also speed up visual search and reduce cognitive load for experienced users, who are knowledgeable about or have become accustomed to the specification of individual macros. In effect, their functionality bears some resemblance to *acronyms*.

This work is motivated by the need to reduce the visual complexity of biological experiment workflows in an extension to work conducted by Maguire *et al.* [**?**]. Such workflows describe the sequence of processes (arranged with respect to a temporal dimension) enacted on biological materials and signals obtained through experimental observations. Large repositories of experimental data offer a data corpus that can be tapped into for workflow analysis and, more specifically, detection of commonly-used subgraphs (also referred to as *motifs*). Success in "compressing" such recurring motifs using macros would significantly reduce the time required for creating, and perhaps more importantly, viewing and comparing workflows.

When sketching out workflows on paper, individual scientists often abstract a commonly performed sequence of steps into a macro procedure or represent a set of parallel steps by using a macro step. Such a macro encapsulates the sense of a bigger block, a higher-level abstraction, and multiple steps, just like in programming and other schematic diagrams. Despite the potential benefits of using macros, one major stumbling block that hinders the availability and use of macros in workflow visualization is the lack of standards. Nevertheless, steps towards standardization can be taken. With the availability of large collections of workflows in biological experiment repositories, computational approaches may be applied to detect commonly-used motifs (*i.e.*, topological patterns in workflows). It provides domain experts with an objective means for establishing a list of candidate macros based on their usage. The final selection of macros can be determined semantically in traditional ways (*e.g.*, community consensus, popularity ranking or recommendations by standards bodies).

The main contributions of this work are as follows:

- Our overall approach for using a computational methodology to identify candidate macros in relation to a workflow repository is new. This enables exhaustive search and objective selection of candidates while still allowing domain experts to make the final decision on macro creation.

- We propose an efficient algorithm for extracting motif patterns in workflows by taking into account node and edge types. This enables motif grouping through inclusion of semantic context. Our tests show that this type-sensitive algorithm performs faster than existing generic algorithms in the literature; and
- Our motif extraction algorithm is based on a finite-state machine. As workflows are directional and acyclic, the state transitions for identifying a motif encode the structure of the motif. We make use of such information to characterize the structural pattern of selected macros visually. This facilitates partial automation when designing an individual glyph for each macro.

4.2 Related Work

Schematic representations of workflows are commonplace in a range of disciplines. A workflow typically describes a sequence of steps, followed from initiation to completion when conducting a piece of work. Perhaps the most widely used workflow visualization is the *Gantt chart*, which depicts tasks, resources and their dependencies in a temporal manner. Efforts such as VisTrails [**?**], VTK [**?**] and SmartLink [**?**] make use of workflows to depict the processes followed to create a visualization. In Taverna [**?**] and Kepler [**?**], workflow visualization allows users to build reproducible pipelines for data analysis. Other workflow visualizations include the Business Process Model and Notation (BPMN), Petri-net, and programming flowchart, all of which convey work flow, data flow and process interactions within often very complex systems.

In this work, we consider workflows used to describe biological experiments. This class of workflow visualization renders the processes enacted on biological materials in experimental setups, from sample collection through experimental perturbation to signal acquisition and interpretation. While most scientists have been using generic text-based graph drawing tools such as GraphViz [**?**], new workflow visualization tools are emerging (*e.g.*, [**?**]).

Graph reduction is a family of algorithms and techniques for reducing visual complexity by using graph filtering and graph aggregation [**?**]. *Graph filtering* involves removal of certain nodes and edges from the graph, either deterministically or stochastically [**?**, **?**]. *Graph aggregation* selectively merges two or more nodes into one, hence preserving some information about the nodes and edges to be removed. Many selection algorithms exist, such as methods for building hierarchical levels of detail, clustering based on node/edge attributes and edge bundling for clutter reduction [**?**, **?**].

One subset of graph reduction techniques is motif based. Motifs, in the context of graphs, are "patterns of interconnections occurring in complex networks" [**?**, **?**]. A considerable amount of effort has been dedicated to the automatic identification and characterization of meaningful motifs in individual graphs or sets of graphs (*e.g.*, [**?**, **?**, **?**, **?**, **?**, **?**, **?**]). Replacing recurring motifs with macros can provide hierarchical concept abstraction, visual compression, improved readability and cost-effective task performance. Macros feature extensively in various graph representations, such as schematic diagrams, communication network diagrams and workflow diagrams. For example, VisMashup [**?**] utilizes macros to simplify the large body of steps required to create a visualization. Dunne and Shneiderman propose to simplify graphs by using fan and arced glyphs to represent common topological structures [**?**]. Shneiderman and Aris propose the use of user-defined semantic substrates for compressing network visualization [**?**]. However, determining suitable macros usually depends on both the semantic content of the corresponding motif and its potential use in graph reduction. Hence, relying solely on topological information may not lead to meaningful visualization, while relying solely on user input does not scale up to a large collection of graphs.

As shown in Table **??**, many motif search/discovery algorithms exist. Ribeiro *et al.* [**?**], Kashani *et al.* [**?**] and Wong *et al.* [**?**] published benchmarks detailing processing time for a subset of the algorithms in Table **??**. As subgraph matching is fundamentally an *NP*-complete problem, these algorithms are computationally intensive. Wong *et al.* report *maVISTO* taking 14,000 seconds to find size three motifs in an *E. Coli* network. Comparing that to one second for FANMOD to analyze the same network, one can see the huge variability in algorithm performance. As the size of the target motifs grows, performance decreases. Using the same *E. Coli* network but searching for size 8 motifs, FANMOD will need 9000 seconds to finish the operation [**?**].

Table 4.1: Some commonly used motif finding algorithms.

Algorithm	Sampling	Node Limit	Focus
mfinder [?]	exact & estimated	6	discovery
FPF (mavisto) [?]	exact	4	search
FANMOD [?]	exact & estimated	8	discovery
NeMoFinder [?]	exact	12	search
MODA [?]	exact	not defined	search
G-Tries [?]	exact	> 9	discovery
Grochow-Kellis [?]	exact	9	search
Kavosh [?]	exact	8	discovery
Color-coding [?]	estimated	10	discovery

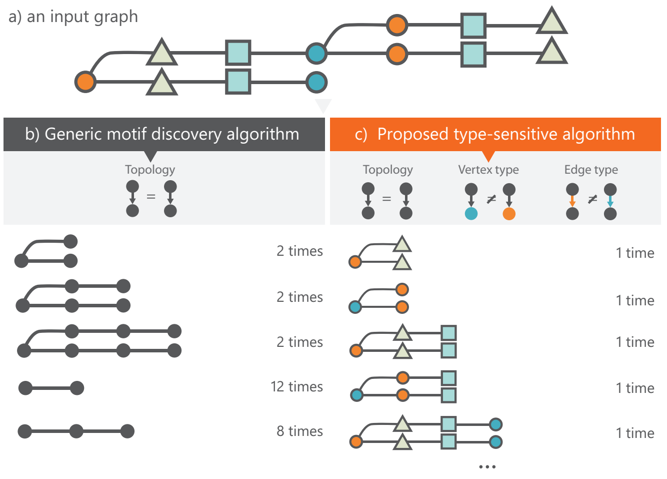


Figure 4.1: (a) A graph, with typed nodes and edges, where different node types are mapped to different shapes and colours. (b) Generic motif discovery (or search) algorithms focus on topological differences. (c) Our motif extracting algorithm takes varying node/edge types into account, yielding semantically aware motif grouping.

In Table **??** the second column indicates whether the sampling in a search space is exact (enumerating all possible candidates) or estimated. The third column indicates the maximum number of nodes in a motif the algorithm can handle. The final column indicates whether the algorithm is focused on *motif discovery* (finding repetitive patterns in a set of graphs), or *motif search* (finding a given motif in a set of graphs). All these algorithms focus on topological patterns in graphs only and do not consider the types of nodes and edges as a search constraint. As illustrated in Figure **??**, these generic algorithms typically focus on topological patterns in a graph. However, the types, or semantic categories of nodes and edges are an interesting property in defining a macro. There is a more semantically aware motif search algorithm implemented in VisComplete [?, ?] that compares node labels and node order to predict the next step in a VisTrails pipeline analysis over a larger corpus of pipelines. However this algorithm is topologically less sensitive and provides no way to identify branch/merge events in a motif, or edge types. A potentially useful algorithm for identifying suitable candidate macros should be type as well as topology sensitive. Our work focuses on such an algorithm, offering additional advantages in computational performance and providing visual mappings with meaningful structural information.

In our work, when considering whether a subset of steps in a given workflow is the same as another subset in the same or a different workflow, we must consider the semantics attached to each step (see Figure **??**). Consequently, the task of motif discovery and search is highly constrained by the types of nodes and edges. Therefore, it is necessary, as well as advantageous, to develop specific motif extraction algorithms for workflow visualizations. In addition, we propose a novel concept of partially automating the design of macro glyphs. We consider the use of multi-resolution glyphs to depict macros at different levels of detail when a user interacts with the visualization, a technique referred to as semantic zooming [?, ?].

4.3 Motivation and System Overview

Over the past two decades, Biology has benefited from entering the digital era, becoming a data intensive field. Ancillary to this development, important efforts have been undertaken to preserve and curate digital artifacts in biology, including experimental workflows. A workflow is a form of directed graph (digraph). Scientists mainly rely on two types of workflow visualization: digraphs with text labeled boxes or glyph nodes. The latter enables more compact visual representation than the former [?] allowing domain experts to perform their tasks such as error checking and comparison more quickly. However, workflows can still be quite large and complex, containing many repeated subgraphs, which demand some effort for identification in 'flat' representations. Hence, it is highly desirable to introduce macro representations in workflow visualizations, as both text-based and glyph-based workflow visualizations can benefit from macro based visual compression. Although experimental workflows in biology exhibit specific properties that are different from workflows in other disciplines, they often share some characteristics. Almost all exhibit temporal ordering and most feature only acyclic digraph topologies. We are therefore hopeful that our algorithm and experience can be transferred to workflow visualizations in other disciplines.

The domain experts involved in this work (also co-authors of this paper) identified the following requirements:

1. Given a number of workflows, can we obtain the most common patterns/motifs within those workflows?
2. For each motif, can we obtain the occurrence frequencies for that motif and information about which workflows they occur in?
3. Can we automatically create macro representations of those motifs, with the possibility of adding extra annotation to them?
4. Can motif patterns in a workflow be substituted by common macros to make the workflow more compact?

Depending on discipline and requirements, macros may be created by individuals based on their own knowledge about needs and usage. However, such an individualized approach would be impractical if applied to the curation of large collections of experimental workflows as found in data repositories. Yet, the availability of such data provides an opportunity for the computational identification of commonly occurring motifs in workflows. The statistics of such motifs offer useful guidance to motif selection when defining macros.

To carry out this work, information was extracted from a curated resource currently holding over 22,000 experimental design workflows (ArrayExpress) [?]. We used a subset of this collection, comprising 9,670 workflows considered to be well-formed with respect to correct connectivity and semantic annotation. This subset of workflows, available in the ISA-Tab format [?, ?], offers a good representation of experiment typology. These ISA-Tab files were processed and transferred to a graph database, which provides an optimized environment for storing graph structures and a query language optimized for graph traversal [?]. For this purpose, we selected *Neo4j* [?] – a freely available graph database providing a query language called Cypher, fast traversal algorithms and high scalability (up to several billions of nodes and edges on a single computer).

Our system consists of three main functional steps, as depicted in Figure **??**. (B, C, D) We algorithmically scan all workflows in the collection and extract all valid subgraphs as candidate motifs. The patterns and occurrence statistics of these motifs are recorded. The definition of validity and the extraction algorithm will be detailed in Sections 4.1 and 4.2. (E, F) We provide a user interface for domain experts to define macros by selecting motifs from an ordered list of candidates, based on the statistics computed as well as their domain knowledge about individual motifs and the context of their usage. This will be discussed in detail in Section 4.3. (G, H) We provide an automatic means for defining the pictogram of a macro by making use of the state transition information captured by the algorithm during the motif discovery phase. In addition, each macro will normally be displayed with a text label, which is defined by the domain expert through the user interface. The mechanism for automatic creation of a pictogram will be detailed in Section 5.

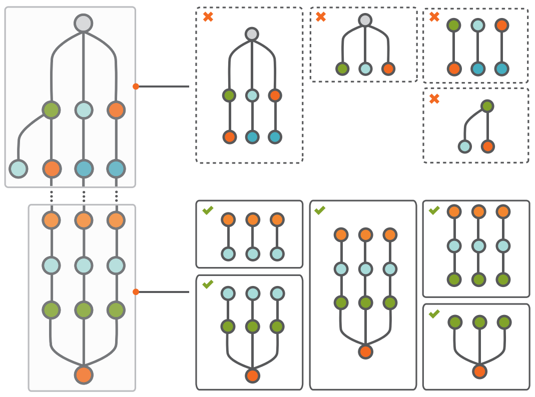


Figure 4.2: In a workflow graph, some subgraphs can be considered as “legitimate motifs” while others cannot, depending on node type. Invalid motifs do not have the same output node type, as indicated by the differing colors. Conversely, valid motifs are those that have the same input and output node type.

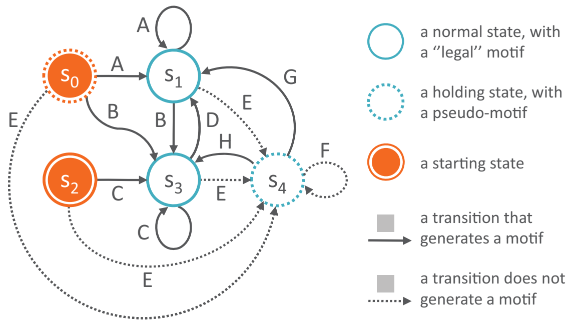

A
A
S0
S1
G
E
B
B
D
E
H
F
S2
C
S3
E
S4
C
E
a normal state, with a ''legal'' motif
a holding state, with a pseudo-motif
a starting state
a transition that generates a motif
a transition does not generate a motif

Figure 4.3: The state transitions in motif generation.

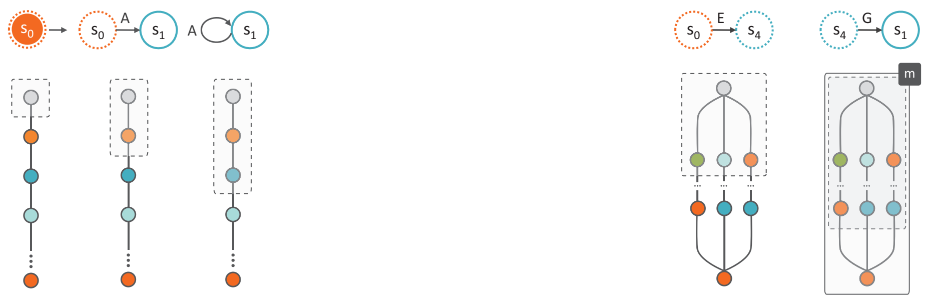
We created a tool, integrating the above requirements. The tool also allows for substitution of frequent motifs with their macro counterparts in experimental workflows. Furthermore, the tool can be deployed as a standalone software package or exposed through an API.

4.4 From Motifs to Macros

Motif discovery and substitution in graphs typically consists of four processes [**?**, **?**]:

1. *Subgraph Generation*: Scan each graph for all possible n-node subgraphs.
2. *Motif Amalgamation*: Group together subgraphs that are topologically equivalent and generate a representative subgraph of each group (a motif).
3. *Macro Selection*: Assign a significance value to each motif with respect to other motifs in the collection (for instance, by computing frequency of occurrence) and select the most significant motifs as macros.
4. *Macro Substitution*: Search graphs for subgraphs that match with macros and replace them with that macro.

In applications such as biological network rendering, it can be safely assumed that subgraphs should be connected. By contrast, in applications such as very-large-scale integration (VLSI) design, such a condition is sometimes relaxed, as macros are often used to group elements for a cost-effective spatial placement. In general, the number of subgraphs can be rather large. That is why many algorithms only deal with small subgraphs, such as the 4-node subgraphs studied in [**?**].



(a) singular branching (b) homogeneous branching (c) heterogeneous branching

Figure 4.4: Examples of state transitions with different rules. Homogeneous edges are depicted by using the same color.

4.4.1 Candidate Macros in Workflows

The workflows and macros considered in this work exhibit the following characteristics:

1. A workflow is an acyclic digraph.
2. A macro must consist of at least two nodes/nodes.
3. Macros are not only topologically sensitive, but also semantically sensitive. In other words, two subgraphs are said to be in the same motif group only if they are isomorphic and every pair of corresponding nodes (and edges) are of the same type.
4. It is not necessary to have a path between every pair of nodes in a motif.
5. For each node in a macro, there must exist a path from the macro's input node, and a path to the macro's output node.
6. Each macro must have a single input type and a single output type (including *bundled* input and output).
7. A bundled input or output may contain any number of connection edges, but they must be of the same material type.
8. A macro may receive a bundled input converging from different preceding nodes and may deliver a bundled output branching off to different successive nodes.

By relying on these characteristics we can significantly reduce the number of subgraphs and motifs to be extracted in the *motif generation* stage of the algorithm. Figure **??** illustrates those subgraphs that are 'legitimate' candidates and others that are 'illegitimate'. In addition, owing to characteristic three described above, the *Motif Amalgamation* and *Macro Substitution* stages are much less onerous in comparison to subgraph matching based on topology only.

4.4.2 Motif Generation

Owing to the aforementioned conditions that are specific to workflows, we could not make use of existing motif generation software and algorithms such as those surveyed in [**?**]. A specialized motif generation algorithm was needed, and for that we adapted the widely accepted 'pattern-growth' approach [**?**]. We describe the algorithm by evolving a state transition diagram as shown in Figure **??**.

Singular Branching (Rule A). As shown in Figure **??**(a), the simplest workflow is perhaps a single path composed of n different steps (*e.g.* experimental steps). The algorithm can be activated from any node and will only move forward, following the flow of the work. Since a single node cannot be a macro, the search will park at state S_0 as illustrated in Figure **??**(a), where the orange background indicates a starting state and the dotted outline implies that it is only a holding state and does not output a "legitimate motif".

When the algorithm encounters the next node n_1, it obtains the first 'legitimate motif', $m_1 = \langle n_0 \rightarrow n_1 \rangle$, and moves to the state S_0. The edge is labeled with A indicating that this follows rule **A**. From n_1, the algorithm then encounters n_2, it outputs another motif $m_2 = \langle n_0 \rightarrow n_1 \rightarrow n_2 \rangle$, and remains at the same state. For the workflow in Figure **??**(a), this self-loop continues to generate motifs in growing sizes until the end of the path or the number of nodes in the motif reaches a predefined maximum.

Homogeneous Branching (Rules B, C, D). One extension of the singular branching case is multiple runs of the same sequence of steps, in parallel. When the work flows forward, the same type of edges connect to the next set of nodes. These edges can consist of bundled together as an input or output of a macro motif (see condition 7 in Section **??**).

When an edge first branches to multiple nodes, as illustrated in Figure **??**, the algorithm moves from state S_0 or S_1 to S_3, following a transition of singular branching to bundled homogeneous branching. This is referred to as rule **B**. S_3 indicates more than one edge is being bundled at this state and the number of edges may vary.

The algorithm will self-loop as long as the motif grows with such bundled edges. Rule **C** indicates a transition within the state of homogeneous branching. The number of edges in an edge bundle can change as long as there is more than one edge and they are of the same type.

When all bundled branches converge to a single node, the algorithm returns to state S_1. This transition is referred to as rule **D**. Figure **??**(b) shows three examples of applying rules **B**, **C** and **D** respectively. Applying any of the three rules will result in a bigger motif than the previous one.

An additional state S_2 is included for a scenario when the algorithm starts with a row of parallel nodes with bundled input and output. We will discuss this scenario later in the context of reactivating the algorithm for bundled edges.

Heterogeneous Branching (Rules E, F, G, H). Recall conditions 6 and 7 in Section **??**: a macro must have a single input and single output and each can be bundled edges of the same material type. When these two conditions are met, we can have heterogeneous flow within a macro. This subset of rules is designed to 'grow' this particular type of motif.

Rule **E** is applied when the algorithm first encounters an heterogeneous pattern. This transition leads to a holding state S_4 and it does not generate any motif. In this state, all nodes scanned so far are grouped as an interim pseudo-motif and output edges are placed in an interim pseudo-bundle.

The interim state is maintained by a self-loop transition; *i.e.*, rule **F** as long as the bundled edges remain heterogeneous. When the edges in an interim pseudo-bundle finally converge to a single node or a set of nodes with homogeneous output edges, the algorithm can leave the holding state S_4. Rule **G** defines the transition to singular branching state S_1, while rule **H** defines the homogeneous branching state S_3. Both rules will generate a new motif, which includes all nodes in the interim pseudo-motif and the newly encountered node(s).

Termination and Reactivation. The algorithm strictly follows the breadth-first search strategy. It may terminate in two situations: (a) when the predefined maximum depth that a macro may be at is reached; (b) when it encounters a node without an output edge (*i.e.*, the termination point of a workflow). The termination condition is tested in all states in Figure **??**.

It is necessary to reactivate the algorithm by choosing each of the different nodes in a workflow as a starting node at state S_0. In addition, each bundle of edges of the same material type can also be a starting point at S_2. Although theoretically appropriate, invoking the algorithm recursively from a starting node of a workflow proved not to be feasible in practice. We therefore made use of a queue, which initially contains all nodes in a workflow. We fetch nodes from the queue one at time to invoke the algorithm from S_0. Every time the algorithm reaches state S_3, we have a set of homogeneous nodes that were just encountered (*e.g.*, $[n_{1a}, n_{1b}, n_{1c}]$ in Figure **??**(b)). We store all k-node subsets of such a set ($k = 2, 3, \ldots$) in a list. When we finish with all individual nodes in the queue, we sort the list and remove redundant subsets. We then invoke the algorithm with the sorted and cleaned list from S_2. Note that each subset in the list is a 'legitimate motif, hence S_2 is not a holding state.

Performance. To evaluate the performance of the algorithm, we tested it against nine graphs representing biological workflows of varying sizes on a MacBook Pro with a 2.53GHz Intel Core i5 CPU and 8GB RAM. The nine graphs were divided into three groups, three small, three medium and three large, based on their node and edge counts. Figure **??** shows the distribution of workflows in our collection in terms of nodes (x-axis) and edges (y-axis). 93% of all workflows have a node count below 1000 and the average number of nodes per workflow is approximately 322. Given these statistics, we define *small* as 200 nodes or below

Table 4.2: Performance of our motif-finding algorithm on graphs of varying size and at a range of search depths. Averages (last column) are taken across three graphs G1, G2 and G3 in each category. For each graph at each depth, the recorded time was the average time over five runs. A more detailed table is available in the supplementary materials.

Graph Size	Motif Depth	G1	G2	G3	Seconds
Small	3	0.018	0.017	0.024	0.02
	4	0.034	0.026	0.025	0.03
	5	0.048	0.038	0.042	0.04
	6	0.059	0.041	0.046	0.05
	7	0.075	0.049	0.049	0.06
	8	0.086	0.063	0.06	0.07
	9	0.095	0.064	0.062	0.07
	10	0.106	0.069	0.072	0.08
Medium	3	0.153	0.065	0.056	0.09
	4	0.212	0.097	0.085	0.13
	5	0.24	0.13	0.115	0.16
	6	0.293	0.161	0.138	0.2
	7	0.343	0.189	0.159	0.23
	8	0.389	0.219	0.178	0.26
	9	0.429	0.241	0.183	0.28
	10	0.477	0.287	0.192	0.32
Large	3	0.296	0.756	0.354	0.47
	4	0.393	1.062	0.45	0.64
	5	0.652	1.309	0.414	0.79
	6	0.632	1.528	0.42	0.86
	7	0.75	1.829	0.341	0.97
	8	0.809	2.028	0.396	1.08
	9	0.889	2.287	0.528	1.23
	10	1.047	2.489	0.604	1.38

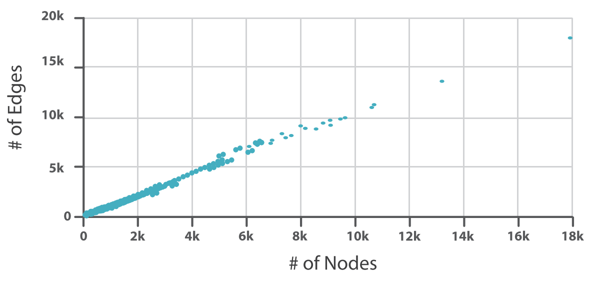


Figure 4.5: Scatter plot showing the distribution of workflows (each depicted as a point) as node count versus edge count.

(covering 58% of workflows); *medium* between 201 and 600 nodes (covering a further 28% of workflows); and *large* over 601 nodes (covering the remaining 14%).

For each of the nine selected graphs, motifs between a depth of three and ten were searched for, with each search repeated 5 times to account for any variability. Table **??** gives the runtime (in seconds) of applying our algorithm to the nine test graphs. The runtime is scalable to our collection of some 10,000 workflows, as the algorithm runs as a batch process. It also shows that the speed of our motif finding algorithm compares favorably the current best general purpose motif finding algorithms.

Our algorithm search space differs from the general purpose algorithms listed in Table **??**, since we have introduced specific constraints on motif structure, taking into account the notion of node/edge types. Nevertheless, we could have theoretically used a two-stage method, by first using a general-purpose algorithm to identity an initial list of motifs, then filtering out those motifs that do not meet our requirements. Since our algorithm is generally faster than these general purpose algorithms, plus the computation to infer back the node/edge type is potentially large, there is no advantage to using this two-stage approach.

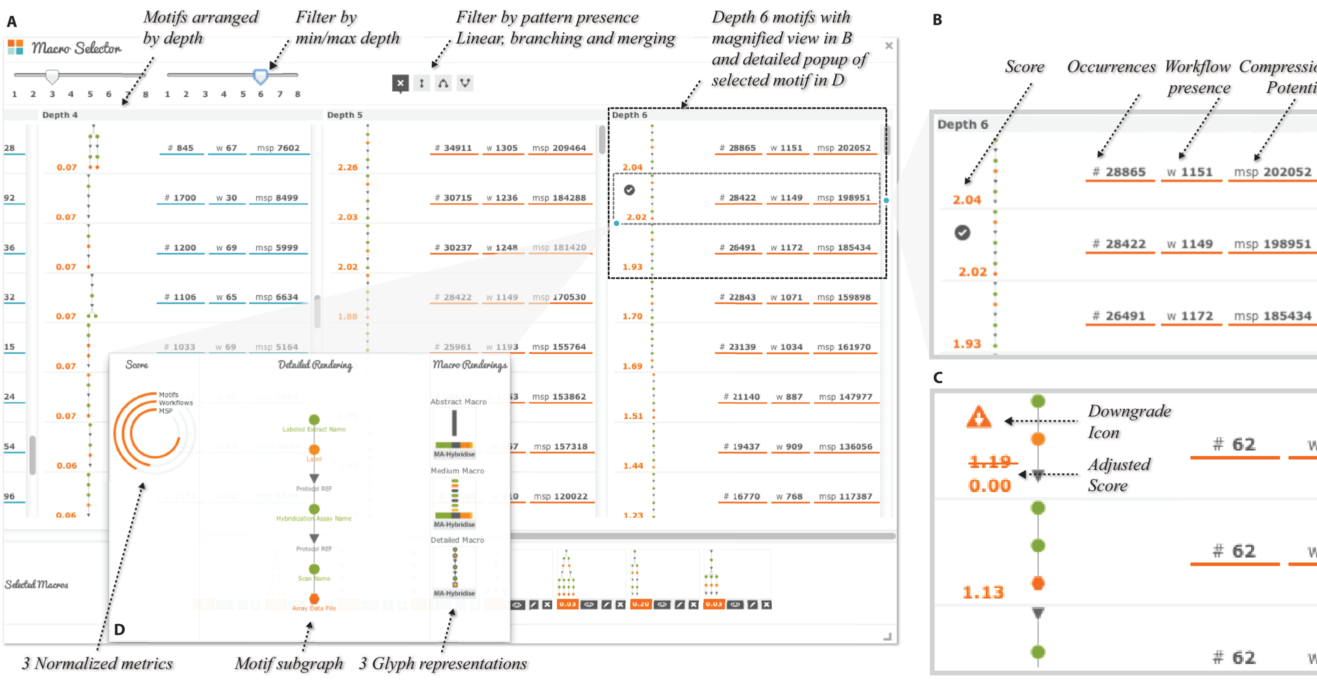


Figure 4.6: AutoMacron provides a user interface (A) for domain experts to select macros from a list of computed motifs. As shown in the detailed view in (B), the overall score determines the order of motifs in the list. The three indicators are shown in unnormalized form in order to be semantically meaningful. The detail view (C) shows an example where a score is adjusted dynamically when a motif encompassing other motifs is selected. (D) shows a pop-up window for a specific macro, detailing its subgraph and three automatically generated pictograms for different levels of details in visualization.

4.4.3 Selecting Macros from Candidate Macros

Given a list of motifs extracted using the algorithm in Section **??**, we can categorize them into individual groups. Motifs in each group share the same subgraph topology, and the same set of nodes and edges in terms of their numbers and types. Each motif group thus becomes a candidate macro. It is important to emphasize that selecting macros from macro candidates requires a fair amount of knowledge about biology, biological experiments and the uses of workflows. In some cases, other information, such as the time when and context in which certain motifs appear frequently, may also feature in the selection process. Therefore, it is not sensible to make this selection process fully automatic. In this work, we provide a user interface to assist domain experts in selecting macros from a large list of candidates.

Following the advice of domain experts specialized in biological data curation, we understood the essential requirement for such a user interface is to provide key indicators for each macro candidate g_i, including the depth of its subgraph, $A_d(g_i)$, the total number of its occurrences in the data repository, $A_t(g_i)$, the number of workflows that contain it, $A_w(g_i)$, and its compression power $A_c(g_i)$. For each indicator, normally the higher value the indicator has, the more selectable the candidate is. However, when the comparison is not clear cut across different indicators, the domain experts will have to make an informed decision based on all the indicators as well as their tacit knowledge about the macro candidate (*e.g.*, biological semantics, importance in science, expected future usage). As shown in Figure **??**B, the user interface displays each candidate with these indicators. The candidates are organized into columns, each representing a specific depth of the subgraphs in all macro candidates. Each candidate is shown with the basic pattern along with three indicators, A_t, A_w, A_c. The detailed structure of each macro candidate can be viewed on demand by using mouse interaction.

In order to help domain experts examine a large number of candidates speedily, we sort macros candidates in each column by a ranking score, allowing domain experts to inspect the most promising candidate first.

The ranking score is based on indicator A_t, A_w, and A_c.
Indicator 1: Occurrences in the data repository. Indicator A_t returns the total number of times a motif has occurred across the entire database of workflows. It emphasizes the importance in selecting motifs that are highly used, inferring their functional importance.
Indicator 2: Workflow presence. Indicator A_w returns the total number of workflows in which a motif has appeared. This provides a measure of how widely a motif is used across different biological experiments, counterbalancing the possible distortion in situations where a motif is heavily used in a relatively small number of workflows.
Indicator 3: Compression Potential. Indicator A_c is calculated by first subtracting the number of nodes in the motif (A_n) by 1, since these nodes will be replaced by a single macro node, and then multiplying by its occurrence (A_t). It is written as $A_c(g_i) = (A_n(g_i) - 1) * A_t(g_i)$.

For each of these three indicators, we map it to a fixed range $[-1,1]$ using a linear mapping based on the min-max range of each indicator, yielding three normalized metrics M_1, M_2 and M_3. These are combined into a single ranking score using a weighted average as:

$$S(g_i) = \sum_{k=1}^{3} \omega_k M_k(g_i)$$

where ω_k are three weights defined by users. Our system makes no assumptions about the merits of one indicator over another, so the default weights are set to one. We chose to have the score $S(g_i)$ in the range of $[-3,3]$, as it helps domain experts to connect the score back to the three indicators.

Figure **??** shows a screenshot of the user interface on the left (A), and two detailed views with annotation on the right (B, C). A domain expert normally examines macro candidates with the largest depth value (the rightmost column) first. Once a motif, $\mathscr{M}$, is selected, it may affect the current results returned by the indicators. As such a motif, $\mathscr{M}$, may contain many other candidate motifs as subgraphs. It is important for the domain expert to be aware of the impact of this decision on those candidate motifs yet to be examined. The system thus updates the indicators of all those candidate motifs included in $\mathscr{M}$. Instead of modifying A_t and A_w directly, the system shows a corrected score calculated by considering the new values for A_t, A_w and A_c, implying the difference if all $\mathscr{M}$ were to be removed from the repository.

4.5 Macro Design

Having obtained a collection of motifs suitable for use as macros within our corpus of workflows, we now have the task of designing their visual representation. It is important to keep the users in mind and ensure that the design reflects what a typical user, in our case a biologist, would expect to find in a macro. We consulted domain experts as to the visual elements that users considered the most important to view. A number of attributes were identified and are listed below in order of importance:

1. an impression of topology/structure within a macro (*e.g.*, it may be an entirely linear path, a set of parallel paths, or it may contain branch/merge events);
2. an impression of types of nodes in a macro (*e.g.*, the overarching theme of the macro);
3. textual description, (*i.e.*, additional annotation to provide concrete semantic meaning and help in understanding);
4. an impression of density within a macro, (*i.e.*, the size of the corresponding subgraph).

Given the attributes listed above, we devised three design options, as illustrated in Figure **??**, for creating pictograms. These pictograms are created automatically based on the states encountered when the motif is found by the motif generation algorithm in Section **??**. When the algorithm moves from one state to another, the pictogram grows by adding a new visual component reflecting the subgraph pattern just encountered. We provide three alternative designs for each macro. The first option is pixel-based, the second is shape-based and the third is a miniature version of the subgraph. All three design options are stored as vector graphics, so

Figure 4.7: From left to right: automatic generation of a micro pictogram based on the state transitions encountered. The lower three rows: Three design options for representing macros.

they are suitable for multi-scale display, for example, through zooming operations. Textual descriptions of the macros are always provided by experts.

Figure **??**(D) shows a pop-up window with a detailed view of a macro, and the three design options. The users can choose to have a fixed design for the macro, or have a multi-scale variation according to the level of detail at which a user is viewing the workflow.

Overview/Low detail. At the overview level, the fine-grained details of the workflow (*e.g.,* lines) utilize visual channels occupying a high spatial frequency. It is more effective for nodes in a graph to occupy low spatial frequencies, which will be distinguishable by a user. The pixel-based design option enables users to use visual channels that are visible and roughly distinguishable in low spatial frequencies. Although individual nodes and edges are not visible, the user can still gain a rough impression about the topology and node types.

Medium detail. At medium detail, users should be able to see more information through the shape-based design. The major steps from input to output (*i.e.*, following the state transitions) become more distinguishable. Each horizontal segment of the pictogram is colored by node type, branching and merging is shown with triangles of mirrored directionality, and heterogeneity of nodes is depicted in separated tracks of differing colors.

High detail. When a user zooms into a small region of the visualization, a miniature version of the subgraph becomes available, showing details of the topology and node types. This representation has a lot of high spatial frequency information and is only suitable for detailed examination in close up views.

4.6 Software Implementation and Use

To demonstrate our approach, we developed an open-source Java tool named *AutoMacron* that may be used in two modes: 1) standalone for those wishing to discover common motifs in a database of graphs; and 2) as an API for those wishing to integrate the utilities for motif discovery into their own software. We are also in the process of adding the capability to import formats other than ISA-Tab. The software provides the following functionalities:

1. Load files that have a handler (*e.g.,* ISA-Tab in our use case) into a graph database;
2. Analysis of all graphs in the database instance to determine the dominant motifs;
3. Allow for selection of macros from the pool of over-represented motifs found by the algorithm;
4. Visualize graphs both in uncompressed/compressed representations, and/or export those graphs in GraphML format for visualization in other environments;
5. Render differing images depending on the zoom level (semantic zooming [**?**, **?**]). For this, we have extended the Prefuse visualization library [**?**].
6. Allow for search of a graph database for a user-defined semantically-annotated motif;
7. Export macros for use in other software.

Using AutoMacron, over 12,000 valid motifs were found in a collection of 9,670 existing workflows from ArrayExpress. From that set, those motifs scoring below zero with the aforementioned aggregated score $S(g_i)$ were removed from consideration leaving just over 400 candidates.

Further examination of these candidates was conducted by domain experts using the macro selection utility shown in Figure **??**. Examination is aided through both presentation of the metrics and 'live' highlighting of motif representations as they occur in the original graph representation. Following the manual selection of 30 of these macros, the domain experts labeled each macro with a textual description, making the corresponding glyphs more meaningful and identifiable. These macros could then be used to substitute the more complex representations in the original graph in an effort to compress the representation.

Aside from the dedicated *AutoMacron* tool, the motif finding functionalities have been incorporated into ISAcreator, which is a tool used by domain scientists to annotate and curate biological experiment workflows and other necessary documentation. As shown in Figure **??**, the user is given the option to compress an experiment workflow using the macros selected by domain experts for biological data curation.

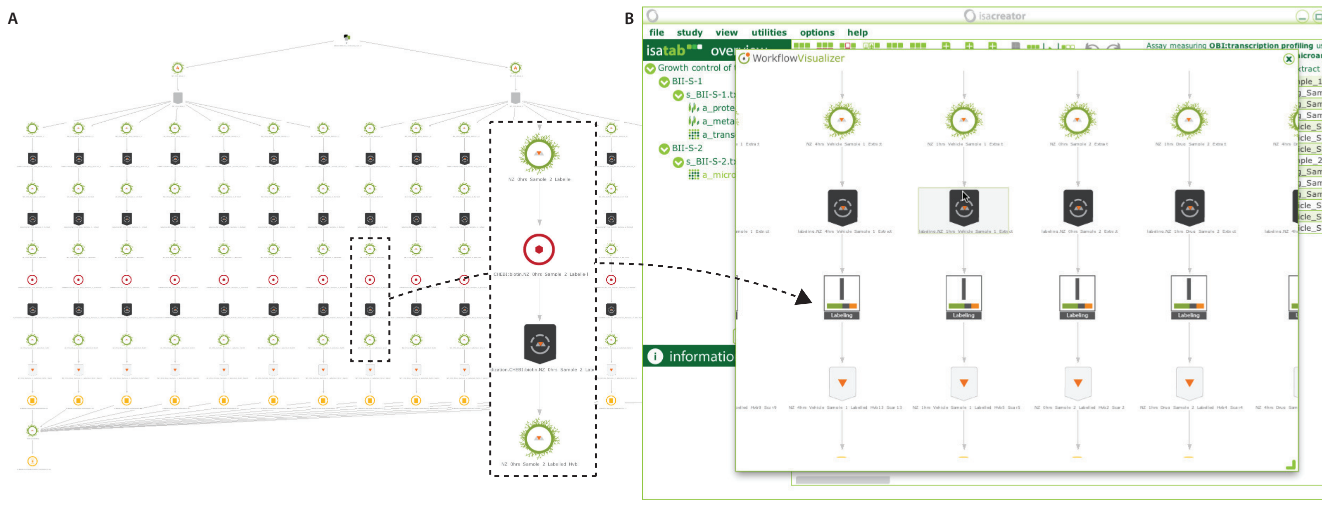


Figure 4.8: A) A typical workflow, where a 4-node motif was selected as a macro using AutoMacron, which provides ISAcreator with an API. (B) A screenshot of ISAcreator, where each occurrence of the original 4-node motif has been replaced with a macro glyph.

4.7 Evaluation

Aided by direct collaboration and regular interaction with domain experts, development of the software followed an evolutionary prototyping process whereby users evaluated the prototype at every major stage of the development. For each prototype users contributed their feedback to the software and algorithm outputs.

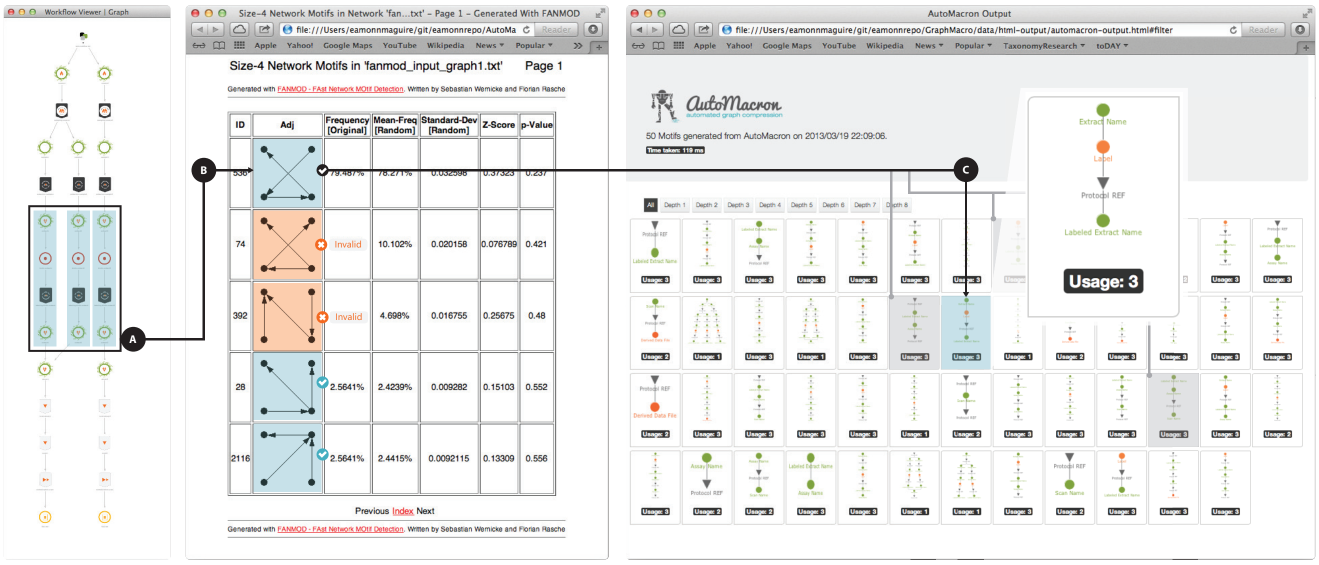
In this section, we summarize the feedback given from the last iteration where we performed the following: 1) analysis of the performance of the algorithm presented in this paper in comparison with that of the best existing (and available) algorithm; and 2) observation of how the software met the initial requirements as identified in Section **??** by interviewing expert users.

4.7.1 Evaluation Against Existing Algorithms

We wished to test the performance of the best in class in existing algorithms with that of the algorithm presented in this paper, not necessarily just in speed, but also in what was found and how that compared with domain experts' expectations. We compared the performance of FANMOD [**?**] and the algorithm presented in this paper in an attempt to discover which motifs could be found and how they related to the expectations of domain experts.

Firstly, we ran a simple test graph through FANMOD, to detect three, four, five, six, seven and eight node graphs. In FANMOD, there is a requirement to run each analysis separately, searching for size 8 node subgraphs does not yield all three to eight node subgraphs. Figure **??**B shows the output of a four node motif analysis and highlights FANMODs motif discovery match for a pattern highlighted in Figure **??**A. As aforementioned, there is no notion of node or edge type, therefore all four node graphs with the same serial topology will be the same, even though they represent entirely different concepts. Additionally, FANMOD, typical of the existing class of algorithms, returns invalid results with respect to our definition of a motif as defined in Section **??**.

Following this, we ran the same graph through AutoMacron to generate motifs up to depth 8. Note that FANMODs restriction is on node number (up to eight), whereas our algorithm can have potentially hundreds of nodes at depth eight. Figure **??**C shows the HTML output from AutoMacron's analysis. The highlighted motif corresponds to those highlighted in Figures. **??**A and B, we magnify the motif to show how our algorithm identifies the exact pattern with topology, node and edge type preserved.



Size-4 Network Motifs in 'fanmod_input_graph1.txt' Page 1

Generated with FANMOD - FAst Network MOtif Detection. Written by Sebastian Wernicke and Florian Rasche

ID	Adj	Frequency [Original]	Mean-Freq [Random]	Standard-Dev [Random]	Z-Score	p-Value
538		79.487%	78.271%	0.032598	0.37323	0.237
74		Invalid	10.102%	0.020158	0.076789	0.421
392		Invalid	4.698%	0.016755	0.25675	0.48
28		2.5641%	2.4239%	0.009282	0.15103	0.552
2116		2.5641%	2.4415%	0.0092115	0.13309	0.556

Previous Index Next

Generated with FANMOD - FAst Network MOtif Detection. Written by Sebastian Wernicke and Florian Rasche

Figure 4.9: A) A simple experimental graph visualized using the software from Maguire *et al.* [?]. In this example, we are showing a specific pattern and how that pattern is represented in both FANMOD and AutoMacron. B) HTML Output for 4-node motifs from FANMOD's analysis identifies 5 motifs with 4 nodes. 2 are incorrect (highlighted in orange) according to our rule set in Section **??**. C) HTML output from AutoMacron's analysis which identifies all 50 motifs. We have highlighted the specific motif being searched for (in blue) matching the highlight pattern in A and B, and show the other serial 4 node motifs (highlighted in grey) that would erroneously be considered the same had it not been for preserved semantics of nodes and edges.

Table 4.3: Results of motif identification by the domain expert, FANMOD and AutoMacron. Analysis included: **MIdent** - the ability of the domain expert to identify motif pictograms generated by the algorithms and match them to the original graph; and **UIdent** - the percentage of motifs found by the algorithm with respect to the number identified manually by the domain expert.

Source	Motif Id.	Valid Motifs	Acc	MIdent	UIdent	Time (ms)
Domain Expert	6	6	n/a	n/a	n/a	n/a
FANMOD	73	19	26%	4/19 (21%)	5/6 (83%)	1030*
AutoMacron	50	50	100%	49/50 (98%)	6/6 100%	119

Finally, we had the domain expert who inspected the graph manually and extract the motifs they would expect the algorithm to find. We compared what FANMOD and AutoMacron found with what the user expected. Our summary results are shown in table **??** with all analysis outputs available at https://bitbucket.org/eamonnmag/automacron-evaluation.

The results show that AutoMacron identifies less macros that FANMOD, however if one was to consider all the invalid motifs AutoMacron filters out due to incompatibility with our rule set (see Section **??**), AutoMacron would have reported many more. When we consider just the 'valid' motifs, AutoMacron has many more than FANMOD (fifty compared with nineteen for FANMOD). This is as a direct result of the semantics added by AutoMacron. To illustrate, consider a simple two node directed subgraph ($v \rightarrow v$) with 3 possible node types (n), AutoMacron could theoretically identify $n(n-1)$ different motifs whereas FANMOD and other algorithms already listed here could only identify one. Figure **??**B and C show for instance how one 4-node motif found in FANMOD maps to five potential, but only one correct motif in AutoMacron. Overall both algorithms identified the majority of motifs expected by the user, not unexpected considering both mechanisms are exhaustive, however FANMOD struggled when it came to identifying the larger motifs, due to the node limit of eight, hence its UIdent value of 83%.

The user was also asked to identify motifs based solely on the pictograms representing topology (macros) output from each program. In 98% of occasions, the user could identify AutoMacron motifs, aided by both color coding and better topological arrangement. Conversely, with FANMOD the user was able to decode 21% of the outputs.

4.7.2 Evaluation Against User Requirements

The algorithm and tool were tested by two domain experts to determine how the software has met the four requirements identified in Section **??**. For each requirement, we summarize their feedback below.

1. Given a number of workflows, can we obtain the most common patterns/motifs within those workflows?

   We tested AutoMacron on nearly 10,000 workflows, the software returned an abundance of motifs that were sorted by a score. The filtering tools helped us in finding motifs with specific topological events.'
2. For each motif, can we obtain the occurrence frequencies for that motif and information about which workflows they occur in?

   For each motif reported by the software, we were able to recover statistics about how often the pattern occurred as well, how many workflows the pattern appeared in and the names of these individual workflows.'
3. Can we automatically create macro representations of motifs with the added ability to add extra annotation?

   Each motif had three variations of 'macro' representations created automatically that could be used depending on the resolution available. We were able to add small pieces of text to these to help identification of the function of these macros, e.g. *labeling, extraction, or scanning.'*
4. Can motif patterns in a workflow be substituted by common macros to make the workflow more compact?

   The software provides a function to show compressed views of a workflow which automatically substitutes motif patterns with their macro representation. The function was also integrated to ISAcreator allowing us to serve out compressed representations of workflows to our users.

Further to this, users added that the software allowed them to identify erroneous annotations very quickly. For example, through inspecting the motifs, one domain expert discovered a prominent motif with nodes in the wrong position. This was only detectable via the algorithm's ability to maintain node type. On this matter, the domain expert commented "*Being able to detect such errors in annotation so quickly has enabled us to build scripts to fix those records and improve annotation quality. We can use this tool to find and fix inaccuracies in biological workflows much more quickly than we could before.*"

In many ways, the domain experts regard AutoMacron as a time-saving tool. Macro glyphs have a similar function to traffic signs. Domain experts normally expect certain macro glyphs in certain workflows. Although glyphs are small, they provide more assurance than observing detailed subgraphs directly because macros are computationally determined. It has helped them to construct the motif space computationally, which would otherwise takes years of effort. It has enabled them to explore the space of motifs efficiently with the aid of the ordered recommendations by the system. It has allowed them to create macros quickly without the need to design a pictogram for each macro. In the medium term, their everyday tasks, such as error checking, comparison, and identifying best practice, can be performed more speedily. In the long run, they also hope that this will bring benefits to the wider community; for instance, in experimental documentation and scientific publication.

While the discipline of biology has led the way in collecting workflows as part of data curation and sharing, we anticipate that some other disciplines will follow this trend soon. Our approach to workflow visualization in general and macro generation in particular can be adapted for other types of workflows if they have been curated.

4.8 Conclusions and Future Work

In this work, we introduced a new approach to macro creation aimed at reducing visual complexity in workflow visualizations. We developed a novel algorithm to discover motifs, with discrimination of node and edge type in a large collection of graphs. The algorithm was specifically designed for motifs in workflows and performed more efficiently than general-purpose motif finding algorithms.

We used a statistically-informed approach and an intuitive user interface to help domain experts in selecting macros from motif candidates. We devised a novel design method for automated creation of pictograms for macros by making use of the state transition information obtained by the motif discovery algorithm. We implemented our methods in a software system, available either through a dedicate graphical user interface (GUI) or through an API. Domain experts were able to use the system both to generate macros for compression of workflows and to find errors in a large corpus of existing workflows. Additionally, the selected macros and graph substitution algorithms are integrated into the ISA tools suite used by a large body of experimental biologists [**?**].

Our future work will cover two directions: the use of motif occurrence information to compare graphs based on ‘motif fingerprints’; and the use of macro-based standardized constructs in an experiment builder to simplify and improve the construction of workflows.

Chapter 5

Future Work

Chapter 6

Conclusion

List of Figures