Visual Compression of Workflow Visualizations with Automated Detection of Macro Motifs

Eamonn Maguire, *Student Member, IEEE*, Philippe Rocca-Serra, Susanna-Assunta Sansone, Jim Davies, and Min Chen, *Member, IEEE*

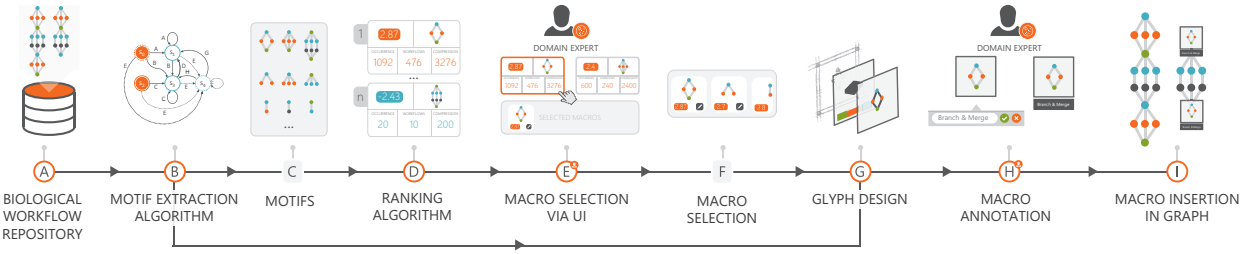


Fig. 1. An overview of the processes implemented in our system, AutoMacron, for compression of workflow visualizations. A) Biological workflows are imported from public repositories. B) The database of workflows is analyzed by our novel motif extraction algorithm that is sensitive to node/edge types and is built using a state-transition approach. C) The algorithm generates a large list of motifs. D) The motifs are ranked using metrics measuring their overall occurrence, scope of influence and compression potential. E) Based on the ordered recommendations by the system as well as their own knowledge, domain experts explore the space of motifs to identify suitable 'macros'. F) A subset of motifs are selected for macro generation. G) A glyph is assigned to each macro automatically using a design pattern that utilizes the state output from the motif extraction algorithm. H) Domain experts are able to annotate macros with additional text labels. I) Macros can be inserted into the relevant workflows as a method for graph compression.

Abstract—This paper is concerned with the creation of 'macros' in workflow visualization as a support tool to increase the efficiency of data curation tasks. We propose computation of candidate macros based on their usage in large collections of workflows in data repositories. We describe an efficient algorithm for extracting macro motifs from workflow graphs. We discovered that the state transition information, used to identify macro candidates, characterizes the structural pattern of the macro and can be harnessed as part of the visual design of the corresponding macro glyph. This facilitates partial automation and consistency in glyph design applicable to a large set of macro glyphs. We tested this approach against a repository of biological data holding some 9,670 workflows and found that the algorithmically generated candidate macros are in keeping with domain expert expectations.

Index Terms—Workflow visualization, motif detection, glyph-based visualization, glyph generation, state-transition-based algorithm

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" [4]. 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 [35], 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.* [24]. 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

- *Eamonn Maguire is with the Oxford e-Research Centre and Department of Computer Science, University of Oxford, UK. E-mail: eamonn.maguire@st-annes.ox.ac.uk.*
- *Philippe Rocca-Serra is with the Oxford e-Research Centre, University of Oxford, UK. E-mail: philippe.rocca-serra@oerc.ox.ac.uk.*
- *Susanna-Assunta Sansone and Min Chen are with the Oxford e-Research Centre, University of Oxford, UK. E-mail: susanna-assunta.sansone@oerc.ox.ac.uk.*
- *Jim Davies is with Department of Computer Science, University of Oxford, UK. E-mail: jim.davies@cs.ox.ac.uk.*
- *Min Chen is with the Oxford e-Research Centre, University of Oxford, UK. E-mail: min.chen@oerc.ox.ac.uk.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

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.

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 [10], VTK [34] and SmartLink [37] make use of workflows to depict the processes followed to create a visualization. In Taverna [16] and Kepler [7], 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 [1], new workflow visualization tools are emerging (*e.g.*, [24]).

Graph reduction is a family of algorithms and techniques for reducing visual complexity by using graph filtering and graph aggregation [39]. *Graph filtering* involves removal of certain nodes and edges from the graph, either deterministically or stochastically [22, 39]. *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 [39, 15].

One subset of graph reduction techniques is motif based. Motifs, in the context of graphs, are "patterns of interconnections occurring in complex networks" [25, 27]. 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.*, [18, 23, 41, 33, 38, 21, 12]). 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 [31] 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 [12]. Shneiderman and Aris propose the use of user-defined semantic substrates for compressing network visualization [36]. However, determining suitable macros

Table 1. Some commonly used motif finding algorithms.

Algorithm	Sampling	Node Limit	Focus
mfinder [19]	exact & estimated	6	discovery
FPF (mavisto) [33]	exact	4	search
FANMOD [41]	exact & estimated	8	discovery
NeMoFinder [11]	exact	12	search
MODA [26]	exact	not defined	search
G-Tries [28]	exact	> 9	discovery
Grochow-Kellis [13]	exact	9	search
Kavosh [17]	exact	8	discovery
Color-coding [5]	estimated	10	discovery

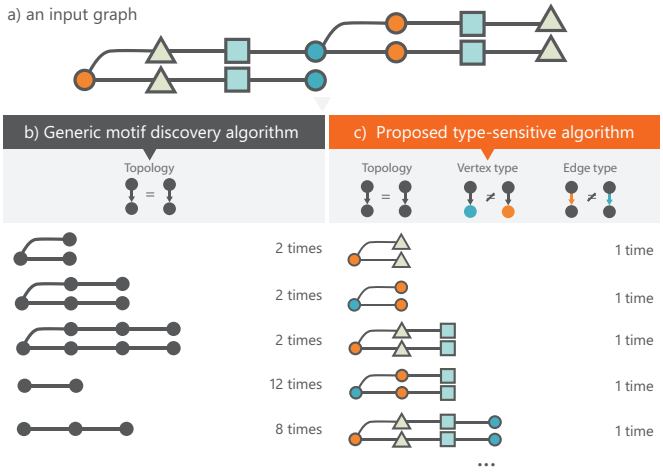


Fig. 2. (a) A graph, with typed nodes and edges, where different node types are mapped to different shapes and colours. (b) Generic motif discovery (or search) algorithms focus on topological differences. (c) Our motif extracting algorithm takes varying node/edge types into account, yielding semantically aware motif grouping.

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 1, many motif search/discovery algorithms exist. Ribeiro *et al.* [28], Kashani *et al.* [17] and Wong *et al.* [42] published benchmarks detailing processing time for a subset of the algorithms in Table 1. 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 [42].

In Table 1 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 2, 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 [32, 20] 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 2). 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 [9, 40].

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 [24] 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) [2]. 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 [29, 30], 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 [8]. For this purpose, we selected *Neo4j* [3] – 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).

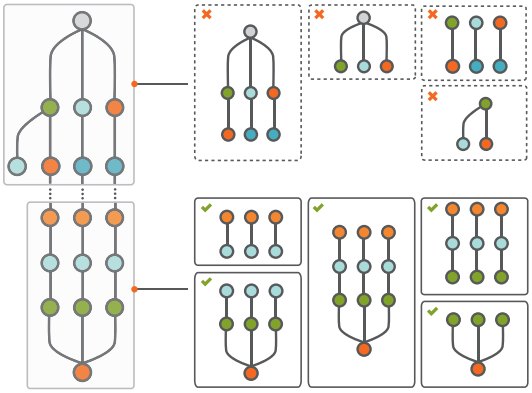


Fig. 3. In a workflow graph, some subgraphs can be considered as "legitimate motifs" while others cannot, depending on node type. Invalid motifs do not have the same output node type, as indicated by the differing colors. Conversely, valid motifs are those that have the same input and output node type.

Our system consists of three main functional steps, as depicted in Figure 1. (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.

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 From Motifs to Macros

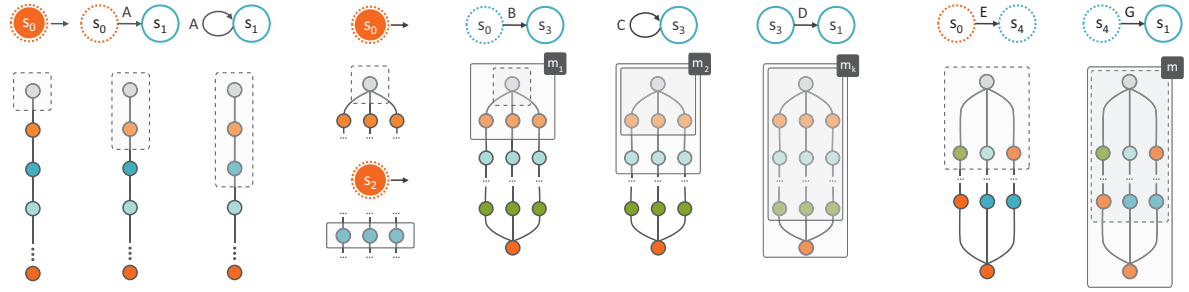
Motif discovery and substitution in graphs typically consists of four processes [25, 6]:

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 [25].

4.1 Candidate Macros in Workflows

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



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

Fig. 5. Examples of state transitions with different rules. Homogeneous edges are depicted by using the same color.

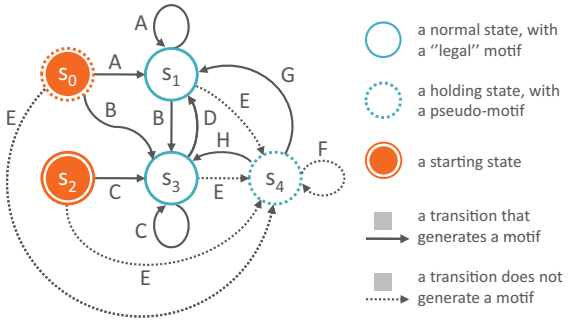


Fig. 4. The state transitions in motif generation.

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 3 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.

4.2 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 [42]. A specialized motif generation algorithm was needed, and for that we adapted the widely accepted 'pattern-growth' approach [42]. We describe the algorithm by evolving a state transition diagram as shown in Figure 4.

Singular Branching (Rule A). As shown in Figure 5(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 5(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 5(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 4.1).

When an edge first branches to multiple nodes, as illustrated in Figure 4, 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 5(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 4.1: 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

Table 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

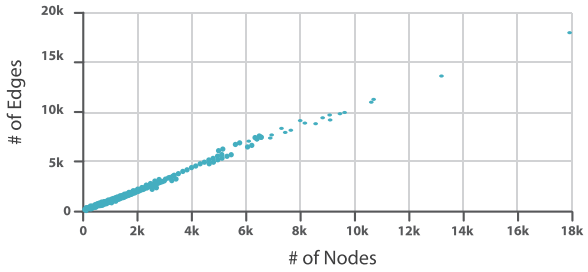


Fig. 6. Scatter plot showing the distribution of workflows (each depicted as a point) as node count versus edge count.

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 4.

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 5(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 6 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 (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 2 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 1, 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.

4.3 Selecting Macros from Candidate Macros

Given a list of motifs extracted using the algorithm in Section 4.2, 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 7B, 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

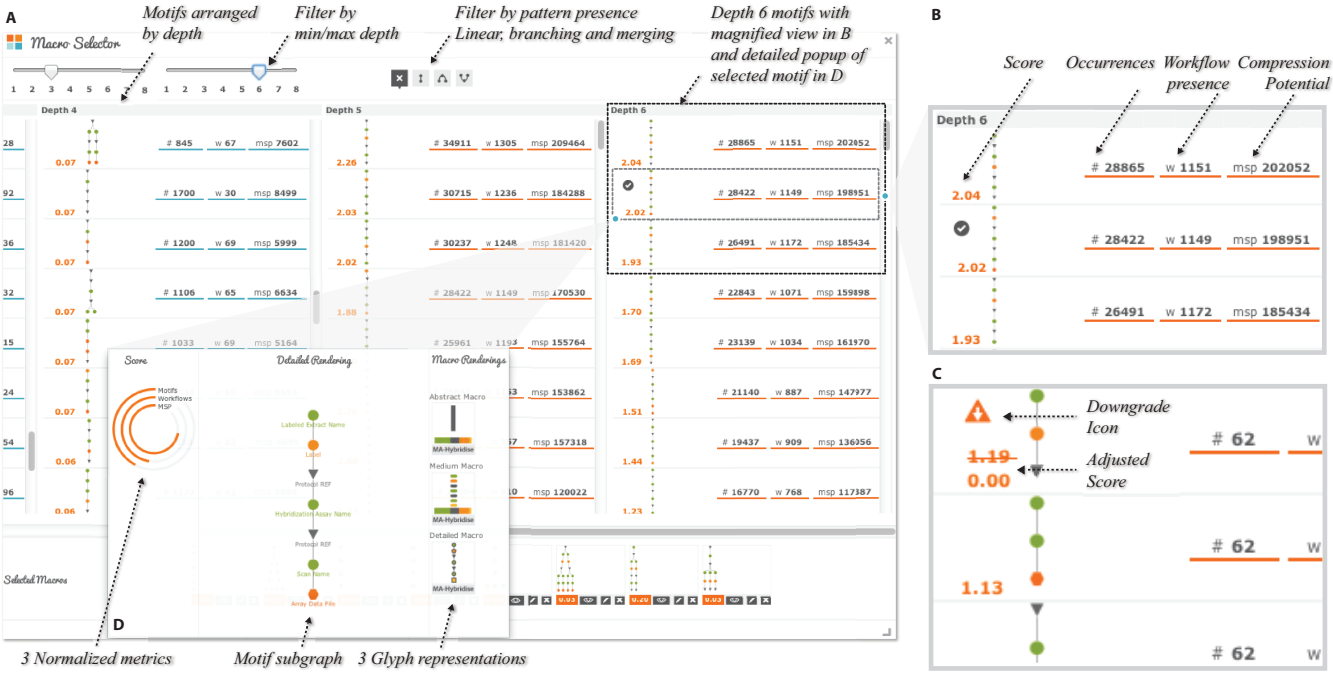


Fig. 7. AutoMacron provides a user interface (A) for domain experts to select macros from a list of computed motifs. As shown in the detailed view in (B), the overall score determines the order of motifs in the list. The three indicators are shown in unnormalized form in order to be semantically meaningful. 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.

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 7 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.

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 8, 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 4.2. 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 they are suitable for multi-scale display, for example, through zooming operations. Textual descriptions of the macros are always provided by experts.

Figure 7(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

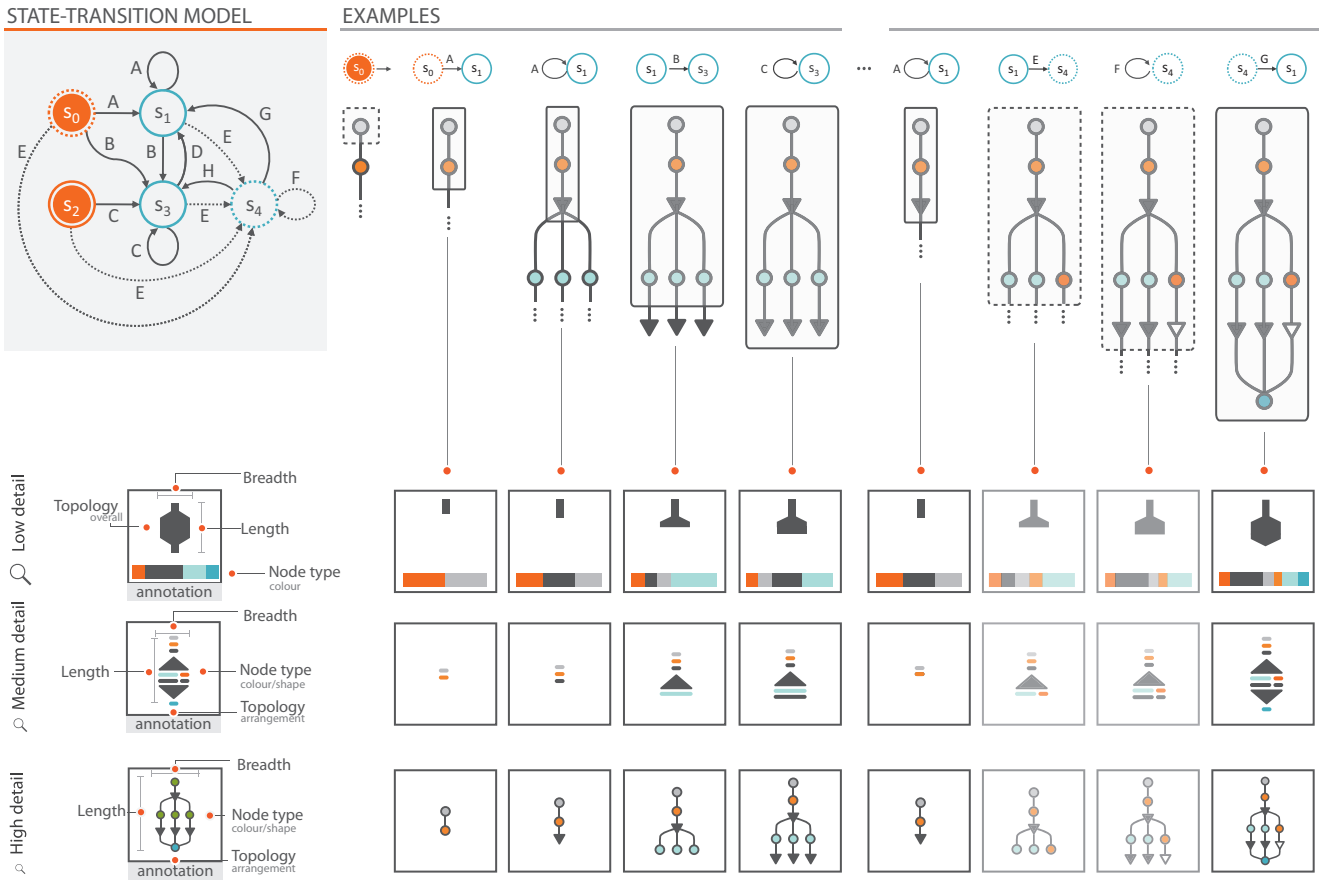


Fig. 8. From left to right: automatic generation of a micro pictogram based on the state transitions encountered. The lower three rows: Three design options for representing macros.

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.

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 [9, 40]). For this, we have extended the Prefuse visualization library [14].
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 7. 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 9, the user is given the option to compress an experiment workflow using the macros selected by domain experts for biological data curation.

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

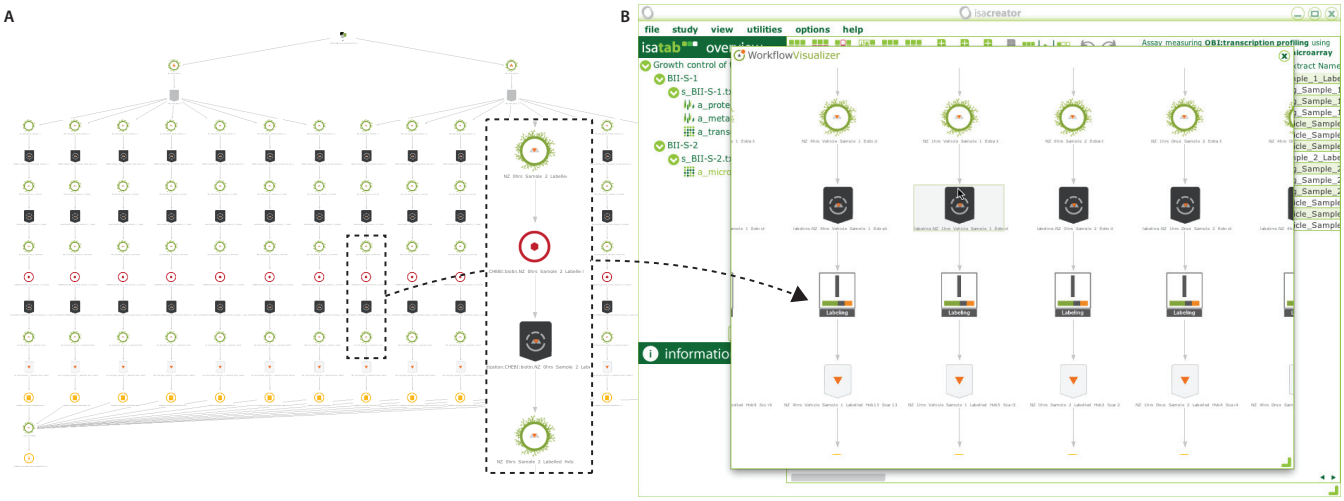


Fig. 9. A) A typical workflow, where a 4-node motif was selected as a macro using AutoMacron, which provides ISAcreator with an API. (B) A screenshot of ISAcreator, where each occurrence of the original 4-node motif has been replaced with a macro glyph.

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 3 by interviewing expert users.

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 [41] 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 10B shows the output of a four node motif analysis and highlights FANMODs motif discovery match for a pattern highlighted in Figure 10A. 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 4.1.

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 10C shows the HTML output from AutoMacron's analysis. The highlighted motif corresponds to those highlighted in Figures. 10A and B, we magnify the motif to show how our algorithm identifies the exact pattern with topology, node and edge type preserved.

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 3 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 4.1), 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,

Table 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

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 10B 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.

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 3. 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?

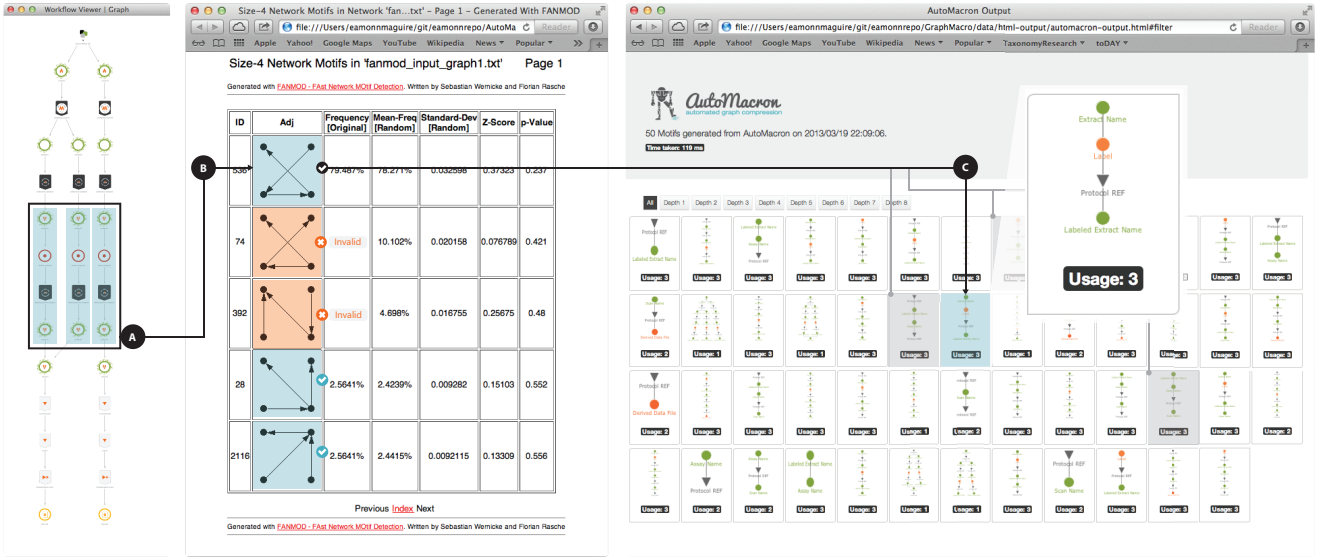


Fig. 10. A) A simple experimental graph visualized using the software from Maguire *et al.* [24]. In this example, we are showing a specific pattern and how that pattern is represented in both FANMOD and AutoMacron. B) HTML Output for 4-node motifs from FANMOD's analysis identifies 5 motifs with 4 nodes. 2 are incorrect (highlighted in orange) according to our rule set in Section 4.1. 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.

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.

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 [30].

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.

Acknowledgments

This work was supported by funding from the BBSRC and NERC, grants BB/I000917/1 and BB/I025840/1.

References

[1] Graphviz. http://graphviz.org/, 2012.
[2] Arrayexpress. http://www.ebi.ac.uk/arrayexpress/, Last Accessed: June 2013.
[3] Neo4j. http://www.neo4j.org, Last Accessed: June 2013.
[4] Oxford dictionaries, *Macro* definition. http://oxforddictionaries.com/definition/english/macro, Last Accessed: June 2013.

[5] N. Alon, P. Dao, I. Hajirasouliha, F. Hormozdiari, and S. Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics (Oxford, England)*, 24(13):241–249, 2008.

[6] U. Alon. Network motifs: theory and experimental approaches. *Nature reviews. Genetics*, 8(6):450–461, 2007.

[7] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. Kepler: an extensible system for design and execution of scientific workflows. *16th International Conference on Scientific and Statistical Database Management.*, pages 423–424, 2004.

[8] S. Batra and C. Tyagi. Comparative analysis of relational and graph databases. *Internation Journal of Soft Computing and Engineering (IJSCE)*, 2, 2012.

[9] B. B. Bederson and J. D. Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. *UIST '94 Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 17–26, 1994.

[10] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. VisTrails: visualization meets data management. *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747, 2006.

[11] J. Chen, W. Hsu, M. Lee, and S. Ng. NeMoFinder: dissecting genome-wide protein-protein interactions with meso-scale network motifs. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 106–115, 2006.

[12] C. Dunne and B. Shneiderman. Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. *CHI '13: Proc. SIGCHI Conference on Human Factors in Computing Systems*, 2013.

[13] J. Grochow and M. Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. *Proceedings of the 11th annual international conference on Research in computational molecular biology*, pages 92–9106, 2007.

[14] J. Heer, S. K. Card, and J. A. Landay. prefuse: A toolkit for interactive information visualization. In *Proceedings of ACM CHI*, pages 421–430, 2005.

[15] D. Holten. Hierarchical edge bundles: visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[16] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucleic acids research*, 34(Web Server issue):W729–W732, 2006.

[17] Z. Kashani, H. Ahrabian, E. Elahi, N. Abbas, E. Ansari, S. Asadi, S. Mohammadi, F. Schreiber, and M. Ali. Kavosh: a new algorithm for finding network motifs. *BMC Bioinformatics*, 10, 2009.

[18] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

[19] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Network motif detection tool mfinder tool guide. *Weizmann Institute of Science: Depts of Mol Cell Bio and Comp Sci & Applied Math, Rehovot, Israel (2002-2005)*, 2005.

[20] D. Koop, C. Scheidegger, S. Callahan, J. Freire, and C. Silva. VisComplete: automating suggestions for visualization pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1691–1698, 2008.

[21] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph*. *Data mining and knowledge discovery*, 11(3):243–271, 2005.

[22] J. Leskovec and C. Faloutsos. Sampling from large graphs. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2006.

[23] A. Ma'ayan, S. Jenkins, R. Webb, S. Berger, S. Purushothaman, N. Abul-Husn, J. Posner, T. Flores, and R. Iyengar. Snavi: Desktop application for analysis and visualization of large-scale signaling networks. *BMC Systems Biology*, 3, 2009.

[24] E. Maguire, P. Rocca-Serra, S.-A. Sansone, J. Davies, and M. Chen. Taxonomy-based glyph design — with a case study on visualizing workflows of biological experiments. *IEEE Transactions on Visualization and Computer Graphics*, 18(12), 2012.

[25] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science (New York, N.Y.)*, 298(5594):824–827, 2002.

[26] S. Omidi, F. Schreiber, and M. Ali. MODA: an efficient algorithm for network motif discovery in biological networks. *Genes & genetic systems*, 84(5):385–395, 2009.

[27] G. Pavlopoulos, M. Secrier, C. Moschopoulos, T. Soldatos, S. Kossida, J. Aerts, R. Schneider, and P. Bagos. Using graph theory to analyze biological networks. *BioData mining*, 4, 2011.

[28] P. Ribeiro and F. Silva. G-tries: an efficient data structure for discovering network motifs. *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1559–1566, 2010.

[29] P. Rocca-Serra, E. Maguire, S.-A. Sansone, and *et al*. ISA software suite: supporting standards-compliant experimental annotation and enabling curation at the community level. *Bioinformatics*, 26(18), 2010.

[30] S.-A. Sansone, P. Rocca-Serra, D. Field, E. Maguire, and *et al*. Toward interoperable bioscience data. *Nature genetics*, 44(2):121–6, 2012.

[31] E. Santos, L. Lins, J. Ahrens, J. Freire, and C. Silva. VisMashup: streamlining the creation of custom visualization applications. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1539–1546, 2009.

[32] C. Scheidegger, H. Vo, D. Koop, J. Freire, and C. Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560–1567, 2007.

[33] F. Schreiber and H. Schwöbbermeyer. Mavisto: a tool for the exploration of network motifs. *Bioinformatics*, 21(17):3572–3574, 2005.

[34] W. J. Schroeder, K. M. Martin, and W. E. Lorensen. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. *VIS '96 Proceedings of the 7th conference on Visualization '96*, pages 93–ff., 1996.

[35] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualization. *Proceedings IEEE Workshop Visual Languages*, pages 336–343, 1996.

[36] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):733–740, 2006.

[37] A. Telea and J. J. van Wijk. SMARTLINK: an agent for supporting dataflow application construction. *Springer*, pages 189–198, 2000.

[38] T. von Landesberger and M. Görner. A system for interactive visual analysis of large graphs using motifs in graph editing and aggregation. *Proceedings of Vision Modeling Visualization Workshop*, 2009.

[39] T. von Landesberger and A. Kuijper. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[40] C. Weaver. Building highly-coordinated visualizations in improvise. *IEEE Symposium on Information Visualization*, pages 159–166, 2004.

[41] S. Wernicke and F. Rasche. Fanmod: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 2006.

[42] E. Wong, B. Baur, S. Quader, and C. Huang. Biological network motif detection: principles and practice. *Briefings in Bioinformatics*, 13(2):202–215, 2012.