A Modular Degree-of-Interest Specification for the Visual Analysis of Large Dynamic Networks

James Abello, Steffen Hadlak, Heidrun Schumann, Hans-Jörg Schulz

Abstract—Large dynamic networks are targets of analysis in many fields. Tracking temporal changes at scale in these networks is challenging due in part to the fact that small changes can be missed or drowned-out by the rest of the network. For static networks, current approaches allow the identification of specific network elements within their context. However, in the case of dynamic networks, the user is left alone with finding salient local network elements and tracking them over time. In this work, we introduce a modular DoI specification to flexibly define what salient changes are and to assign them a measure of their importance in a time-varying setting. The specification takes into account neighborhood structure information, numerical attributes of nodes/edges, and their temporal evolution. A tailored visualization of the DoI specification complements our approach. Alongside a traditional node-link view of the dynamic network, it serves as an interface for the interactive definition of a DoI function. By using it to successively refine and investigate the captured details, it supports the analysis of dynamic networks from an initial view until pinpointing a user's analysis goal. We report on applying our approach to scientific co-authorship networks and give concrete results for the DBLP dataset.

Index Terms—Time-varying graphs, dynamic graph visualization, degree-of-interest.

1 Introduction

It is inherent in many time-varying data sets that a series of minuscule changes here and there accumulates to form the overall dynamics of the data. In a visualization of that data, it is important to be able to spot and track these small changes to form an understanding of the data dynamics. Yet, as the data grows larger, this becomes much harder, because these minor changes are either drowned out by overplotting from the rest of the data, or abstracted/clustered away along with other details to make room for a tidy overview. Providing both, overview and detail, in a meaningful way is a challenge in dynamic graph visualization – a challenge that becomes even harder, when the notion of what is overview and what is detail shifts in the course of the visual analysis. This paper addresses this challenge for the case of time-varying networks with hundreds of thousands of nodes and millions of edges.

Recent work by Farrugia and Quigley identifies two levels of analysis for dynamic graphs [14]: the *global network overview level* concerning changes influencing the graph as a whole and the *local individual node level* concerning smaller, more confined changes in the graph. To be able to support both levels – analyzing local changes, while at the same time maintaining an abstracted overview of the global network dynamics – one must first and foremost be able to differentiate one from the other. To realize this differentiation, we follow the idea of a *degree-of-interest (DoI) function* that quantifies the interestingness of each graph element at each point of time. Consequently, changes of elements with high DoI values are then tracked on an individual detailed level, whereas elements with low DoI values are only tracked in bulk on a global overview level. This raises the following three challenges:

- Devising a **comprehensive DoI definition** to make the necessary distinction between graph elements to follow on a local level and those to follow on a global level.
- Utilizing the DoI values in a **visual analysis setup** to make local and global changes distinguishable and permitting for an interactive refinement of this distinction.
- Showing that such a DoI-based approach indeed enables the user to form an understanding of the network dynamics in a **real world scenario**.

After a brief survey of the existing DoI-based graph visualization approaches in Section 2, the overall structure of this paper follows these three points and describes our contribution to each of them. Section 3 introduces a novel modular DoI definition that exceeds the existing DoI-based approaches for static graphs in two aspects. First, it can be utilized to hierarchically compose and recompose DoI functions on the fly, and second, it explicitly incorporates means to capture the dynamics of a network. Section 4 presents our visual analysis setup that shows the dynamic network according to the DoI values and that permits for interactive adjustment and steering of the DoI computation in all its aspects. This effectively breaks open the "black box" of DoI computation, as it not only enables a user to see whether a part of the network exhibits any dynamic feature of interest, but also to explore which feature(s) concretely lead to the display of a part of the network. Section 5 illustrates the utility and value of this DoI-based analytical process by taking a closer look at the top authors of the largest connected component of the DBLP data set [23] with its 914,492 nodes and 3,802,317 edges over 22 time points (years). This use case is also detailed in the video that accompanies this paper. It serves to give an impression of

- *James Abello is with the DIMACS Institute, Rutgers University. E-mail: abello@dimacs.rutgers.edu*
- *Steffen Hadlak, Heidrun Schumann, and Hans-Jörg Schulz are with the University of Rostock. E-mail: {hadlak,schumann,hjschulz}@informatik.uni-rostock.de*

how different DoI definitions are built, adjusted, and used to gain insight in the dynamics of this large network – something that would be hard to do with any other approach existing to date. Finally, Section 6 concludes our paper and details some ideas to be addressed in future work.

2 Related Work

One of the first DoI functions was specified by Furnas in 1986 for static hierarchies [17]. It already considered two aspects of the data: the user interest in each node (inverse to the distance to a currently selected focus node) and an a priori interest inherent in each node (inverse to its distance to the hierarchy's root). These two values are summed and then used to generate a contiguous subtree of high interest by pruning subtrees of lower interest. In this way, the limited screen real estate is used to convey only the subtree of highest interest. Furnas' overall idea is still the same until today and was merely extended in subsequent research to cope with

- other a priori interest values than the distance to the root,
- networks instead of trees,
- multiple user-selected focus nodes instead of a single one,
- and dynamic instead of static graphs.

Other a priori interest values than the rather specific distance to the root are regularly defined to capture interest according to certain application domains. Examples are the definition of a priori interest for nodes representing concepts in ontologies [20] and importance values for nodes corresponding to given search terms [36]. It is very common that a single DoI function combines multiple such a priori interest values, whose influence on the overall DoI value can be adjusted through adaptable weights. As a result, a user can no longer distinguish which of the a priori values triggered a node to receive a high combined DoI value. To communicate the cause for a node's high DoI value, most approaches simply use different colors [7], [8]. Furthermore, it has been observed that the combined DoI values may not necessarily be "well distributed", with lower values at the leaf nodes and higher values closer to the root node, as it would be needed for pruning [36]. To solve this problem, a priori interest values are often aggregated bottom-up, so that they are guaranteed to be monotonously increasing towards the root.

For networks, this requirement of strictly increasing interest values cannot be guaranteed as there is no singular (root) node towards which to aggregate them. To nevertheless counter the problem of heterogeneously distributed interest values, DoI-based approaches use, for example, a diffusion of DoI values across the network [36] to even them out. Script-based approaches can be used to tailor such a diffusion process, for example, to only diffuse the values to every other node, as it makes sense for bipartite networks [34]. The same problem of a missing singular reference node also affects the representation of networks that have been reduced according to a DoI function: For the pruning of subtrees, it was always apparent where details were removed (at the bottom), yet this is no longer the case for networks in which nodes may get removed or collapsed into metanodes in various places. In the latter case of a partially collapsed network, glyphs are often used to highlight metanodes among the uncontracted nodes and thus to inform a user where information is hidden from his view [18], [31]. In the former case of nodes having been removed or filtered, no metanodes exist in which to embed this information. In order to nevertheless indicate where subgraphs were removed, graph cues and signposts can be added to point towards invisible parts of the network and thus to provide a handle for navigating the abstracted context [26], [30], [36].

For multiple focus nodes, the challenge lies first and foremost in the computation of the distance between each node and the selected foci in order to determine each node's DoI value. While this is straightforward for a single focus, there is no standard way to compute a single distance value to multiple points in a network – regardless of whether the geodesic distance is used or any other distance metric, such as a geometric distance [20] or a similarity metric [8]. In either case, the most common approaches are to use the distance to the closest focal node [18] or to use the average distance to all focal nodes [26]. To reduce the number of distance computations, the number of focal nodes is often bound by a maximum quantity, which is realized by decreasing the influence of all existing foci whenever a new focus is selected [9], [18], [20]. For their depiction, networks with multiple focus nodes can no longer simply be laid out by using the geodesic distance to the focus, as it is done, for example, in [32]. Instead two other approaches prevail: those that aim to assign them prominent positions through an adapted force-based layout [29] and those that simply use color or label size to highlight them [9], [20] in a regular layout.

For dynamic graphs, DoI values are computed for each time point individually. Here, the same problem can occur as when generalizing to networks: the DoI values of a node can vary a lot from one time point to the next. In case of an animated view over time, this can lead to pop-up artifacts in the visualization when nodes are of high interest at one time point and of low interest at the next time point. The first DoI-based approach for dynamic trees did not directly counter this problem, but instead put all focus nodes into a set of high interest nodes that remained the same for all time points [7]. This way, pop-up artifacts will still occur, but not for focus nodes, as these will always be of high interest and thus visible. In [31], a different approach was presented for dynamic hierarchical compound graphs. It uses a temporal relaxation to distribute DoI values from and to previous and future time steps to limit such visual effects. After treating sudden changes with either method, both approaches rely on animations to transition smoothly from one time point to the next. To navigate through time, [7] utilizes the DoI values as part of a complex time slider that denotes time points at which specific events have occurred for a set of focus nodes.

In summary, it can be observed that the many existing DoI-based visualization approaches are all provided as final, fixed monolithic solutions that are tailored to a specific application domain with specific tasks and specific types of graphs. Possibilities to adjust them rarely go beyond the adaptation of a few weights for their individual terms. While this simplification prevents the user from being exposed to the full complexity

of a comprehensively defined DoI function, it also limits his ability to express his own interest for a particular task or a particular input graph that was not anticipated by the visualization designers.

Other fields have already gone a step further in this regard. For example, for high-dimensional numerical data, more flexible DoI approaches exist, such as the Feature Definition Language (FDL) by Doleisch et al. [10]. It allows for specifying basic features of interest and to combine them into more complex features. These features can even include temporal measures, such as derivatives of attributes [11]. In order to communicate and modify structure and parameters of an FDL-defined complex feature, a hierarchical representation is used [10], [11]. Yet, due to its specificity to numerical input data, it cannot be applied to graphs as it does not provide means to take the graph structure into account. In particular for large dynamic graphs, no approach with comparable flexibility has yet been proposed to the best of our knowledge. This is the gap, which we aim to address in the following section with our modular DoI specification for dynamic networks.

3 A Modular DoI Definition for Dynamic Networks

The aim of this section is to introduce a DoI definition that is able to cope with the changing demands of interactive visual analysis of dynamic networks from various domains. We achieve this by bringing two novel ideas to the field of DoI functions for graphs. The first is a modular way of assembling DoI functions from predefined functional components. This way, a DoI function can be flexibly adapted to new application domains or newly found network characteristics by simply exchanging or reconfiguring its individual components. These components can be grouped into four categories, which we named depending on the practical role they play in the overall DoI definition:

- *DoI generators* provide the base DoI values that specify the user interest in individual characteristics of graph elements. The corresponding function blocks are called **specification components**.
- *Unary DoI operators* transform a single given DoI value – e.g., to amplify or reduce it. The corresponding function blocks are thus called **transformation components**.
- *N-ary DoI operators (1 graph element, n DoI values)* combine two or more DoI values defined for a single graph element. The corresponding function blocks are therefore called **combination components**.
- *N-ary DoI operators (n graph elements, 1 DoI value)* propagate a DoI value across neighboring graph elements. The corresponding function blocks are therefore called **propagation components**.

The second novel idea is the incorporation of the temporal aspect in the DoI definition. This is achieved by providing specific components that allow the user on the one hand to define something as interesting, because of its temporal dynamics, and on the other hand to define something as interesting, because it was/will be interesting in past/future time steps. The former is embodied in particular specification components, the latter is made available through a temporal propagation component – both are novel in the domain of DoI functions for networks.

The following section briefly defines the basic terms and the functional formalism that permits us to plug these components into each other. The four types of components themselves are then described in the sections thereafter, before they are finally brought together in an example of how to express Furnas' original DoI function.

3.1 Basic Terms

We consider a totally ordered set T of time points t_i with $1 \leq i \leq |T|$. These time points do not have to be equidistant, i.e., the duration between two time points $t_{i+1} - t_i$ may vary. In this context, a dynamic network G is a sequence of graphs $G_i = (V_i, Av_i, E_i, Ae_i, t_i)$ forming a 5-tuple of a node set V_i, a set of node attributes Av_i, an edge set E_i, and a set of edge attributes Ae_i at the time point $t_i \in T$. We intentionally refrain from using the term *edge weights*, so that numerical attributes of both, nodes and edges, are consistently called as such. This makes sense, as in most of the following discussions, we do not distinguish between nodes $v_i \in V_i$ and edges $e_i \in E_i$, but instead we speak of *graph elements* $x_i \in V_i \cup E_i$. Hence, node and edge attributes both map a given graph element to a numerical attribute $attr(x_i) : V_i \cup E_i \mapsto \mathbb{R}$ with $attr(x_i) \in Av_i \cup Ae_i$. In case that an attribute is only partially defined, i.e., only for nodes or only for edges, it maps all other elements to `undefined`. Common examples for node attributes are the degree $deg(v_i)$ or the cluster coefficient $cc(v_i)$, whereas for edge attributes an often used example would be the edge betweenness centrality $bc(e_i)$. In accordance with established DoI-based graph visualization approaches, we define a DoI value as a real number in the interval $[0 \ldots 1]$, with 0 expressing *no interest* and 1 expressing the *highest degree of interest*.

3.2 Specification Components

Overall, the specification $spec(x_i)$ captures the user interest in a graph element $x_i \in V_i \cup E_i$ at time point $t_i \in T$ based on some property of that element, for example, an attribute $attr(x_i)$. It consists of two functions: a computation function $comp : V_i \cup E_i \mapsto \mathbb{R}$ that calculates a single numerical value for a graph element and the interest function $inter : \mathbb{R} \mapsto [0 \ldots 1]$ that maps this value to the interval $[0 \ldots 1]$ in accordance to the user interest. Hence, any specification component can generally be expressed in the form $spec(x_i) = inter(comp(x_i))$. This is noteworthy, as most existing DoI approaches use computed node/edge properties directly in their DoI function, without explicitly defining the user interest over each property's value range. We decouple these two notions to help the user distinguish between *what* influences his interest in a graph element (computation function) from *how* it influences it (interest function).

3.2.1 The Computation Function

The computation function $comp$ evaluates structural properties and attribute values for a given graph element $x_i \in V_i \cup E_i$ and computes a single numerical value to express them.

TABLE 1: Categorization of basic sources for specifying interest.

Domain \ Time Points	current (one time point)	immediate changes (two consecutive time points)	higher order changes (three or more consecutive time points)
structure (node / edge)	presence / absence	addition / removal	age, frequency,...
values (attribute / weight)	identity, normalization,...	change, rate of change,...	moving average, trend,...

As it is evaluated before the interest function is applied, it allows for capturing network dynamics that span multiple time points, deriving a singular value from them, before defining one's interest on top of such derived values. Depending on the number of time points that *comp* takes into account, we differentiate three cases that are shown in Table 1 and discussed in the following.

Current / One time point: This first case is usually used for static graphs, for which there are no time points to consider. Structurally, this means to specify a DoI in terms of the presence or absence of graph elements (usually edges). In terms of a graph element's numerical attributes, the computation function is usually an identity function or performs at most a normalization of these values. The latter is frequently used in order to be able to apply the same DoI function across different time points with potentially very different absolute node/edge counts, as well as across differently sized input graphs. Overall, this case can be expressed for a given attribute *attr* as $comp(x_i) : attr(x_i) \mapsto \mathbb{R}$.

Immediate changes / Two consecutive time points: This case contains all computation functions, which are defined on two subsequent time points. Structurally, this can be either an addition of a graph element from the last time point t_{i-1} to the current time point t_i, or a removal that will occur at the next time point t_{i+1}. Changes of a numerical attribute are determined in a straightforward manner by, for example, computing the difference between the two values or the rate of change between them. Again, depending on whether to compute the change in hindsight or foresight, this can be expressed for a given attribute *attr* as

$$comp(x_i) : attr(x_{i-1}) \times attr(x_i) \mapsto \mathbb{R} \quad \text{(hindsight)}$$
$$comp(x_i) : attr(x_i) \times attr(x_{i+1}) \mapsto \mathbb{R} \quad \text{(foresight)}$$

Higher order changes / Three or more consecutive time points: This case subsumes all computation functions, which compute a development of the graph over more than two consecutive time points. With respect to the graph structure, this can be, for example, the age of a graph element, i.e., for how long it is already present, or any other measure over multiple time points, such as the frequency with which an element appears [1]. Whereas the development of attribute values and weights of a graph can be assessed in the manner known from time series data, e.g., by computing moving averages over multiple time points or trends. For a given attribute *attr*, as well as a number p of past time points and a number f of future time points to include, this case can be expressed in general as

$$comp(x_i) : attr(x_{i-p}) \times \ldots \times attr(x_i) \times \ldots \times attr(x_{i+f}) \mapsto \mathbb{R}.$$

3.2.2 The Interest Function

The interest function *inter* defined on top of the computation *comp* realizes the mapping from real values to DoI values. It can be envisioned as "carving" the interesting parts from the value range of a computed numerical property. This includes, that it also maps any `undefined` attributes to 0 to express no interest in them. While the interest function can take on any required shape or form, there are a number of commonly used ones that are briefly listed in the following, where x stands for any previously computed numerical value $comp(x_i)$.

Gaussian function: $inter(x) = e^{-(x-\alpha)^2/\beta}$
This function is used to pinpoint a certain value of interest and produce a smooth decline of interest with increasing distance to this value. The value of interest can be set through the parameter α, the gradient of its decline is governed by the parameter β with $\beta > 0$.

2-Sided exponential function: $inter(x) = \beta^{|x-\alpha|}$
This function is also used to focus on a specific value, but with a steep decline towards smaller values. The value itself can be set via the parameter α, while the rate of the decline is governed by the parameter β with $0 < \beta < 1$.

Sigmoid function: $inter(x) = 1/\left(1 + e^{-\beta*(x-\alpha)}\right)$
This function is used to express interest in only high values with a smooth decline at a certain threshold. The threshold can be set by the parameter α, whereas the rate of the decline is governed by the parameter β with $\beta > 0$.

Piecewise constant function: This function is used to define intervals of different interest levels. There is no fixed set of parameters for its specification, as the number of intervals may vary.

3.2.3 Utilizing a Combination of Computation and Interest Functions to Capture Interactive Selections

So far, we have only discussed instances in which we wanted to capture properties of the network data and its dynamics. Yet, our two-part DoI specification is much more versatile than that and it is even able to express such aspects as the interactive selection of graph elements by encoding the selection logic in the computation function and an interest in more recently selected elements in the interest function.

For a simple selection, it needs merely a running global counter *count* that is increased by 1 each time a selection is made and an attribute $clicked\_at(x_i)$ to store the current *count* when element x_i gets selected. The computation function can

then be used, for example, to limit the selected elements to only those of the n last selection operations

$$last_n(x_i) = \begin{cases} clicked\_at(x_i) > count - n & : \quad clicked\_at(x_i) \\ else & : \quad 0 \end{cases}$$

The interest function defined on top of that computation can be, for example, a decay function similar to [9]:

$$decay_d(last_n(x_i)) = \begin{cases} last_n(x_i) > 0 & : \quad d^{count - last_n(x_i)} \\ else & : \quad 0 \end{cases}$$

Depending on the decay factor d, the resulting DoI values can range from all 1's for $d = 1$ (multiple foci if $n > 1$) to only the last selected element having a DoI value of 1 for $d = 0$ (single focus). A decay factor between 0 and 1 can be used to balance these two approaches by fading-out older selections and thus ensuring that only a limited number of elements is selected at all times. In the remainder of this paper, we use the term $select_d(x_i)$ as a placeholder for any such DoI specification $decay_d(last_n(x_i))$ that captures an interactive selection with decay factor d over the last n graph elements that were clicked.

Every specification component $spec(x_i)$ is in itself already a very simple DoI function $doi(x_i)$. The remaining components merely take DoI functions – such simple ones as $spec(x_i)$, but also more complex ones as they result from applying the following components – as an input to yield more powerful DoI functions.

3.3 Transformation Components

Functional components falling in this category are mainly used to modify a previously specified DoI function by emphasizing certain parts, cutting off others, or simply inverting it. Thus, a transformation component can be realized through any function, which fulfills $trans : [0 \dots 1] \mapsto [0 \dots 1]$ The two transformations that are mainly used are an **inversion** and a **scaling** of a DoI.

Inversion function: $inv(doi(x_i)) = 1 - doi(x_i)$
Inverting a DoI function is suitable, if the opposite of the user interest can be specified more easily than the actual interest. For example, when the user is not interested in all values above a certain threshold, as expressed with the sigmoid transfer function, but instead in all values below that threshold, then the inversion of the sigmoid can be used to express this.

Scaling function: The scaling of DoI values is suitable, for example, to lessen their influence or to ensure a guaranteed minimal influence, even if the values are low. The two forms that are often used, the multiplication and the exponentiation, can be expressed in one function $scale_{const,exp}(doi(x_i)) = const * doi(x_i)^{exp}$ with $const \in [0 \dots 1]$ and $exp \in \mathbb{R}$.

3.4 Combination Components

Since a DoI definition is usually based on multiple attributes, properties, dynamics, etc., combination components can be used to form more complex patterns from multiple DoI definitions that capture different features. Such patterns can even have different temporal dependencies, i.e., a pattern defining a linear function can be combined from a value at some time point (first column of Table 1) and a slope (second column of Table 1). Combinations thus realize a mapping $comb : [0 \dots 1]^n \mapsto [0 \dots 1]$. Numerous useful combination functions are known [3]. However, the most common ways to incorporate multiple DoI terms in one function are either **Min/Max combinations** (often used in DoI functions for multivariate data, such as the FDL) or **weighted sums** (often used in DoI functions for network data).

Min/Max combinations: The minimum combination is used to express that all input functions must return a high DoI value, in order for the combined function to also return a high DoI value. In this regard, it can be viewed as the fuzzy-logical AND operator on the input functions. The maximum combination works the other way around by returning a high DoI value if at least one of the input functions has a high value. It is thus comparable to the fuzzy-logical OR operator.

Weighted sum: This type of combination aims to balance the influence of the n individual DoI terms by multiplying them with weights $w_1, \dots, w_n \in \mathbb{R}$ before simply adding them up. To ensure that the resulting function value will still be within the range of $[0 \dots 1]$, the weighted sum is divided by the sum of all weights. Depending on how the weights are chosen, weighted sums can be used to express a variety of different operations, such as averages ($w_k = 1/n$) or the maximum likelihood estimator ($w_k = 1/\sigma_k^2$), both with $k \in [1 \dots n]$.

3.5 Propagation Components

The DoI functions defined so far are always bound to one specific graph element x_i at one specific time point t_i. The region around them – structure-wise, as well as time-wise – is not yet considered. To realize our goal of not only pinpointing elements of interest, but also showing them in their immediate context, we use propagation functions to disseminate high DoI values to surrounding graph elements and time points. Consequently, we discern between the propagation of DoI values across the graph topology and along the temporal axis.

3.5.1 Structural DoI Propagation

A structural propagation distributes DoI values across multiple graph elements for each time point individually. It is basically determined by four functions:

- An **input DoI function** $doi(x_i)$ whose values are to be gathered for a graph element x_i.
- A **distance function** $dist_{attr}(x_i, y_i) : (V_i \cup E_i)^2 \mapsto \mathbb{R}$ that takes two graph elements from the same time point t_i as an input and measures w.r.t. the edge attribute $attr$ how far they are separated from each other as a real value.
- An **edge attribute** $attr(x_i)$ to be used for computing the distance function, with only positive edge attributes being allowed: $attr(x_i) : E_i \mapsto \mathbb{R}^+$. If the used edge attributes represent capacities, where large values actually stand for a tighter connection than small values do, they have to be transformed into costs. This is simply done by computing $attr' = 1/(attr + 1)$.

- A **drop-off function** $drop(doi(x_i), dist_{attr}(x_i,y_i)) : [0\dots1]\times\mathbb{R}\mapsto[0\dots1]$ that is monotonic decreasing, so that at larger distances the DoI value recedes.

These functions are then combined with a maximum to preserve the highest DoI value, so that no element of high interest will be drowned out by a neighborhood of lesser interest:

$$prop_s(doi(y_i),x_i) = \underset{y_i\in V_i\cup E_i}{MAX}\, drop(doi(y_i), dist_{attr}(x_i,y_i))$$

For its computation, different distance functions are possible. Since we do not distinguish between nodes and edges and want to diffuse DoI values defined for nodes also to their incident edges and vice versa, we need a generalized distance function, which permits us to do so. For this, we define the distance between two nodes d_{vv} as the usual geodesic distance $gd(x_i,y_i)$ between them. The distance between an edge and a node d_{ve}/d_{ev} is computed as the minimum distance between the two incident nodes to that edge and the node plus half of the edge attribute, which mimics a node lying halfway on the edge. And the distance between two edges d_{ee} is reduced to the former case. All in all, the distance function is defined as:

$$dist_{attr}(x_i,y_i) = \begin{cases} x_i\in V_i\wedge y_i\in V_i & : & d_{vv}(x_i,y_i)\\ x_i\in V_i\wedge y_i=(v_1,v_2)\in E_i & : & d_{ve}(x_i,y_i)\\ x_i=(v_1,v_2)\in E_i\wedge y_i\in V_i & : & d_{ev}(x_i,y_i)\\ x_i=(v_1,v_2)\in E_i\wedge y_i\in E_i & : & d_{ee}(x_i,y_i)\end{cases}$$

$$d_{vv}(x_i,y_i) = gd(x_i,y_i)$$
$$d_{ve}(x_i,y_i) = MIN(d_{vv}(x_i,v_1),d_{vv}(x_i,v_2)) + attr(y_i)/2$$
$$d_{ev}(x_i,y_i) = MIN(d_{vv}(v_1,y_i),d_{vv}(v_2,y_i)) + attr(x_i)/2$$
$$d_{ee}(x_i,y_i) = MIN(d_{ve}(v_1,y_i),d_{ve}(v_2,y_i)) + attr(x_i)/2$$

3.5.2 Temporal DoI Propagation

A temporal propagation distributes DoI values across multiple time points for each graph element individually. It consists of an input DoI function $doi(x_i)$, but x_i's DoI values are in this case to be gathered from all time points t_j with $1\le j\le|T|$ and scaled w.r.t. t_i using the given drop-off function:

$$prop_t(doi(x_j),x_i) = \underset{j=1}{\overset{|T|}{MAX}}\, drop(doi(x_j), t_j - t_i)$$

The distance between two time points is simply their difference. If it is negative, the time point under consideration lies before t_i – otherwise after it. This way, the drop-off function can be defined differently for negative values than for positive values, allowing the user to adjust the spreading of DoI values independently. Thus a foreshadowing of the occurrence of an interesting element can be handled in another way than its influence in retrospect. As a result, the propagation across time prevents sudden jumps in the DoI values and smoothes any process that relies on these values, i.e., prevents pop-up artifacts in animations. But, as the example in the following section shows, propagations can also be used for other purposes.

3.6 Example: Realizing Furnas' DoI Function

Since our approach combines many features of existing DoI functions, it is not surprising that most of them can be expressed by a combination of the functional components described above. To illustrate the overall idea of how this is done, this section shows how to formulate Furnas' original DoI function using our approach.

Furnas' DoI function has two parts: the a priori importance $API(x) = -dist(x,root)$ and the distance $D(x,y) = -dist(x,y)$ of a node x to a focus node y, which are summed to yield:

$$DOI(x|y) = -dist(x,root) - dist(x,y)$$

The value range of this function spans $[-3*max\_height\dots0]$ where *max_height* denotes the maximum height of the tree in question. Consequently, a DoI value of $-3*max\_height$ represents the lowest interest and is produced when x and y are both leaves at depth *max_height* and *root* is their least common ancestor. On the other end of the spectrum, a DoI value of 0 stands for the highest interest, which results for $x=y=root$. This kind of an open ended DoI value range that depends on the size of the input graph and that can be observed for many existing DoI functions is an aspect that we cannot model with our fixed $[0\dots1]$ interval of DoI values.

While the pros and cons of open or closed value ranges are debatable, a fixed closed interval is an absolute necessity in our case in order to make the different functional components compatible with each other and thus to permit for their flexible combination. Yet, even though we cannot produce the very same DoI values of Furnas' DoI function, we can nevertheless capture its functionality in the range of our $[0\dots1]$ interval. Note that the index i has been dropped from the following DoI definition, as Furnas' DoI function does not deal with dynamics in the tree.

$$doi(x) = 1/3*inter(height(x)) + 2/3*prop_s(select_0(y),x)$$
$$inter(height(x)) = 1 - height(x)/max\_height$$
$$dist_{attr}(x,y) = gd(x,y) \text{ with } attr(x) = 1$$
$$drop(select_0(y), dist_1(x,y)) = 1 - dist_1(x,y)/(2*max\_height)$$

The overall DoI function is a weighted sum of two individual DoI functions: one defined over the height of a node x in the tree and another one defined over the selection status of a node y. The weights make the contribution of each function to the overall DoI value explicit and thus also adjustable if necessary. They can be observed in Furnas' DoI function, for example, in the case of the minimum DoI value of $-3*max\_height$ to which the a priori interest contributes $-max\_height$ (the maximum distance to the root) and the distance to the focus node contributes $-2*max\_height$ (the maximum distance between two nodes).

The first function is a straightforward encoding of Furnas' a priori interest, with $height(x) = dist(x,root)$ being the computation function and the given interest function $inter(height(x))$ mapping it inverse linearly on the interval $[0\dots1]$. The latter ensures that the root at $height = 0$ receives the maximum DoI value of 1 and the deepest leaves receive the minimum DoI value of 0.

The second function $select_0(y)$ captures the selection status of a node y according to its specification in Section 3.2.3. The selection status of each node (either 0 or 1) is then propagated to all other nodes using the common geodesic distance $gd(x,y)$

with all edge attributes set to a constant 1 and a linear drop-off function that assigns 1 if the current node itself is selected and 0 if the current node is at the largest distance to the selected node. This procedure is necessary, as we do not know the interactively selected focus node "a priori". Thus, we cannot precompute all distances to it and store it as a node attribute, as it was possible for the height. Instead, the structural propagation is used to automatically readjust the DoI values every time the selection status changes.

As a result, the overall DoI function behaves similar to Furnas' DoI function and yields the maximum DoI value of 1 for the root node, if it is also the focus node, and 0 for a leaf at the deepest level and furthest away from the focus node. Yet, in contrast to Furnas' DoI function, we can change its behavior very easily, now that all its components are exposed. For example, to turn the DoI function into one that handles multiple foci in the way that DOITrees [18] do, one can simply change $select_0(y)$ to $select_1(y)$. This is possible, because the closest focus node that has been least affected by the drop-off function and thus has the highest DoI value will be the natural result of the maximum combination that realizes the structural propagation.

As this example shows, it is not only possible to build one's own DoI function with the proposed components, but also to reproduce existing DoI functions with them. This provides endless combination and adjustment opportunities, which on the one hand are absolutely necessary to express the user interest as precisely as possible to form a meaningful functional base for the subsequent visualization. On the other hand, all these different options also create additional complexity that the user must master in order to make fullest use of our approach. The following section discusses these points, illustrating a realization of a visual analysis setup built around our modular DoI definition concept.

4 An Interactive Visual Analysis Setup based on the Defined DoI Values

With the modular DoI definition, the previous section detailed a new approach to define how much interest each graph element receives at each point of time. Once the DoI function is computed, the resulting DoI values provide an extremely valuable annotation to the data that can be used in many more aspects than just for reducing the network. Focusing on the DoI, we refrain from describing the architectural details and design issues, which our visual analysis setup has in common with most other existing graph visualization systems, such as TugGraph [2] or CGV [35]. Instead, this section illustrates specifically, how our visual analysis setup makes extensive use of the DoI values to provide DoI-based representations and interaction mechanisms. For this, we rely on two distinct views:

- The **DoI view** serves the purpose to interactively define and refine a DoI function that captures the user interest by exposing its functional components as reorderable, hierarchically stacked visual components.
- The **Network view** shows the result of applying the defined DoI function by agglomerating graph elements of lesser interest on the global network overview level, while highlighting graph elements of high interest on the local individual node level.

The overall setup consisting of these views is exemplified in Figure 1. The next two sections will discuss these two views, followed by a description of how the views are interlinked to support the user in defining and adjusting the DoI function to express his interest in detail.

4.1 The DoI View

The purpose of the DoI view is to interactively define *what will be shown* of the network by giving access to the DoI function and a few other parameters. This view goes well beyond the usual simple adjustment of some weights by exposing the entire DoI function to the user and allowing for its complete reconfiguration. The view consists of five panels that each has an important role in the course of the DoI definition:

The **network statistics** (Figure 1, panel (a)) provide a brief numerical overview on the current reduction of the network by showing how many graph elements exist (overall, in the current time step, and in its reduced form). These numbers give the user a concrete idea of how big the network is and how much of it is actually hidden from the Network view – in other time points, as well as inside the metanodes, into which elements of lesser interest are contracted.

The **DoI definition** (Figure 1, panel (b)) permits the user to adjust the DoI function that forms the basis of the reduction into metanodes. The view is an interactive, hierarchically structured graphical representation that directly encodes the different DoI components of the currently used DoI function. Each functional component defined in Section 3 corresponds to a matching visual component that can be plugged into the DoI view and that provides the necessary handles to adjust a component's parameters. Depending on the complexity of the DoI function, the embedded component views may require a large part of the available screen real estate. To counter this drawback, we enable the user to switch to a reduced view that only shows the left-hand labels of the component views in an Icicle-plot-like fashion and hides the visualizations and parameters on the right-hand side. While this reduces the width of the DoI view, the height can be reduced by folding known branches of the hierarchically composed view to a mere label as well. This feature is exemplified in one of the later screenshots in Section 5, where known DoI terms from previous visual analysis steps are folded and only the ones that have been newly added in the current step are shown in full detail.

The **manual selection** (Figure 1, panel (c)) gives access to individual graph elements for adding or removing them manually as high-interest nodes through the $select_d$-term in the DoI function. This access is quite important, as it permits the user to not only express his interest in a particular behavior

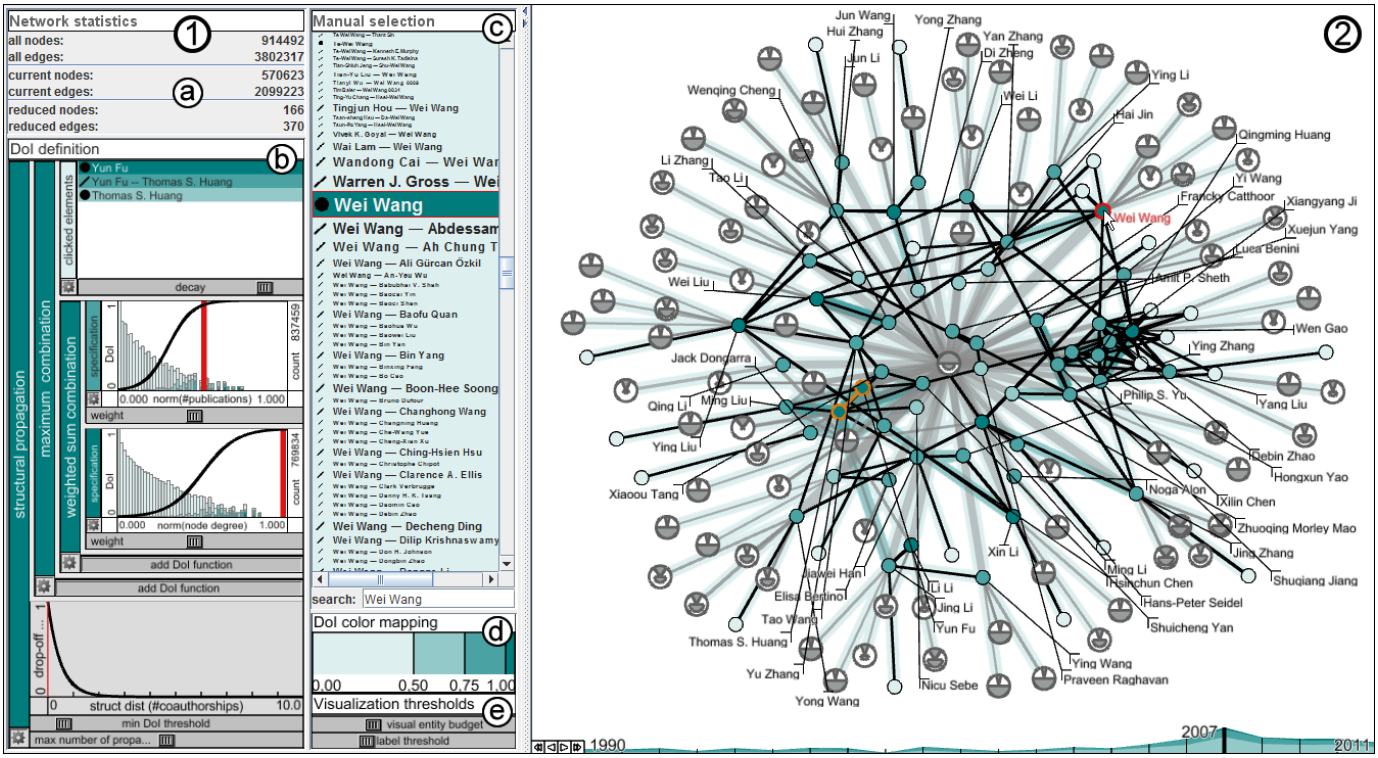


Fig. 1: The proposed overall visual setup with its two main views: (1) the DoI view and (2) the Network view. The Network view shows a snapshot of the DBLP dataset for the year 2007, which has been reduced according to the defined DoI function. The DoI view is partitioned in five panels (a)-(e). Panel (a) gives some numerical details on the reduction. Panel (b) shows the DoI function $doi(x_i) = prop_s(max(\{select_{0.85}(y_i), doi_{topAuthors}(y_i)\}), x_i)$. The term $doi_{topAuthors}(x_i) = sum(\{inter_1(norm(publications(x_i))), inter_2(norm(node\_degree(x_i)))\})$ selects the most active authors, while the *select*-term governs how graph elements picked by the user in panel (c) are treated. Here, they are added as focus nodes (*MAX* combination) and shown in orange in the Network view. The colors of all other elements are given by the adjustable color scale in panel (d) and the amount of reduction can be fine-tuned in panel (e). The labels of the DoI components in panel (b) are currently colored to reflect their individual DoI values for the hovered node "Wei Wang", which is in highlighted in red.

through the DoI function for *identification tasks* (indirect look-up), but also in certain graph elements for *localization tasks* (direct look-up). For this, we provide a compact list-based view that shows the search result for a search term typed at the bottom. The entries in the list are highlighted in the color that corresponds to the respective DoI value computed for them. In case it cannot be determined from the label itself, a small icon in front of each entry denotes whether an element is an edge or a node. Since there can be a large number of matching results, we use a fisheye distortion that couples the applied magnification with an entry's DoI value, so that elements with higher interest are shown in a larger font. To support the user in selecting the smaller entries of lesser interest, a table-lens enlarges the rows under the mouse cursor for an easier selection. The effect of the selection depends on how the $select_d$-term is used in the DoI function: a simple $MAX(doi, select_d)$-combination will result in an addition of the selected element, whereas a $MIN(doi, INV(select_d))$-combination will remove selected elements.

The **DoI color mapping** (Figure 1, panel (d)) allows the user to change how the DoI values are mapped onto color. In order to help distinguish elements of different interest more clearly, we chose to partition the full interval of DoI values $[0 \ldots 1]$ into multiple ranges. This way, one can set a clear threshold between two ranges of different interest and they will get assigned two clearly distinguishable colors, according to the range of DoI values in which they lie – something that would be hard to do with a continuous color scale. The number of the ranges and the interval of DoI values they span can be adjusted by the user directly within the interactive color scale. The color mapping set by this means is used throughout the DoI view and the Network view.

The **visualization thresholds** (Figure 1, panel (e)) are controlled via sliders that effectively permit to tune the size of the shown network in the Network view. As the graph is too large to be shown in its entirety, it has to be reduced according to the computed DoI values. For this, we successively contract the edge(s) whose averaged DoI values of its two incident nodes and itself is minimal across the entire network. Another option would be to make the probability of an edge to be contracted inversely proportional to the summed DoI values and then choose random edges for contraction w.r.t. this probability to yield a sampling effect. A contracted edge and its two nodes are subsumed by a metanode that receives the maximum DoI

value of all contained elements. The contraction is performed until the number of remaining nodes and edges falls below the *visual entity budget* [13] that the user defines via the first of the sliders. Such a visual entity budget is usually a better way to limit the size of the graph than a fixed cutoff with respect to the DoI value, as it maximizes what is shown no matter how skewed the distribution of DoI values is. The second slider governs the number of labels in the Network view.

4.2 The Network View

While the previous section detailed how the DoI view helps to determine *what will be shown* (i.e., the reduced network), this section discusses *how it is shown* in the Network view. For this, we consider those aspects, which tie in closely with the DoI values.

The **positioning** is done for each time point individually, yet in a fashion that ensures some stability across time points, so that browsing along the time axis gives a coherent impression of the overall behavior. For this, we combine a simple base layout with an additional stabilization mechanism. For the base layout, we employ the Fruchterman-Reingold force-directed layout [16] and generate it incrementally for each time point by taking the resulting layout of the previous time point as a starting point for the next time point's layout. As a stabilization mechanism, we use pinning weights, as it has been suggested in [15]. By choosing the pinning weight of a node to be proportional to its DoI value, a node's position becomes automatically more rigid, the higher its DoI value is. This way, nodes of higher interest can be followed more easily, while nodes of lesser interest are still allowed to adapt more freely, as they do not need to be tracked individually. In order to enhance the trackability of high-interest elements already during the force-directed layout, we assign spring constants to the edges that are proportional to the largest DoI value of its incident nodes. This mechanism is similar to the heat model used in [29]. Optimally, it creates a fisheye-like effect around nodes with high DoI values as it elongates its incident edges. But in cases of densely connected subsets of detail nodes, this effect is neutralized since all edges receive similarly high spring constants.

The **shapes** of the nodes are probably the most prominent feature to tell apart detail nodes and metanodes. While detail nodes are shown in the form of small circles, we aim to give metanodes a more distinct look that makes it immediately clear that they are representing multiple contracted nodes instead of a single detail node. We do so by embedding a glyph in the metanodes. These glyphs give some basic node/edge statistics on the respective collapsed subgraph they represent. It consists of a circle section on top of a semicircle and can have different appearances depending on the size and density of the subgraph, as schematically illustrated in Figure 2. The circle section at the top encodes information about the number of subsumed nodes, which is mapped logarithmically on its angle $\alpha = 180^\circ * \frac{log(\#nodes)}{log(max_{\#nodes})}$. Whereas, information about the subsumed edges is mapped in the form of the subgraph's density (the ratio of existing edges vs. all possible edges) logarithmically on the radius of the lower semicircle $r = r_{full} * \frac{log(\#edges)}{log(\#nodes*(\#nodes-1)/2)}$.

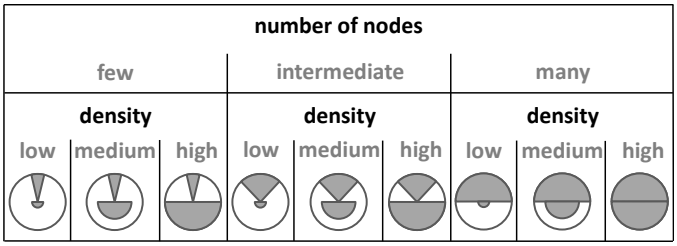


Fig. 2: Different appearances of the glyph encoding contracted subgraphs. The size of the subgraph is mapped to the angle of the circle section at the top. The density (number of edges out of all possible edges) is mapped to the radius of the circle section at the bottom. Thus, the glyph at the very left encodes a small set of nodes with minimal connectivity, whereas the full circle at the right represents a large clique.

The **color** follows the color mapping that is set in the DoI view. Detail nodes are color-coded directly according to their DoI value, while the color-coding of the edges is shown via halos [24] around the edges to not interfere with the topological information they convey. To better distinguish detail nodes from metanodes that form the context of the observed details, we furthermore color metanodes and edges that lead up to them as gray.

The **labeling** prioritizes nodes with higher DoI values for labeling overall and direct labeling specifically, so that high interest nodes are always labeled with best possible attribution of the label to them. We achieve this by utilizing the particle-based approach described in [25]. We carry out this approach starting with the detail node with the highest DoI value and stopping when the labeling threshold set in the DoI view is reached. This has the effect that nodes with higher DoI values also have a higher chance of getting a direct label, since not many other labels have been placed yet. Whereas nodes with lower DoI values are then more likely to receive eccentric labels as all the close spots are potentially already taken by previously placed labels. This also helps to set high-interest nodes apart from nodes with lower DoI values, for which labels are only available on demand.

The **interaction** facilities provided by the Network view are mainly the standard interaction techniques, such as zooming, panning, and relocating nodes. While these are not influenced by the DoI, they nevertheless affect the DoI-based view. For example, a zoom-in will lead to fewer visual objects being displayed in the zoomed-in part of the view. This leaves in turn more space for additional labels, which are automatically added to previously unlabeled nodes in this case. Other, more exotic couplings of standard interaction with the DoI values are easily imaginable. For example, when relocating a node, one could set the mouse movement speed inversely proportional to the node's DoI value, similar to the idea of pseudohaptics [21].

This way, nodes with higher DoI values would be harder to relocate not only for the layout algorithm due to the pinning weight, but also by the user who would thus get an additional feedback about the node's DoI value.

The **dynamics** of the network can be accessed via a DoI-based time slider in the form of a stacked graph [5] at the bottom of the Network view. As the buttons on its left-hand side indicate, it supports all the usual interactions, such as stepping through the time points one by one or playing an animation over all of them in both directions. Its overall height reflects the number of all elements having a non-zero DoI value at a given time point. It is subdivided into horizontal bands of the different color intervals chosen in the color scale in the DoI view. These bands are stacked from the lower DoI values to the highest ones, so that a user can judge the interestingness of a time point from the size of its topmost layers encoding the number of elements with high DoI values. To achieve that even the few elements of the highest DoI interval are visible, we do not simply add up element counts in the stacked graph, but their DoI values instead. This way, elements with a DoI value of zero are not taken into account and elements with higher DoI values get more emphasis. The encoding as a stacked graph informs a user about time points at which larger numbers of interesting graph elements occur. With this information, the user can pinpoint a time point of high interest and directly jump to it with a single mouse click.

4.3 Supporting DoI Definition and Adjustment with Presets and Linking

The DoI view and the Network view provide the interactive and graphical tools a user needs to pursue a DoI-based visual analysis of a dynamic network. As powerful as these tools are, their flexibility introduces additional complexity that a user has to master before the tools become useful. To aid in this process, we offer support by supplementing the views with two additional mechanisms:

- for the initial definition of a DoI function, we provide **pre-configured DoI presets** to choose from and visual cues for their parametrization, and
- for the interactive adjustment of a DoI function, we provide **linking between the views** to allow for a peek into the DoI computation for "debugging" and fine-tuning.

The **pre-configured DoI presets** and visual cues provide help for defining meaningful DoI functions and thus a sensible distinction between detailed focus nodes and contracted context nodes in the Network view. Presets are preassembled and preconfigured DoI definitions that can be selected as starting points and building blocks for one's own definition. As different domains and different types of input graphs require different presets, it is not possible to give a comprehensive list of such function blocks. For example, network intrusion detection requires different components than citation network analysis and trees require different functional DoI terms than a bipartite graph or a general network, as in each case the distinction between focus and context is made differently. Hence, DoI presets have to be tailored to the application domain and the graph type of the input data to be semantically meaningful to a user from a particular area. For example, for our use case from Section 5, we provide ready-made presets that capture the common interests in bibliometric research and citation analysis.

Once selected, the presets can then be adapted to the specific characteristics of the network to be analyzed by using the visual cues. These are given in the form of a histogram of the distribution of attribute values in the background of the DoI specification components. This way, a user can make an informed decision when adjusting the interest function directly on top of the values of the computation function, as they are shown in the histogram. Furthermore, the histogram provides feedback of how many nodes in each histogram bar are already assigned high DoI values through the currently specified DoI function. This is encoded in the histogram by means of stacked bar charts, which use the same global color ranges as they are defined in the color mapping.

The **linking between the views** ensures that the user can investigate in detail which functional component caused particular DoI values in the Network view. By hovering over an element in the Network view, it gets marked red and the DoI view changes the background color of the labels of all DoI components to reflect its DoI value. One can then track the high DoI values from the "root" of the DoI composition hierarchy down to the individual feature(s) that contributed the most to the resulting DoI value. In Figure 1, it apparently stems from the weighted sum combination, which is easily identified by its darker color in the functional DoI hierarchy, whereas for example the component of the clicked elements is colored in a lighter shade. Furthermore, the red marker at distance 0 at the bottom of the structural propagation component indicates that the high DoI value originates indeed from the highlighted element itself and was not propagated from another element. This marker denotes the spatial or temporal distance at which the element lies that propagated its DoI value to the hovered element. Histograms feature a similar marker to show where the investigated element lies in the value spectrum of a particular attribute. How this interlinking is used in a real world example is shown in the following section.

5 A Use Case Example

Citation networks and co-authorship networks are an important subject of study as their analysis can reveal interesting features about a scientific community and expose emerging research topics [28]. Especially when analyzed over time, they allow for observing the evolution of a scientific community. One source for such a co-authorship network particularly for the computer science research community is the DBLP database [23] that contains around 2 million publications as of today. Because of the size of this network, most analysis attempts are based merely on statistical evaluations of these networks or consider only publications from a specific community or specific conferences [4], [6], [12], [19], [27]. Such statistical evaluations represent the network in an abstracted, numerical way that yields at most lists of important authors or highly influential

Fig. 3: A view of the stacked graph along the timeline for the DoI function $doi_{filter}(x_i) = min(\{doi_{topAuthors}(x_i), inter_3(clustering\_coefficient(x_i))\})$, which captures top authors with low clustering coefficients. It can be seen that this does not yet pinpoint the trend of an increasing number of namesakes starting in 2006/2007.

publications without their neighborhood or any other structural information. Here, we report on our experiences depicting the DBLP co-authorship network (snapshot from June 2012) with our DoI-based visualization approach.

5.1 Data Description

In co-authorship networks, the nodes correspond to authors and an edge exists between two nodes iff the authors, whom these nodes represent, have published a paper together. We have extracted such graphs from the DBLP database for each year from 1990 to 2011. These graphs are cumulative, which means that each graph contains all authors and co-authorships that have occurred up to that year. This makes sense, as two persons having coauthored a publication once will stay coauthors forever. Additionally for the graph at each time point, the nodes are assigned the number of papers published by its corresponding author in that year and the edges are associated with the number of papers the two incident authors have published together in that year. Based on these attributes, it is possible to extract the network of a particular year by taking only nodes and edges into account, which have an attribute value > 0 in that year. As proposed in [27], we focus solely on the largest connected component of the network, as this component alone makes up 94.2% of the overall data set. This results in a dynamic network with 22 time points (years) containing 914,492 nodes and 3,802,317 edges.

5.2 Analysis Goals and First Steps

In the analysis of co-authorship networks, the main interest lies often on top authors. According to [12], [27] and [37, ch.5], these authors can be identified by a number of different characteristics, such as a high number of published papers and a high number of coauthors, which corresponds to a high node degree. These two aspects are easily combined into a DoI function by means of a weighted sum combination that takes both of them into account. Manually selected focus nodes are then added through another term, such as the one given in Section 3.2.3. As a last step, a structural DoI propagation is applied to ensure that the immediate neighborhoods of top authors and selected nodes also receive high DoI values and will thus be shown in their context. For the year 2007, the result is depicted in Figure 1.

This takes co-authorship network analysis as far as it would be possible with a fixed monolithic DoI function that has been developed to address this scenario. Maybe some other characteristics of top authors would also be included and one could change their weight and thus their influence on the combined DoI function. The result will always be the same: this works fine up to the year 2005, but starting around 2006/2007 one can observe that the visualization is taken over by a large number of Asian names. The difficulty with these Asian names is that many of them get listed as top authors with a large number of publications and coauthors, because they actually represent more than one person. For example, the author(s) *Wei Wang*[1] (highlighted in Figure 1) actually subsumes over 40 different individual persons. The cause of this known problem is the use of the author's name as the key for identifying a person in the DBLP database, which also leads to the problem that authors appear multiple times due to different spellings [22]. A tool with a monolithic DoI function cannot adapt to such specifics of individual datasets and will thus show incorrect results.

With our approach, however, a visual analysis of the top authors does not have to end here, because we can alter the DoI function to filter out the namesakes and yield a correct view of the top authors. This is detailed in the following section.

5.3 Further Refinement through DoI Modification

We take a two-step approach for our analysis, which aims first at finding a DoI function that pinpoints the namesakes and then subtracts them from the initial DoI function in order to get a view that is unperturbed by this phenomenon.

To select these namesakes, it is necessary to find characteristics allowing to discern them from regular authors. A first such characteristic is based on the observation that authors who have published with the same person are also likely to have published together [28]. As a result, this person with whom they have published will have a higher clustering coefficient, meaning his immediate neighborhood is well connected. Contrary to this, coauthors of namesakes are not as well connected, because the multiple "overloaded" persons belong to different scientific communities, and so do their coauthors.

The problem with using this characteristic alone can be observed already in the stacked graph along the timeline, which is shown in Figure 3: while the trend of a growing number of namesakes was identified to begin in 2006/2007, the clustering coefficient captures a number of authors throughout the entire time interval. It turns out that this characteristic is somewhat too generous in declaring people as namesakes, as there is another category of people, who also fit this characteristic, but are not namesakes: advisors who have published with generations of graduate students with only few joint publications between these students due to temporal separation and thus also resulting in a low clustering coefficient. Advisors can be told apart from namesakes by their slow increase of coauthors, which is likely to correspond to a few new grad students each year. Whereas namesakes exhibit an almost unreal surge of new coauthors each year that is due to the combined research collaborations of multiple persons subsumed under a single

1. http://dblp.uni-trier.de/pers/hd/w/Wang:Wei.html

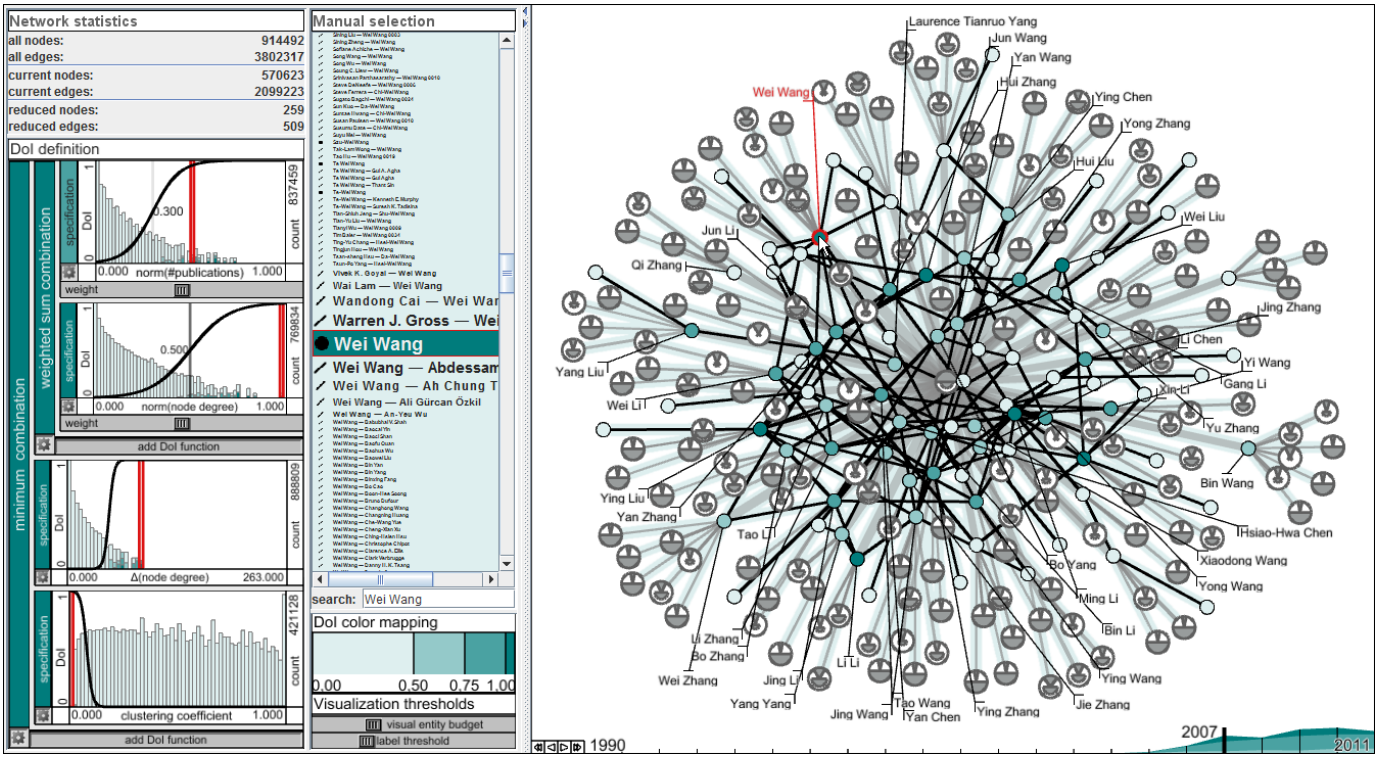


Fig. 4: A full view of the DBLP dataset for 2007, capturing all top authors with low clustering coefficients and a high increase of node degree over time, which is expressed through the DoI function $doi_{filter}(x_i) = min(\{doi_{topAuthors}(x_i), inter_3(clustering\_coefficient(x_i)), inter_4(change(node\_degree(x_i)))\})$. The stacked graph view along the timeline shows that this refined DoI function better fits the observed starting point and the selected author(s) *Wei Wang* have correctly been identified as multiple persons by the function.

name. So, we combine both aspects – the low clustering coefficient and the high increase in node degree (2nd column of Table 1) – together with the original characteristics of top authors – a high publication count and a high node degree. The result of this DoI combination is shown in Figure 4, again with *Wei Wang* highlighted. From the stacked graph along the time axis, one can clearly see how this phenomenon starts in 2006/2007, as we had observed it before. The resulting DoI function pinpoints the namesakes well enough to use it in combination with the initial DoI function to yield a tidied-up view of the top authors.

To filter out these namesakes, we have to subtract them from the top authors. This subtraction can be performed through a minimum combination of the top author selection from Figure 1 and the inversion of the namesake selection from Figure 4. However, the selection of namesakes is based on a temporal feature (the change of node degree) that only leads to a high DoI value for a node in a specific year if the steep increase actually occurred in that year. As a result, authors are classified as namesakes in one year, while showing an unsuspicious behavior in their node degree in other years. Hence, we define a namesake as such starting from the year it was identified as one. Therefore, before the actual subtraction, we utilize a temporal propagation that distributes high DoI values corresponding to a found namesake to all future years. The result of the subtraction is shown in Figure 5. As a result of the number of namesakes being reduced, there is now room in the visual entity budget to show actual top authors who have previously been hidden. To demonstrate the subtraction, *Wei Wang* has again been highlighted to show that while this node fulfills all aspects of a top author (the weighted sum component in the DoI view is shown in dark cyan), it also conforms to the characteristics of a namesake (folded function block "namesakes" is colored even darker).

Establishing and verifying such a tailored DoI function would have been very cumbersome without our approach that permitted us to derive it interactively with immediate visual feedback. The use case also illustrated the concerted use of all three of our newly introduced concepts: the modular DoI specification to iteratively refine the DoI function, the specification component capturing the dynamics of graph elements, and the propagation of DoI values over time to spread a once identified namesake to all future time points. Now that such a DoI function has been interactive identified, it would be possible to use it in a pre-computation or data cleansing step to identify namesakes in future co-authorship input data that matches authors by their names.

6 Conclusion

In this paper, we have introduced a flexible modular DoI specification for dynamic graphs, which goes beyond the established fixed monolithic DoI-based approaches. While DoI-

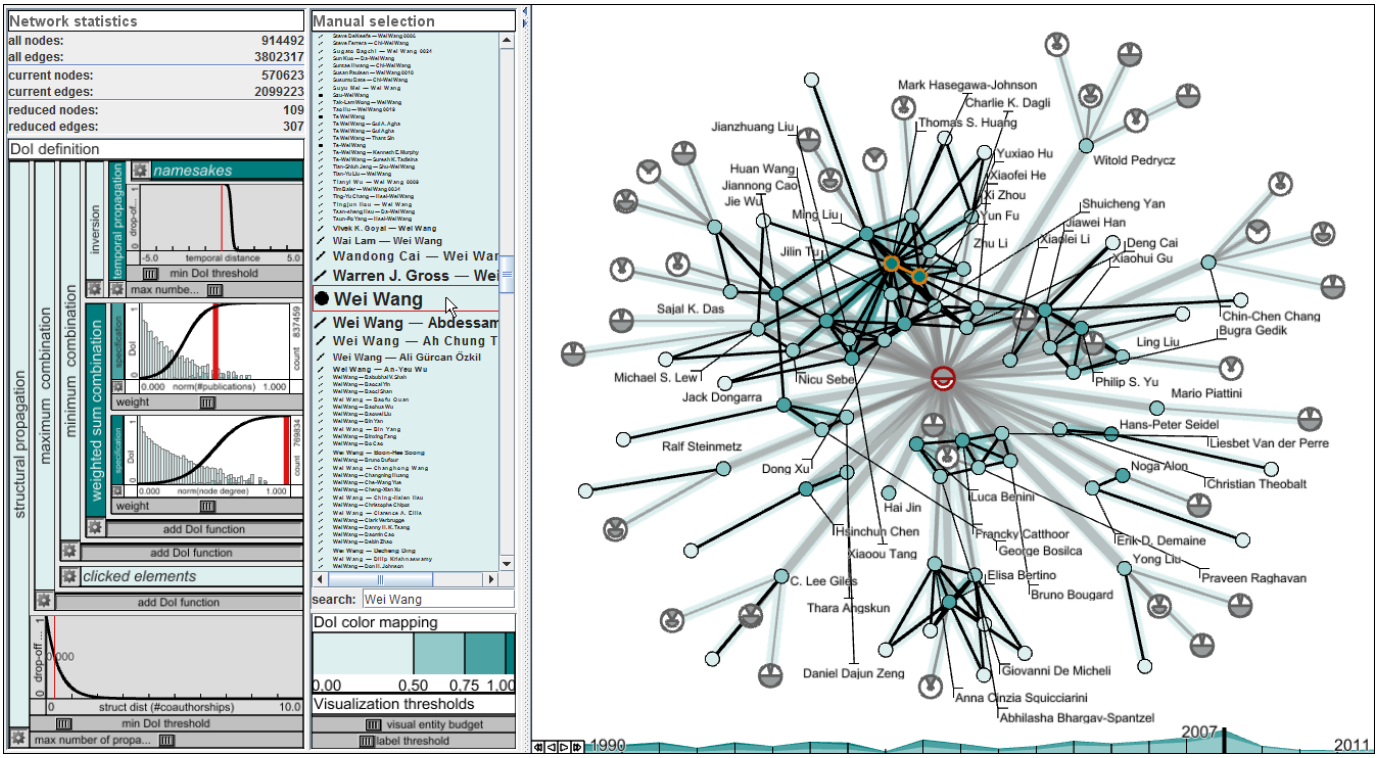


Fig. 5: The same view as in Figure 1, but this time with the found namesakes from Figure 4 having been subtracted from it. This has been achieved by combining both DoI functions into $doi(x_i) = prop_s(max(\{select_{0.85}(y_i), min(\{inv(prop_t(doi_{filter}(y_j)), y_i), doi_{topAuthors}(y_i)\})\}), x_i)$.

based techniques are usually used for generating focus+context visualizations, we take advantage of the flexibility of our approach to employ it for supporting visual analysis of large dynamic networks. This is made possible by providing a user with the capability to express dynamic features of interest and to assemble these features into more complex patterns. Nodes and edges that fit a pattern (e.g., a certain specified dynamic) are then put into focus of an otherwise abstracted context representation of the remainder of the network. As such, our approach indeed permits to analyze local changes, while at the same time maintaining an abstracted overview of the global network dynamics.

Our approach stands and falls with the ability to specify meaningful DoI functions that exactly reflect a given dynamic pattern a user is interested in. Thus, if our approach was to be extended in future work, it would be first and foremost by adding even more ways to specify features. This regards mainly a better support for the specification of features that are intricately placed in a high dimensional, multi-faceted attribute space. While it is possible to capture them with a combination of the DoI components that we provide, this may require a large number of base DoI functions and be very tedious to put together. Hence, it would be desirable to provide components in which such complex patterns can be specified directly, for example, by selecting a 2D region in a scatterplot of two different attributes ($inter(comp_1(\ldots), comp_2(\ldots))$) or of one attribute vs. the time axis ($inter(comp(\ldots), T)$). Another way to ease the specification of complex patterns in a high-D feature space would be to also allow for defining DoI values based on principal components, as proposed in [33].

ACKNOWLEDGMENTS

This work was supported in part by the German Research Foundation (DFG), the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), and mgvis.com. The authors thank Christian Tominski, Martin Luboschik, and Youwei Zheng for their valuable feedback on earlier versions of this paper.

REFERENCES

[1] J. Abello, T. Eliassi-Rad, and N. Devanur. Detecting novel discrepancies in communication networks. In *Proceedings of the IEEE International Conference on Data Mining*, pages 8–17, 2010.

[2] D. Archambault, T. Munzner, and D. Auber. TugGraph: Path-preserving hierarchies for browsing proximity and paths in graphs. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 113–120, 2009.

[3] G. Beliakov, A. Pradera, and T. Calvo. *Aggregation Functions: A Guide for Practitioners*. Studies in Fuzziness and Soft Computing. Springer, 2007.

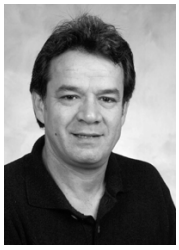
[4] M. Biryukov and C. Dong. Analysis of computer science communities based on DBLP. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries*, pages 228–235, 2010.

[5] L. Byron and M. Wattenberg. Stacked graphs – geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252, 2008.

[6] N. Cao, D. Gotz, J. Sun, Y.-R. Lin, and H. Qu. SolarMap: Multifaceted visual analytics for topic exploration. In *Proceedings of the IEEE International Conference on Data Mining*, pages 101–110, 2011.

[7] S. Card, B. Sun, B. Pendleton, J. Heer, and J. Bodnar. TimeTree: Exploring time changing hierarchies. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 3–10, 2006.

[8] T. Crnovrsanin, I. Liao, Y. Wu, and K.-L. Ma. Visual recommendations for network navigation. *Computer Graphics Forum*, 30(3):1081–1090, 2011.
[9] T. d'Entremont and M.-A. Storey. Using a degree of interest model to facilitate ontology navigation. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 127–131, 2009.
[10] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the Symposium on Data Visualisation*, pages 239–248, 2003.
[11] H. Doleisch, H. Hauser, M. Gasser, and R. Kosara. Interactive focus+context analysis of large, time-dependent flow simulation data. *Simulation*, 82(12):851–865, 2006.
[12] E. Elmacioglu and D. Lee. On six degrees of separation in DBLP-DB and more. *SIGMOD Record*, 34(2):33–40, 2005.
[13] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2009.
[14] M. Farrugia and A. Quigley. Effective temporal graph layout: A comparative study of animation versus static display methods. *Information Visualization*, 10(1):47–64, 2011.
[15] Y. Frishman and A. Tal. Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, 2008.
[16] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software–Practice & Experience*, 21(11):1129–1164, 1991.
[17] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, 1986.
[18] J. Heer and S. K. Card. DOITrees revisited: Scalable, space-constrained visualization of hierarchical data. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 421–424, 2004.
[19] T.-H. Huang and M. Huang. Analysis and visualization of co-authorship networks for understanding academic collaboration and knowledge domain of individual researchers. In *Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation*, pages 18–23, 2006.
[20] P. Hüsken and J. Ziegler. Degree-of-interest visualization for ontology exploration. In *Proceedings of the IFIP TC 13 International Conference on Human-Computer Interaction*, pages 116–119, 2007.
[21] A. Lécuyer, J.-M. Burkhardt, and L. Etienne. Feeling bumps and holes without a haptic interface: The perception of pseudo-haptic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 239–246, 2004.
[22] M. Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *Proceedings of the International Symposium on String Processing and Information Retrieval*, pages 1–10, 2002.
[23] M. Ley. DBLP – some lessons learned. *Proceedings of the VLDB Endowment*, 2(2):1493–1500, 2009.
[24] M. Luboschik and H. Schumann. Illustrative halos in information visualization. In *Proceedings of Working Conference on Advanced Visual Interfaces*, pages 384–387, 2008.
[25] M. Luboschik, H. Schumann, and H. Cords. Particle-based labeling: Fast point-feature labeling without obscuring other visual features. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1237–1244, 2008.
[26] T. May, M. Steiger, J. Davey, and J. Kohlhammer. Using signposts for navigation in large graphs. *Computer Graphics Forum*, 31(3pt2):985–994, 2012.
[27] M. A. Nascimento, J. Sander, and J. Pound. Analysis of SIGMOD's co-authorship graph. *SIGMOD Record*, 32(3):8–10, 2003.
[28] M. E. J. Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5200–5205, 2004.
[29] N. Osawa. A multiple-focus graph browsing technique using heat models and force-directed layout. In *Proceedings of the International Conference on Information Visualisation*, pages 277–283, 2001.
[30] A. Perer and F. van Ham. Integrating querying and browsing in partial graph visualization. Technical Report 12-01, IBM Research, 2012.
[31] F. Reitz, M. Pohl, and S. Diehl. Focused animation of dynamic compound graphs. In *Proceedings of International Conference on Information Visualisation*, pages 679–684, 2009.
[32] L. Ren, L. Zhang, D. Teng, G. Dai, and Q. Li. DOI-Wave: A focus+context interaction technique for networks based on attention-reactive interface. In M. L. Huang, Q. V. Nguyen, and K. Zhang, editors, *Visual Information Communication*, chapter 5, pages 85–94. Springer, 2009.
[33] C. Salama, M. Keller, and P. Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1021–1028, 2006.
[34] H.-J. Schulz, M. John, A. Unger, and H. Schumann. Visual analysis of bipartite biological networks. In *Proceedings of the Eurographics Workshop on Visual Computing for Biomedicine*, pages 135–142, 2008.
[35] C. Tominski, J. Abello, and H. Schumann. CGV – an interactive graph visualization system. *Computers and Graphics*, 33(6):660–678, 2009.
[36] F. van Ham and A. Perer. "Search, show context, expand on demand": Supporting large graph exploration with degree-of-interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):953–960, 2009.
[37] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1994.



James Abello is Research Professor at the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Rutgers University. He has been a Senior Research scientist at Ask.com and Senior Member of Technical Staff at AT&T Shannon Laboratories and Bell Labs. He has lead the development of various software systems and is the co-editor of External Memory Algorithms, Vol. 50 of the AMS-DIMACS series, The Kluwer Handbook of Massive Data Sets, and Discrete Methods in Epidemiology. More about his research can be found at http://www.mgvis.com.

Steffen Hadlak received the diploma (MCS) in 2009 from the University of Rostock. There he is currently working toward the PhD degree on multilevel visualization of dynamic networks for different applications such as bioinformatics and communication network analysis. His research interests include GPU-based rendering and alternative interaction techniques as well as their application to Information Visualization. More about his research can be found at http://www.informatik.uni-rostock.de/~hadlak/.

Heidrun Schumann received her Master's, PhD, and habilitation degrees in 1977, 1981, and 1989, respectively, from the University of Rostock, where she is heading the Computer Graphics Research Group since 1992. Her research covers Information Visualization, Visual Analytics, and Rendering. Her current projects, supported by funding agencies and industry, include scalable frameworks for information visualization and adaptive visual interfaces. More about her research can be found at http://www.informatik.uni-rostock.de/~schumann/.

Hans-Jörg Schulz received his master's (2004) and doctorate degree (2010) from the University of Rostock, where he is currently working on a research project on the visualization of heterogeneous data. In this context, he is developing graph-based visualization concepts for data stemming from multiple sources. He applies his research in the biomedical domain and in systems biology. More about his research can be found at http://www.informatik.uni-rostock.de/~hs162/.