


Structure Identification—
Concept, Operationalization, and Application

Oliver Niggemann Benno Stein

Dept. of Mathematics and Computer Science—Knowledge-based Systems Group
email: {murray,stein}@uni-paderborn.de

September, 1998

 University of Paderborn
Department of Mathematics / Computer Science
Knowledge-based Systems Group
33095 Paderborn, Germany

Series Computer Science
September, 1998

Structure Identification— Concept, Operationalization, and Application

Oliver Niggemann Benno Stein

*Dept. of Mathematics and Computer Science—Knowledge-based Systems Group
University of Paderborn, D-33095 Paderborn, Germany
Email: {murray,stein}@uni-paderborn.de*

Abstract

When working on systems of the real world, abstractions in the form of graphs have proven a superior modeling and representation approach. This paper is on the analysis of such graphs. Based on the paradigm that a graph of a system contains information about the system's structure, the paper contributes within the following respects:

Starting with an informal introduction of the term “structure”, the role of structure identification in different problem classes is outlined. The central contributions of this paper are (i) a formal structure measure, the so-called weighted partial connectivity, Λ , whose maximization defines a graph's structure (Section 2), and (ii) a fast algorithm that approximates a graph's optimum Λ value (Section 3).

Moreover, the proposed structure definition is compared to existing clustering approaches, resulting in a new splitting theorem concerning the well-known minimum cut splitting measure. A key concept of the proposed structure definition is its implicit determination of an optimum number of clusters.

Two examples, which illustrate the usability of the measure, round off the paper.

Keywords: structure identification, graph analysis, clustering, knowledge-based methods.

1 What is Structure?

“Structure defines the organization of parts as dominated by the general character of the whole.”

“Structure defines the aggregate of elements of an entity in their relationships to each other.”¹

These informal definitions reflect the common sense understanding of the notion “structure”. Structure information is some kind of meta information and may take different shapes. However, the nature of a graph often resembles structure information—Figure 1 shows a gantry crane, its graph representation, and related structural abstractions.

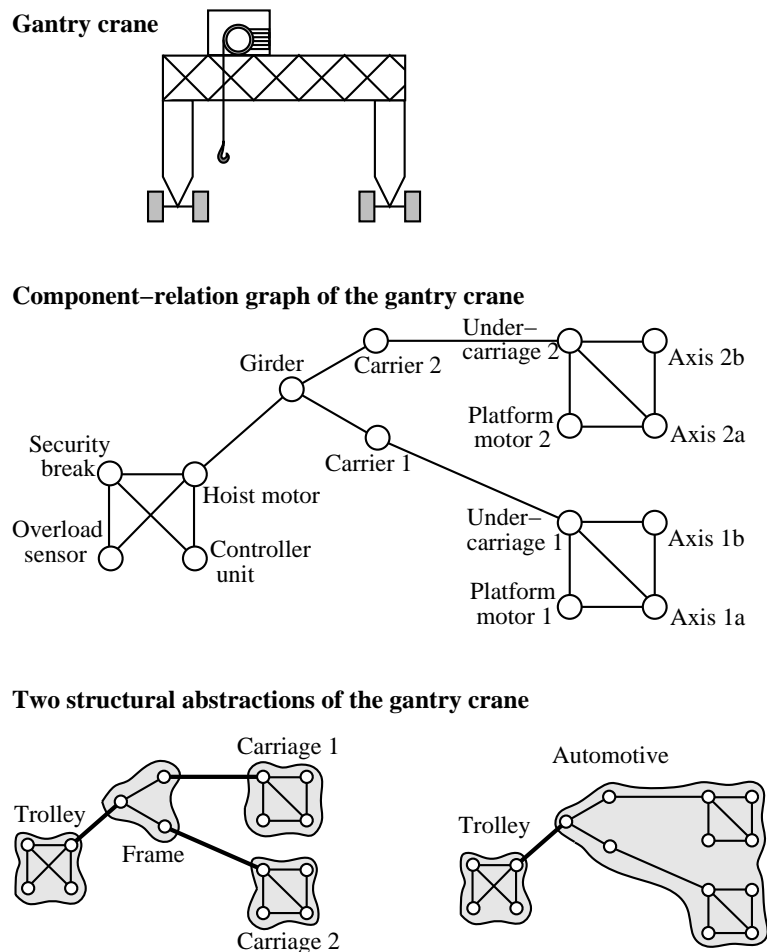


Figure 1: Graph representation and structure of a crane.

To allow of a more formal definition of the term structure, the following abstractions are useful:

1. The interesting system, in the above definition designated by the terms “whole” and “entity” respectively, is mapped onto a graph, $G = \langle V, E \rangle$.² The system’s

¹The Merriam-Webster’s Collegiate Dictionary, Tenth Edition.

²See Appendix A for a concise definition of the graph-theoretical concepts used in this paper.

elements form the set of nodes, V ; the relations between the elements are represented by the set of edges, E .

2. The system's structure (its "general character") is reflected by the *distribution* of G 's edges.

Of course several concepts are conceivable that define in which way a system's structure is reflected by the distribution of its graph edges. The understanding of structure as it is subject to this paper relies on the following paradigms:

1. *Domain concepts.* Associated to G , a set of $1 \leq i \leq |V|$ domain concepts c_i along with a mapping from V to the c_i can be stated.

Each domain concept corresponds to a particular function, devision, module, or role of the system. All elements assigned to the same domain concept contribute to the same function, say, each c_i defines a unary predicate on the system elements.

2. *Connectivity.* Domain concepts are defined implicitly, merely exploiting the graph-theoretical concept of connectivity:

The connectivity between nodes assigned to the same domain concept c_i is assumed to be higher than the connectivity between any two nodes v and w , where v, w are assigned to c_i and $c_{j, j \neq i}$ respectively.

3. *Contraction.* A system's structure is defined as that contraction of G where a single node is substituted for all nodes of the same domain concept.

Remarks. Point 1 reflects hierarchy or decentralization aspects of a system's or an organization's structure. Point 2 is based on the observation, that the elements within a module are closely related; the modules themselves, however, are coupled by narrow interfaces only. A similar observation can be made respecting organizational or biological structures. Point 3 states that structure information can be derived by a rather simple abstraction method.

These structuring paradigms may not apply to all kinds of systems—but, for a broad class of (technical) systems they form a useful set of assumptions.

1.1 The Role of Structure Identification in Different Problem Classes

Structure identification, as defined by the three paradigms above, could be regarded as a weak (or basic) *problem solving method*.³ Although the term "structure identification by graph contraction" does not imply a concrete algorithm (a problem solving method should), it defines, on the other hand, a clear purpose respecting the processing of domain knowledge. Moreover, structure identification is related to *weak* methods

³A problem solving method designates an algorithm that describes in which way domain knowledge is utilized to solve a problem. A weak problem solving method is less specialized respecting knowledge representation, and thus its range of application is broader. Examples for weak problem solving methods are forward-chaining-with-rules or the hypothesize-and-test strategy. Strong problem solving methods can be considered as weak methods that have been tailored towards a particular domain or situation [16].

rather than to strong ones, since it provides a basic (preprocessing) step, which is independent from a domain or a problem class. In the following, examples for a graph-based structure identification in different problem classes are given.

- *Diagnosis.* Complex diagnoses problems are tackled by a hierarchical approach, which breaks up the entire problem by focusing. The focusing step may render a heuristic diagnosis step, while the remaining smaller problem may be solved by a model-based diagnosis step. Focusing means concentrating on a subsystem, which could be isolated by structure identification. In [8] the authors pursue such a strategy: Within a structure identification step a complex hydraulic system is decomposed into so-called hydraulic axes, which in a second step are treated locally.
- *Configuration.* Resource-based configuration is a promising approach for configuring modular technical systems [7, 21]. In order to apply this configuration paradigm, each component of the system is modeled locally; the components are connected to each other via so-called properties, which they supply or demand. The configuration algorithm tries to satisfy some given initial demand by choosing a suitable set of components.

When working on a resource-based configuration problem, a domain-oriented interpretation of the underlying component-property graph is of a great value [21]. By smartly clustering this graph, a functional structure within a complex technical system can be identified, crucial points in the modeling exhibited, the configuration algorithm be tailored, or the knowledge base of a large system organized into useful parts [14].

- *Visualization.* To visualize complex graphs, a preprocessing in the form of node clustering followed by graph contraction has proven to be a key strategy [22]. Several concepts have been developed, which rely on clustering when arranging a graph's nodes hierarchically [5, 18], on a grid [19, 4], or by means of simulated annealing [14].
- *Monitoring.* When monitoring network traffic, the communication intensity is reflected by the network communication matrix. This matrix can be interpreted as a (weighted) virtual network graph, which is embedded in the real network. Monitoring and analysing the network traffic corresponds to the identification of the network graph's structure.

Section 5 discusses a visualization and a monitoring application in greater detail.

2 Quantifying a Graph's Structure

The structure of a system G has been introduced as some contraction of G . This descriptive definition can be quantified by means of a new measure called “weighted partial connectivity”, Λ , which is introduced now. The weighted partial connectivity is defined for a decomposition of a graph G , and it is based on the graph-theoretical concept of edge connectivity.

Let $G = \langle V, E \rangle$ be the graph abstraction of the interesting system.

1. $\mathcal{C}(G) = (C_1, \dots, C_n)$ is a *decomposition* of G into n subgraphs induced on the C_i , if $\bigcup_{C_i \in \mathcal{C}} C_i = V$ and $C_i \cap C_{j, j \neq i} = \emptyset$. The induced subgraphs $G(C_i)$ are called *cluster*. $E_{\mathcal{C}} \subseteq E$ consists of the set of edges between the clusters.
2. The *edge connectivity* of a graph G denotes the minimum number of edges that must be removed to make G a not-connected graph (see Appendix A for details).

Definition 2.1 (Λ). Let G be a graph, and let $\mathcal{C} = (C_1, \dots, C_n)$ be a decomposition of G . The *weighted partial connectivity* of \mathcal{C} , $\Lambda(\mathcal{C})$, is defined as

$$\Lambda(\mathcal{C}) := \sum_{i=1}^n |C_i| \cdot \lambda_i,$$

where $\lambda(C_i) \equiv \lambda_i$ designates the edge connectivity of $G(C_i)$.

Figure 2 illustrates the weighted partial connectivity measure Λ .

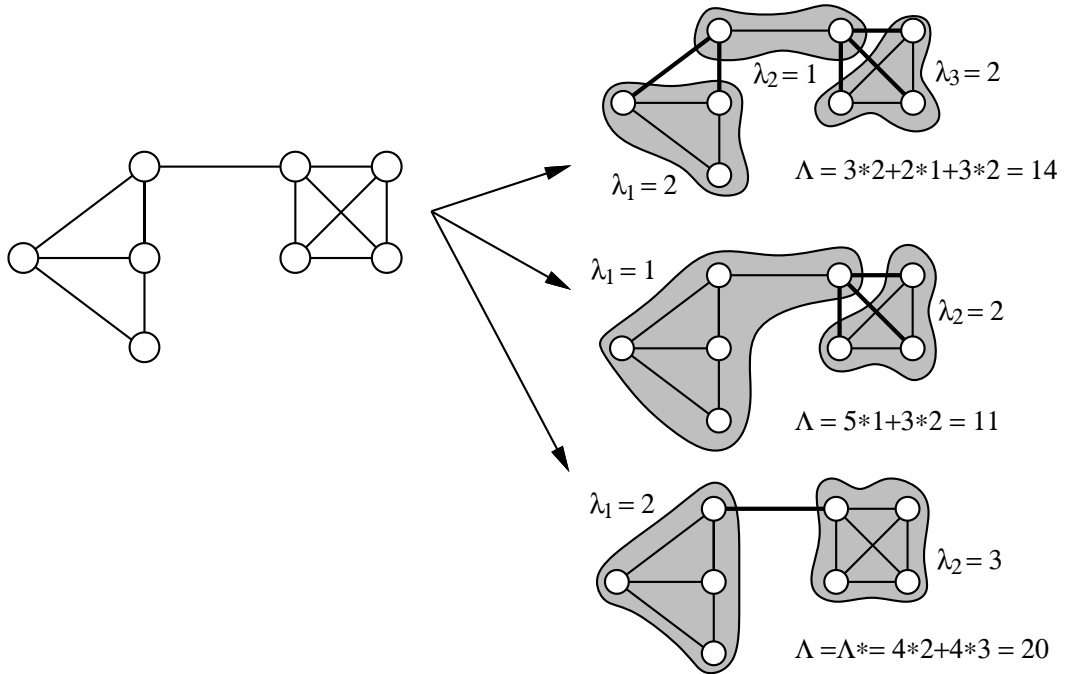


Figure 2: Example for graph decompositions and related Λ values.

Definition 2.2 (Connectivity Structure). Let G be a graph, and let \mathcal{C}^* be a decomposition of G that maximizes Λ :

$$\Lambda(\mathcal{C}^*) \equiv \Lambda^* := \max\{\Lambda(\mathcal{C}) \mid \mathcal{C} \text{ is a decomposition of } G\}$$

Then the contraction $H = \langle \mathcal{C}^*(G), E_{\mathcal{C}^*} \rangle$ is called *connectivity structure* (or simply: *structure*) of the system represented by G .

Figure 3 shows that Λ maximization means structure identification.

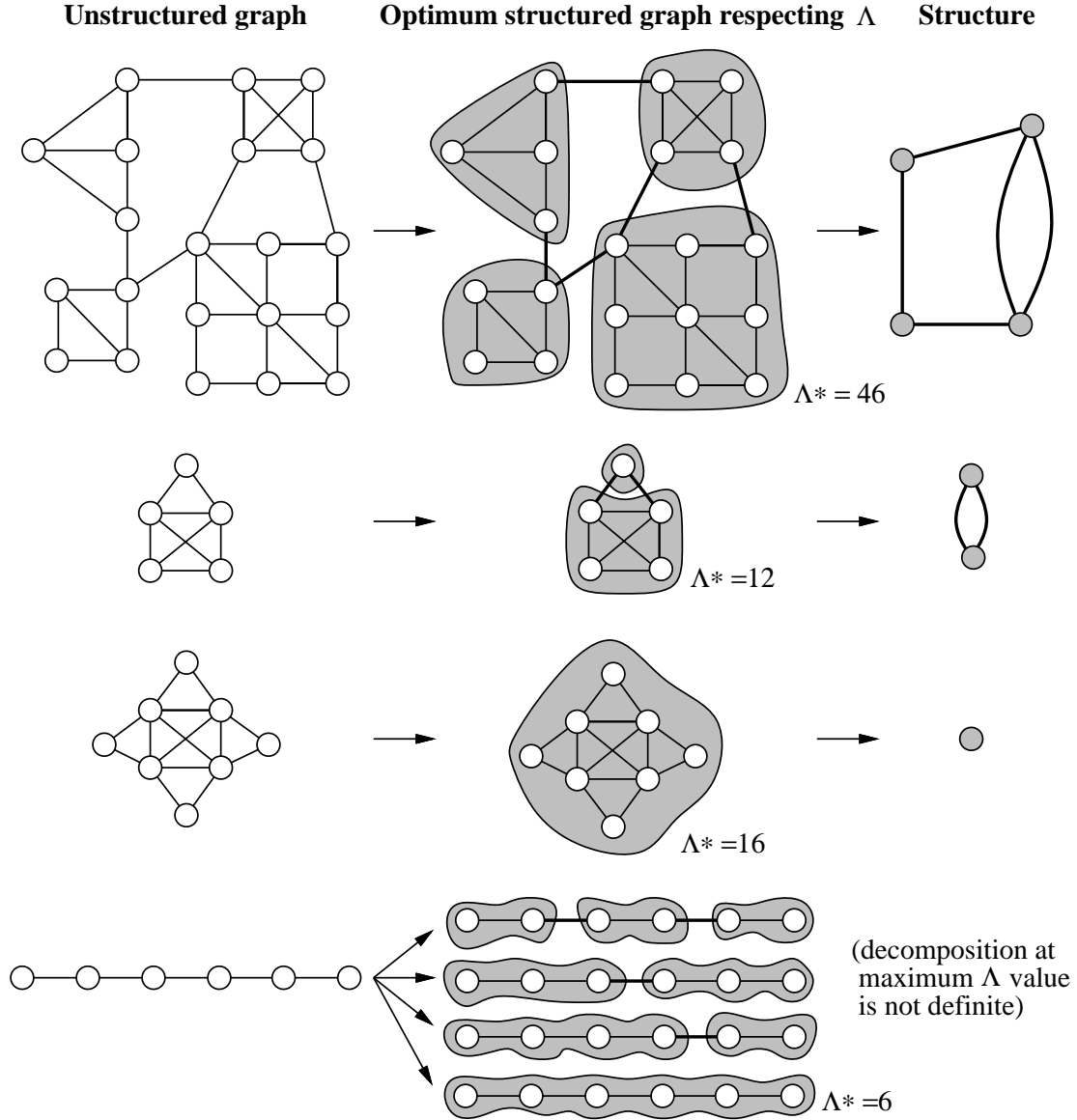


Figure 3: Examples for decomposing a graph according to our structure definition.

Remarks. A key feature of the structure definition is that a structure's number of clusters is defined implicitly.

Two rules of decomposition, which are implied in the above structure definition, are worth to be noted.

(i) If for a (sub)graph $G = \langle V, E \rangle$ and a decomposition (C_1, \dots, C_n) the *strong splitting condition*

$$\lambda(G) < \min\{\lambda_1, \dots, \lambda_n\}$$

is fulfilled, G will be decomposed. Note that the strong splitting condition is commensurate for decomposition, and its application lessens the mean value of the standard deviations of the clusters' connectivity values λ_i . Obviously this splitting rule follows the human sense when identifying clusters in a graph, and there is a relation to the Min-Cut splitting approach, which is derived in Section 4.

(ii) If for no decomposition \mathcal{C} the strong splitting condition holds, G will be decomposed only, if for some \mathcal{C} the condition $|V| \cdot \lambda(G) < \Lambda(\mathcal{C})$ is fulfilled. This inequality forms a necessary condition for decomposition—it is equivalent to the following special case of the structure definition: $\max\{\Lambda(\{V\}), \Lambda(\mathcal{C})\} = \Lambda(\mathcal{C})$, because $\Lambda(\{V\}) \equiv |V| \cdot \lambda(G)$.

The weighted partial connectivity, Λ , can be made independent of the graph size by dividing it by the graph's node number $|V|$. The resulting normalized Λ value is designated by $\bar{\Lambda} \equiv \frac{1}{|V|} \cdot \Lambda$.

3 Operationalizing Structure Identification

In this section a fast clustering algorithm optimizing the weighted partial connectivity Λ is presented. This algorithm implements a local heuristic and is therefore suboptimal.

Initially, the algorithm assigns each node of a graph its own cluster. Within the following re-clustering steps, a node adopts the same cluster as the majority of its neighbors belong to. If there exist several such clusters, one of them is chosen randomly. If re-clustering comes to an end or rather unlikely, the algorithm terminates.

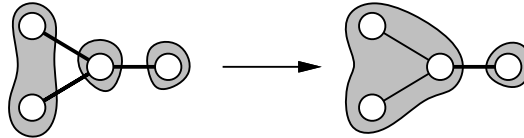


Figure 4: A definite majority clustering situation.

Figure 4 shows the definite case; most of the neighbors of the central node belong to the left cluster, and the central node becomes a member of that cluster. In the situation of Figure 5 the central node has the choice between the left and the right cluster.

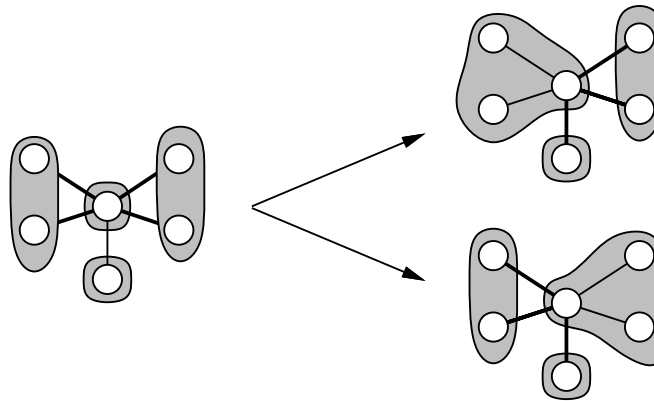


Figure 5: An undecided majority clustering situation.

We now introduce this algorithm formally.

MAJORCLUST.

Input. A graph $G = \langle V, E \rangle$.

Output. A function $c : V \rightarrow \mathbf{N}$, which assigns a cluster number to each node.

- (1) $n = 0, t = false$
- (2) $\forall v \in V$ **do** $n = n + 1, c(v) = n$ **end**
- (3) **while** $t = false$ **do**
- (4) $t = true$
- (5) $\forall v \in V$ **do**
- (6) $c^* = i$ **if** $|\{u : \{u, v\} \in E \wedge c(u) = i\}|$ is max.
- (7) **if** $c(v) \neq c^*$ **then** $c(v) = c^*, t = false$
- (8) **end**
- (9) **end**

Remarks. If a node is neighbored to more than one maximum cluster, there is an indeterminacy respecting the cluster choice in step 6. Figure 6 shows an extreme situation respecting such a random choice. A vertically decomposition as shown on the right hand side is unlikely, because if some of the nodes at the corners is moved to the other cluster, only that cluster will be left at the end of the turn. Hence, this clustering is unstable, which is also mirrored by the fact that the presented decomposition does, unlike the trivial decomposition into only one cluster, not maximize the weighted partial connectivity Λ . However, a decomposition by some horizontal splitting line as shown on the left side is more likely and more stable. Noticeably, such a decomposition is also optimal regarding Λ .

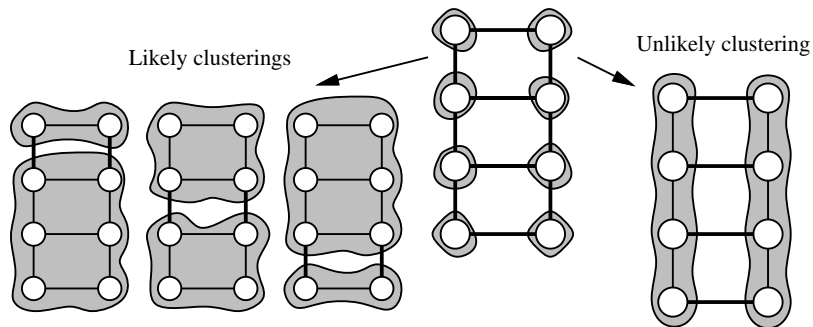


Figure 6: Likely and unlikely clusterings in the standoff case.

The runtime complexity of MAJORCLUST is $\Theta(|E| \cdot |C_{max}|)$, where $C_{max} \subseteq V$ designates a maximum cluster. In the While-loop (line 3 to 8) each edge of G is investigated twice; within each pass, a growing cluster is enlarged by at least one node; if no node changes its cluster MAJORCLUST terminates. Note that this evaluation neglects “pathological” cases, where the algorithm oscillates between two (or more) decompositions. However, such a situation constitutes neither a clustering nor a runtime problem: It can be detected easily since all nodes are either stable or in an undecided constellation. This advisements and experimental results (see Section 5) show the usability of the algorithm for large graphs with several thousand nodes.

The algorithm’s greatest strength, its restriction to local decisions, is bound up with its suboptimality. In every step only a node’s neighbors are considered, resulting in an excellent runtime behavior. On the other hand, by disregarding global criteria like the connectivity, MAJORCLUST cannot always find the optimum solution. Figure 7 illustrates this.

The optimum solution for graph (a) is one cluster, which is also the solution as found by MAJORCLUST. For graph (b), a splitting into the two clusters $\{v_1\}$ and $V \setminus \{v_1\}$ is optimum. MAJORCLUST cannot find this decomposition—working strictly locally, it behaves exactly as on graph (a) and creates only one cluster.

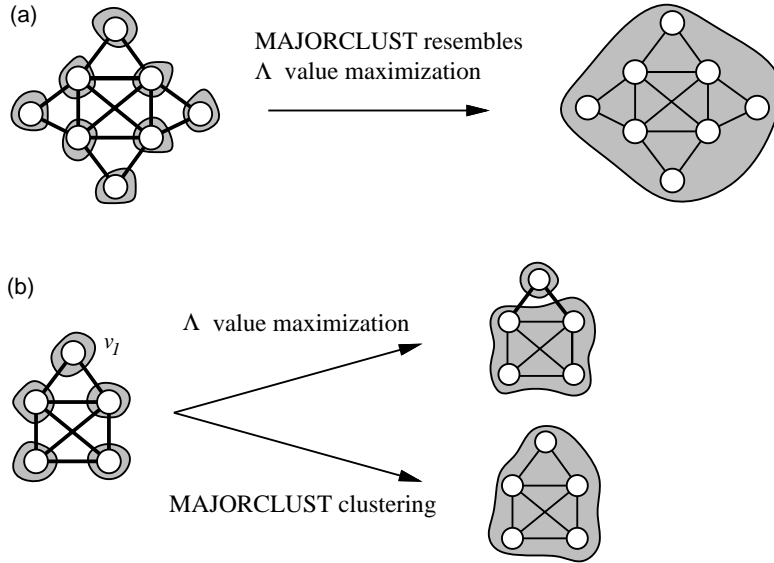


Figure 7: *The local nature of MAJORCLUST.*

4 Existing Clustering Approaches

The problem of decomposing a graph has emerged in different areas. In this section an overview and a demarcation of similar problems and their solutions will be given.

4.1 Artificial Intelligence

Clustering data and data classification have been a focus of research in the artificial intelligence community for years. Data is normally given as a set of positions in an m -dimensional Euclidean feature space, i.e. the distance between positions is well defined. Transforming data into a graph means to model positions as nodes and distances as edge capacities. Several attempts aim at detecting clusters in such a graph [24, 1, 9, 13], including threshold graphs, measures combining the number of edges within and between clusters, and the number of nodes in each clusters [1, 18], Fuzzy sets [25], minimum spanning tree, or neighborhood sets. Most approaches require domain knowledge about the graph or use more or less reasonable heuristics. Because of its theoretical foundation and its general applicableness, we concentrate here on a promising approach: clustering based on minimum cuts [12, 24].

By proving the following theorem, the idea of dividing a graph at its smallest cut is related to the structuring criterion presented in this paper.

Theorem 4.1 (Strong Splitting Condition). Applying the strong splitting condition (see Section 2) results in a decomposition at minimum cuts.

To proof this theorem we first show that $\lambda(G)$ equals the cardinality of the minimum cut of G .

Proof of Lemma. Let $\mu(G)$ denote the minimum cut of G . $\lambda(G) \leq |\mu(G)|$ because the removal of all edges belonging to the cut splits G into two components. $\lambda(G) \geq |\mu(G)|$ because in G there exists $v_1, v_2 \in V$ so that exactly $\lambda(G)$ edge disjoint paths connect

them. By removing one edge from each path, v_1 will not be connected to v_2 anymore, therefore exists a cut with $\lambda(G)$ edges.

Proof of Theorem. Let $cut(V_i, V_j)$ denote the edges between $G(V_i)$ and $G(V_j)$. From $\lambda(G) \leq \min\{\lambda_1, \dots, \lambda_r\}$ follows $|\mu(G)| \leq \min\{|\mu(G(V_1))|, \dots, |\mu(G(V_r))|\}$, i.e. no cut in $G(V_i), i = 1, \dots, r$ is smaller than $\mu(G)$. Since every cut $\mu'(G)$ except of $cut(V_1, \dots, V_r)$ decomposes at least one $G(V_i)$, $\mu'(G)$ must consist of more than $|\mu(G)|$ edges. It follows that $cut(V_1, \dots, V_r)$ must be minimum.

When the strong splitting condition does not hold, an optimum decomposition according to the structuring value need not be the same decomposition as found using the minimum cut. This is because of the latter's disregard for cluster sizes. Figure 8 is such an example. Here C_x refers to a clique with $x \geq 3$ nodes. An optimum solution according to the weighted partial connectivity Λ (which is also closer to human sense of esthetics) consists of one cluster $\{v_1, v_2, v_3, v_4\}$ and a second cluster C_x . An algorithm using the minimum cut would only separate v_1 .

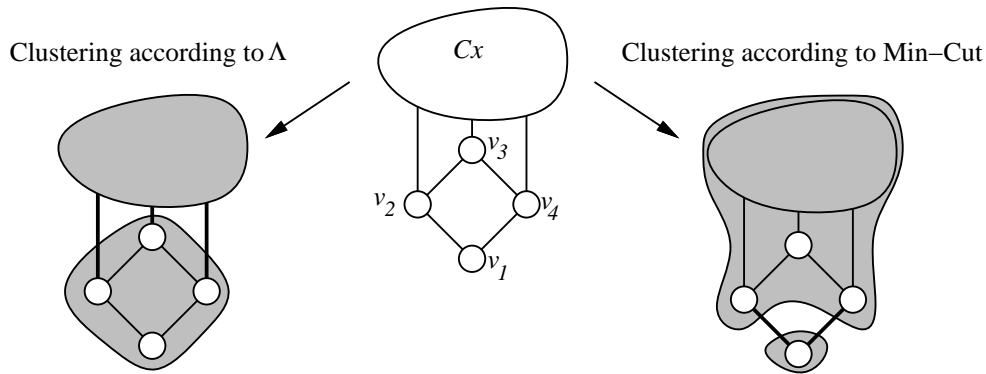


Figure 8: Weighted partial connectivity (Λ) maximization versus Min-Cut clustering.

The reader may also notice that, as mentioned before, maximizing the weighted partial connectivity implies an optimum number of clusters, while the minimum cut approach lacks any criterion for the number of necessary division steps.

4.2 Parallel Computing and Circuit Layout

Crucial to the field of parallel computing is the task of dividing a given job or process graph into k disjoint subgraphs and mapping these subgraphs onto a known processor topology (e.g. a grid). Here almost equally sized subgraphs are desirable, since this results in a balanced utilization of the processors. As edges correspond to communication between processes, minimizing the cut between subgraphs is another objective.

In connection with circuit layout a very similar problem exists: Dividing a circuit into equally sized parts while keeping a small cut between the parts obviously reduces the complexity for an automatic layout.

One of the best known formulations of a graph division problem is the so-called PARTITIONING problem. It has been used intensively in the fields of parallel processing and circuit design.

PARTITIONING.

Input. A graph $G = \langle V, E \rangle$, a vertex weight function $\beta : V \rightarrow \mathbf{N}$, an edge cost

function $\gamma : E \rightarrow \mathbf{N}$, a number $r \in \mathbf{N}$, maximum part sizes $B(i) \in \mathbf{N}, i = 1, \dots, r$, and minimum part sizes $b(i) \in \mathbf{N}, i = 1, \dots, r$.

Output. A decomposition $\mathcal{C} = (V_1, \dots, V_r)$ of V , so that:

- $\forall i = 1, \dots, r : b(i) \leq \sum_{v \in V_i} \beta(v) \leq B(i)$, and
- $\frac{1}{2} \sum_{e \in E, e=(v_i, v_j), v_i \in V_k \wedge v_j \notin V_k} \gamma(e)$ is minimum (cut minimization).

PARTITIONING is strongly NP-hard [12]. This still applies if $\beta(v) = 1 \forall v \in V$, $B(i) = |V| \forall i = 1, \dots, r$ and $b(i) = 1 \forall i = 1, \dots, r$.

The probably most successful approach for solving PARTITIONING dates from 1970 by B. W. Kernighan and S. Lin [11]. In their paper they explain:

“We can start with any arbitrary partition of the graph, calling one set “ A ” and the other “ B ” (A contains half the nodes, B the other half). If this solution isn’t already minimum-cost, we can certainly get another partition with a lower cost by moving some nodes from A to B , and others from B to A , if we choose the right ones.”

A survey of other solutions can be found in [12].

Unlike the problem treated in this paper, no existing structures are considered by this related problem. An equally sized decomposition into a predefined number of subgraphs is achieved by accepting a higher cut. This obviously reduces the applicableness of solutions to the PARTITIONING problem for detecting structures in graphs.

4.3 Graph Visualization

Dividing a graph also proved useful in the field of automatic graph layout. Efforts have been made to reduce the complexity of this problem by using Divide-and-Conquer approaches [4, 19, 18]. The authors rely on the known algorithms for PARTITIONING or exploit known features of their graphs such as biconnected components [23], circles of cliques [5], etc.

Some authors, among others [4], use mathematical definitions for the term cluster in order to find a good decomposition. Unfortunately there does not exist an agreed definition, cf. [18]:

“In spite of the differences of opinion as to what constitutes a cluster, one idea is universally accepted: the nodes belonging to a cluster must have a *strong relationship* between them in comparison with the nodes outside the cluster.”

Most functions for defining the quality of a cluster try to express this statement in mathematical terms, often by combining the demand for equally sized cluster with a small inter-cluster cut [4, 19, 18]. Unlike the structure definition given in this paper, these definitions often fail to define an optimum number of clusters, or the use of average values may lead to suboptimum clusters.

5 Application

This section outlines applications for structure identification from different fields. Note that the presented problems are typically tackled by knowledge-based concepts, and that structure identification as shown here plays the role of a knowledge preprocessor.

5.1 Monitoring

Monitoring the traffic is substantial for administrating and analyzing a modern computer network. Many tools support the recording and statistical analysis of inter-computer communications. To make use of this information the network administrator is faced with the interpretation of the so-called traffic matrix. In this matrix the amount of traffic between all pairs of computers in the network is recorded.

The identification of network parts with a high inter-node communication is interesting for planning and understanding a network. Knowing such communication structures helps the network administrator to decide e.g. whether it is reasonable to bundle such clusters in a single VLAN (virtual LAN), to change the network topology, or to upgrade to switching technology.

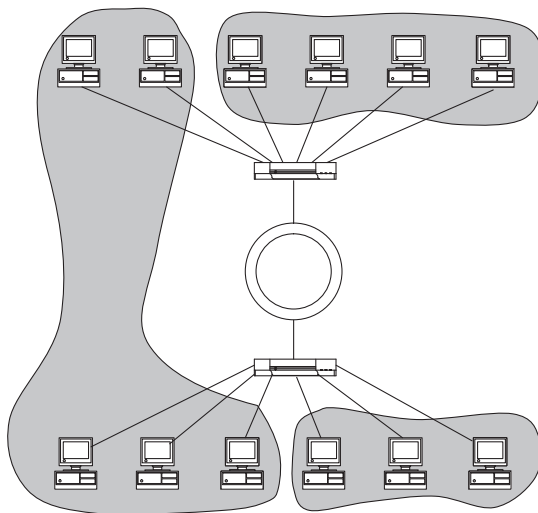


Figure 9: A computer network with clusters indicating a high amount of traffic.

Based on our structure definition, a network matrix can be evaluated by unveiling its underlying traffic structure. A prerequisite for this is a generalization of $\Lambda(\mathcal{C})$ by introducing the edge connectivity $\bar{\lambda}$ of a graph, which considers edge weights that correspond to the network traffic. In the same way the algorithm MAJORCLUST is extended: Every node v now adapts the same cluster as the *weighted* majority of its neighbors, i. e. every neighbor is weighted by the amount of its traffic with v .

Figure 9 shows a section of a local area network. Three communication clusters have been discovered, one of them comprising computers from two different subnets and thus causing high traffic on the backbone.

5.2 Visualization

Graphs proved one of the most adaptable and most frequently used means of modeling. Examples can be found in several fields, some of them mentioned in this paper. As graph size reflects the complexity of a problem, more complex problems result in larger graphs. Understanding problems and their structure turns out to be a key factor for solving them. Such an understanding can be supported by visualization, in particular by visualizing graphs that represent the interesting problem, system, or domain.

Since being a part of the real world, these graphs inherit a (problem / system / domain-) specific structure. By identifying this structure, i. e. by decomposing the graph, the complexity of an automatic graph layout can be reduced using a divide-and-conquer approach. Instead of visualizing a graph as a whole, clusters are visualized separately and connected afterwards. Because these clusters are not formed arbitrarily but reflect concepts of the domain (see section 1), the functional understanding is also furthered.

In the field of automatic graph drawing several standard algorithm exist, such as hierarchical graph layout [3, 17, 6, 20], spring embedding [15], or circles [17]. Also the use of clustering has been strongly discussed in the last years.

Having analyzed the existing approaches, we propose a visualization algorithm that is based on the idea of structure identification, and that is comprised of the following steps:

1. Structure identification = graph clustering
2. Cluster arrangement on a grid
3. Node positioning within the clusters

Step 1 is accomplished by MAJORCLUST as introduced in Section 3, step 2 is done by applying a simulated-annealing strategy, and respecting step 3, the algorithms for hierarchical graph layout have been adapted. For details concerning step 2 and 3, the reader may refer to [14].

Figure 10 shows the structured graph of a configuration knowledge base of a telecommunication system.

Figure 11 shows a simple hydraulic circuit where two clusters representing a hydraulic axis and a supply unit were detected.

6 Summary

The paper presented a new approach to quantify the structure of a graph. Following this approach, a domain, a problem, or a system can syntactically be analyzed regarding its structure—provided that a graph constitutes the adequate modeling paradigm.

The proposed structure measure, the weighted partial connectivity Λ , relies on subgraph connectivity, which is weighted with the subgraphs' sizes. The subgraphs in turn are determined by that decomposition of a graph that maximizes Λ . Hence, cluster number as well as cluster size of the structure are defined implicitly by the optimization—a characteristic which makes this approach superior to other clustering concepts. Λ maximization resembles the human sense when trying to identify a

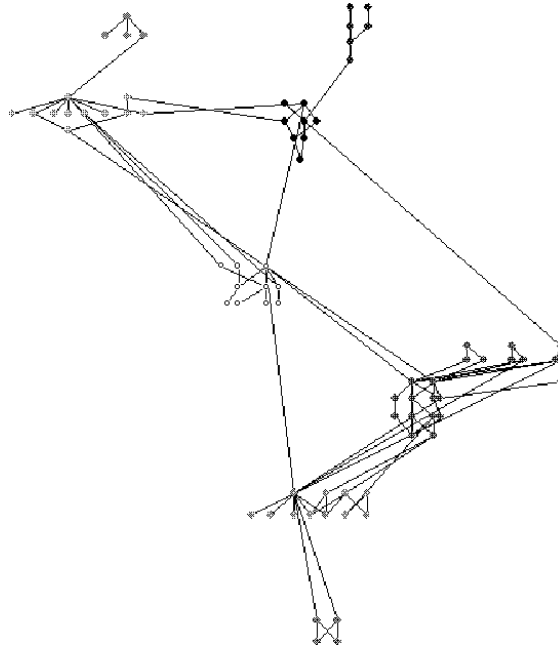


Figure 10: *Structure of a telecommunication knowledge base (screen snapshot).*

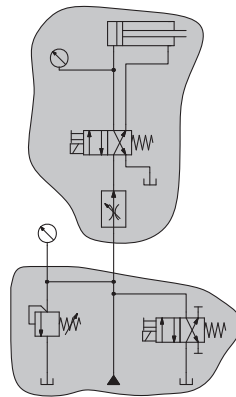


Figure 11: *Hydraulic circuit with hydraulic axis (above) and supply unit (below).*

graph's structure: Rather than searching for a given number of clusters, the density distribution of a graph's edges is analyzed.

Aside from the mathematical definition, a fast algorithm "MAJORCLUST" operationalizing Λ maximization has been developed. Applications from the field of configuration, monitoring, and visualization revealed both usability (the detected structures are reasonable) and applicability (efficient runtime behavior). Structure processing as proposed here thus provides a powerful knowledge preprocessing concept.

References

- [1] T. Bailey and J. Cowles. Cluster definition by the optimization of simple measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 1983.
- [2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 1990.
- [3] P. Eades and N. C. Wormald. Edge Crossing in Drawing of Bipartite Graphs. In *Mathematica 1994*, Springer Verlag, 1994.
- [4] L. A. R. Eli B. Messinger and R. R. Henry. A divide-and-conquer algorithm for the automatic layout of large directed graphs. *IEEE Transactions on Systems, Man, and Cybernetics*, January/February 1991.
- [5] M. H. F. J. Brandenburg and K. Skodinis. Graph Clustering: Circles of Cliques. In G. DiBattista, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer Verlag, 1997.
- [6] Erkki Mäkinen. Experiments on drawing 2-level hierarchical graphs. In *Intern. J. Computer Math. Vol. 36*, Gordon and Breach Science Publishers, 1990.
- [7] M. Heinrich and E. W. Jüngst. A Resource-based Paradigm for the Configuring of Technical Systems for Modular Components. In *Proc. CAIA '91*, pages 257–264, 1991.
- [8] T. Hesse and B. Stein. Hybrid Diagnosis in the Fluidic Domain. *Proc. EIS 98, International ICSC Symposium on Engineering of Intelligent Systems, University of La Laguna, Tenerife, Spain*, Feb. 1998.
- [9] D. K. Jitender S. Deogun and G. Steiner. An approximation algorithm for clustering graphs with dominating diametral path. *Information Processing Letters* 61, 1997.
- [10] D. Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI Wissenschaftsverlag, Wien, 1990.
- [11] B. Kernighan and S. Lin. Partitioning graphs. *Bell Laboratories Record*, January 1970.
- [12] T. Lengauer. *Combinatorial algorithms for integrated circuit layout*. Applicable Theory in Computer Science. Teubner-Wiley, 1990.
- [13] F. J. Newbery. Edge concentration: A method for clustering directed graphs. *Software Engineering Notes, ACM Press*, 14(5), 1997.
- [14] O. Niggemann, B. Stein, and M. Suermann. On Resource-based Configuration—Rendering Component-Property Graphs. In J. Sauer and B. Stein, editors, *12. Workshop “Planen und Konfigurieren”*, tr-ri-98-193, Paderborn, Apr. 1998. University of Paderborn, Department of Mathematics and Computer Science.

- [15] X. L. Peter Eades and W. F. Smyth. A fast and effective heuristic for the feedback arc set problem. In *Information Processing Letters 47*, Elsevier Science Publishers, 1993.
- [16] F. Puppe. *Problemlösungsmethoden für Expertensysteme*. Springer-Verlag, 1990.
- [17] M. G. Reggiani and F. E. Marchetti. A proposed Method for Representing Hierarchies. In *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, No.1 January*, IEEE, 1988.
- [18] T. Roxborough and Arunabha. Graph Clustering using Multiway Ratio Cut. In S. North, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer Verlag, 1996.
- [19] R. Sablowski and A. Frick. Automatic Graph Clustering. In S. North, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer Verlag, 1996.
- [20] G. Sander. Graph Layout through the VCG Tool. Technical Report A/03/94, 1994.
- [21] B. Stein. *Functional Models in Configuration Systems*. Dissertation, University of Paderborn, Department of Mathematics and Computer Science, 1995.
- [22] B. Stein and E. Vier. Computer-aided Control Systems Design for Hydraulic Drives. *Proc. CACSD 97, Gent*, Apr. 1997.
- [23] B. M. U. Dogrusoz and P. Madden. Circular Layout in the Graph Layout Toolkit. In S. North, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer Verlag, 1996.
- [24] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 1993.
- [25] J.-T. Yan and P.-Y. Hsiao. A fuzzy clustering algorithm for graph bisection. *Information Processing Letters 52*, 1994.

A Graph-theoretical Definitions

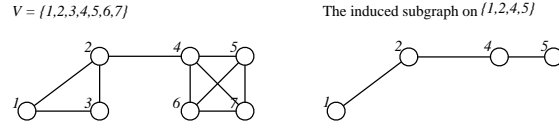
The following definitions of graph theory are adopted from [2, 12, 10]; they are used in their standard way.

1. A *graph* G is a tuple $\langle V, E \rangle$ where V is an emptyset, and $E \subseteq 2^V$ is a subset of the two-elemented subsets of V . The elements $v \in V$ are called vertices (nodes, points), the elements $e = \{v, w\} \in E$ are called edges, v, w are called *adjacent* to each other, and they are called *incident* to e .

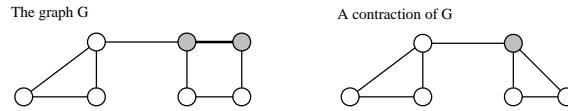
A *multigraph* allows $E \subseteq 2^V$ to be a multiset of two-elemented subsets of V . Often, a multigraph is defined as a triple $\langle V, E, g \rangle$, where $V, E \neq \emptyset$ are finite sets, $V \cap E = \emptyset$, and $g : E \rightarrow 2^V$ is a mapping with $2^V = \{U : U \subseteq V, |U| = 2\}$. g is called the incidence map.

2. A graph $H = \langle V_H, E_H \rangle$ is called *subgraph* of $G = \langle V, E \rangle$, if $V_H \subseteq V, E_H \subseteq E$.

A subgraph is called *induced subgraph* on V_H , if $E_H \subseteq E$ contains all edges whose endpoints are both in V_H . The subgraph induced on V_H is denoted with $G(V_H)$.



3. A (multi)graph $G' = \langle V', E' \rangle$ that results from $G = \langle V, E \rangle$ by identifying two points $v, w, \{v, w\} \in E$ is called *contraction* of G . $V' = V \setminus \{w\}, E' = E \setminus \{\{v, w\}\}$.



4. A tuple (e_1, \dots, e_n) is called a *walk* from v_0 to v_n , if $g(e_i) = \{v_{i-1}, v_i\}, v_i \in V, i = 1, \dots, n$.

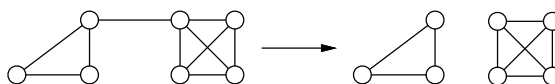
G is called *connected*, if for each two points $v_i, v_j \in V$ there is a walk from v_i to v_j .

5. $\lambda(G)$ is called the *edge connectivity* of G and is defined as follows: $\lambda(G) = \min\{|E'| : E' \subset E \text{ and } G' = \langle V, E \setminus E' \rangle \text{ is not connected}\}$.

G is called *m-fold line connected*, if $\lambda(G) \geq m$.

6. $\mathcal{C}(G) = (C_1, \dots, C_n)$ is called *decomposition* or *clustering* of G into n subgraphs induced on the C_i , if $\bigcup_{C_i \in \mathcal{C}} C_i = V, C_i \cap C_{j, j \neq i} = \emptyset$. The induced subgraphs $G(C_i)$ are called *cluster*. $E_{\mathcal{C}} \subseteq E$ consists of the set of edges between the clusters.

A graph with edge connectivity 1.



A graph with edge connectivity 3.



7. Let $G = \langle V, E \rangle$ be a graph and let $C_1, C_2 \subseteq V, C_1 \cap C_2 = \emptyset$. The *cut* between C_1, C_2 is defined as the number of edges between these sets: $cut(C_1, C_2) = |\{(v_1, v_2) : (v_1, v_2) \in E, v_1 \in C_1, v_2 \in C_2\}|$.

The *cut of a decomposition (clustering)*, $cut(\mathcal{C})$, is defined as $\sum_{C_i, C_j \in \mathcal{C}, i < j} cut(C_i, C_j)$.