On the Nature of Structure and Its Identification

Benno Stein and Oliver Niggemann

Dept. of Mathematics and Computer Science—Knowledge-based Systems,
University of Paderborn, D–33095 Paderborn, Germany
{stein,murray}@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:

1. It introduces a new and lucid structure measure, the so-called weighted partial connectivity, Λ, whose maximization defines a graph's structure (Section 2).
2. It presents a fast algorithm that approximates a graph's optimum Λ-value (Section 3).

Moreover, the proposed structure definition is compared to existing clustering approaches (Section 4), 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.

Different applications, which illustrate the usability of the measure and the algorithm, round off the paper (Section 5).

1 What Is Structure?

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

This informal definition reflects 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 structure can often be captured by a graph. Figure 1, for example, shows a gantry crane, its graph representation in the form of the component graph G, and two abstractions, say contractions of G, that can be interpreted as the crane's structure. The paper in hand is on the automatic detection of such structure information.

To allow of a more formal definition of the term structure, the following abstraction is useful.

1. The system, the "whole", is mapped onto a graph, $G = \langle V, E \rangle$. The system's elements form the set of nodes, V; the relations between the elements are represented by the set of (weighted) edges, E.

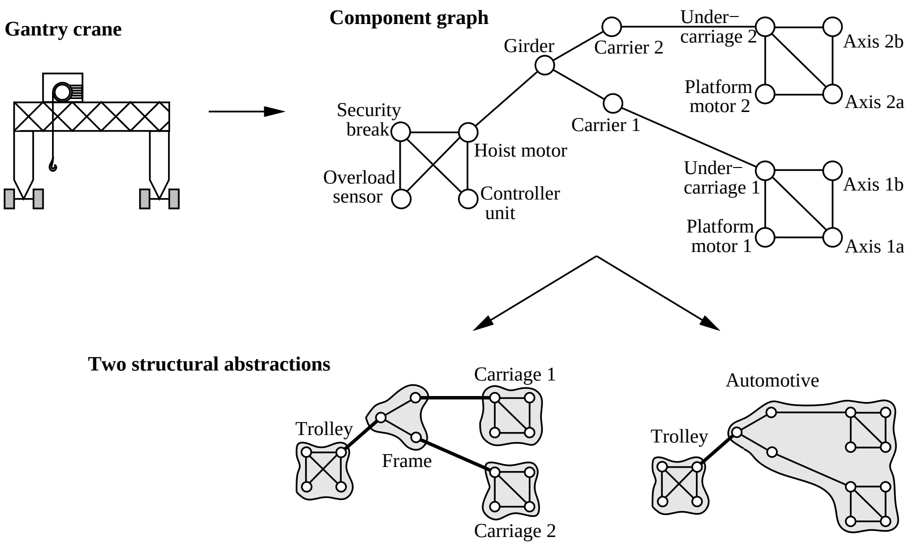


Fig. 1. Gantry crane with component graph and two structural abstractions.

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

This understanding of a system's structure relies on the following paradigms.

1. *Modular Character.* The system (say, the graph $G = \langle V, E\rangle$) can be decomposed into several modules or functions such that each element of the system (say, each node $v \in V$) belongs to exactly one module.
2. *Connectivity.* Modules are defined implicitly, merely exploiting the graph-theoretical concept of connectivity: The connectivity between nodes assigned to the same module is assumed to be higher than the connectivity between any two nodes from two different modules.
3. *Contraction.* The system's structure is the contraction of G where a single node is substituted for all nodes belonging to the same module.

Remarks. Point 1 reflects hierarchy or decentralization aspects of a system or an organization. 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 simple abstraction.

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.

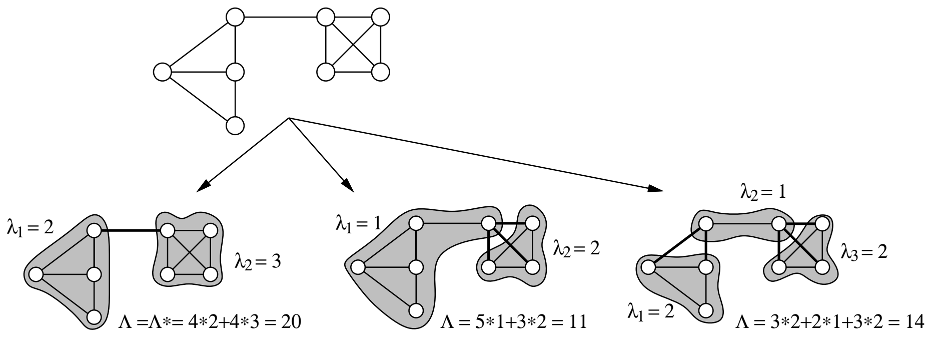


Fig. 2. Graph decompositions and related Λ values.

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]

1. $\mathcal{C}(G) = (C_1, \ldots, C_n)$ is a *decomposition* of G into n subgraphs induced on the C_i, if $\bigcup_{C_i \in \mathcal{C}} = V$ 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* $\lambda(G)$ of a graph G denotes the minimum number of edges that must be removed to make G a not-connected graph: $\lambda(G) = \min\{|E^{'}| : E^{'} \subset E$ and $G^{'} = \langle V, E \setminus E'\rangle$ is not connected$\}$.

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

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

$\lambda(C_i) \equiv \lambda_i$ designates the edge connectivity of $G(C_i)$.
Figure 2 illustrates the weighted partial connectivity Λ.

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.

[1] Concepts and definitions of graph theory are used in their standard way; they are adopted from [10,7].

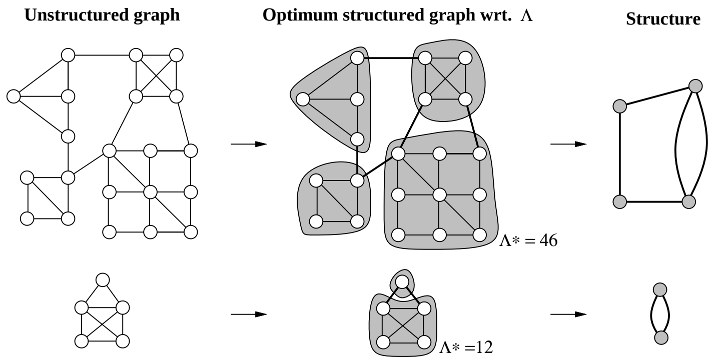


Fig. 3. Examples for decomposing a graph according to our structure definition.

Remarks. A key feature of the above structure measure is its *implicit definition of a structure's number of clusters.*

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

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

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

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 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

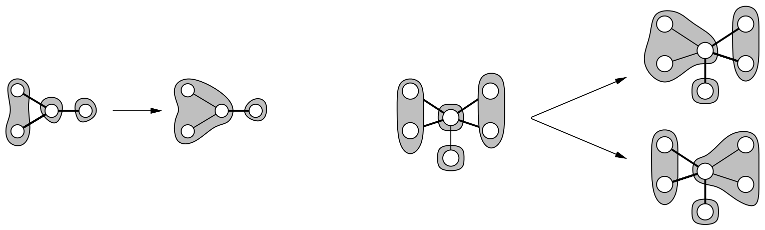


Fig. 4. A definite majority clustering situation (left) and an undecided majority clustering situation (right).

of its neighbors belong to. If there exist several such clusters, one of them is chosen randomly. If re-clustering comes to an end, the algorithm terminates.

The left hand side of 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 depicted on the right hand side, the central node has the choice between the left and the right cluster.

We now write down this algorithm formally.

MAJORCLUST.
Input. A graph $G = \langle V, E \rangle$.
Output. A function $c : V \mapsto \mathbf{N}$, 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 $\left|\{u : \{u, v\} \in E \wedge c(u) = i\}\right|$ is max.
(7)         if $c(v) \neq c^*$ then $c(v) = c^*, t = false$
(8)     end
(9) end
```

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 sub-optimality. In every step only a node's neighbors are considered, resulting in an excellent runtime behavior. On the other hand, by disregard-

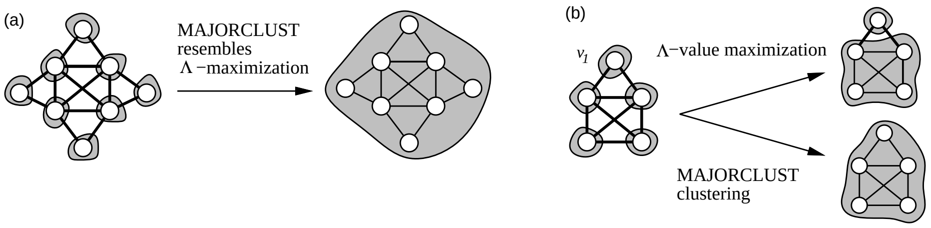


Fig. 5. The local behavior of MAJORCLUST may lead to sub-optimum Λ values.

ing global criteria like the connectivity, MAJORCLUST cannot always find the optimum solution. Figure 5 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.

3.1 Extension for Weighted Graphs

It is both useful and obvious to extend our structure identification approach by introducing edge weights. The amount of the weight $w(e)$ models the importance of an edge e. A prerequisite for this is a generalization of $\Lambda(\mathcal{C})$ by introducing the *weighted edge connectivity* $\bar{\lambda}$ of a graph as follows:
$\tilde{\lambda}(G) = \min\{\sum_{e \in E'} w(e) : E' \subset E$ and $G' = \langle V, E \setminus E' \rangle$ is not connected$\}$. Using this definition all results from Section 2 can be directly extended to graphs with edge weights.

In the same way the algorithm MAJORCLUST is altered: Every node v now adapts the same cluster as the *weighted* majority of its neighbors, i. e. every neighbor counts according to the weight (i.e. importance) of the edge connecting it to v.

4 Existing Clustering Approaches

Clustering data given as graphs has been a focus of research for years. The existing approaches can be classified as follows.

Hierarchical versus Non-hierarchical Algorithms. Hierarchical algorithms create a tree of node subsets by successively subdividing or merging the graph's nodes. In order to obtain a unique clustering, a second step is necessary that prunes this tree at adequate places.

Hierarchical algorithms can be further classified into divisive and agglomerative approaches. Divisive algorithms start with each vertex being its own cluster and union clusters iteratively. For agglomerative algorithms on the other hand, the entire graph initially forms one single cluster which is successively subdivided. Examples for divisive algorithms are Min-cut-clustering [10,20] or dissimilarity-based algorithms e. g. [11]. Typical agglomerative algorithms are k-nearest-neighbor or linkage methods [4,16,6].

Non-Hierarchical algorithms subdivide the graph into clusters within one step. Examples are clustering techniques based on Minimal-Spanning-Trees [22], self-organizing Kohonen networks [9] or approaches which optimize a given goal criterion [1,13,14,13].

Exclusive versus Non-exclusive Algorithms. Exclusive clustering algorithms assign every node to exactly one cluster, while non-exclusive algorithms assign to a node a membership value respecting each cluster. The algorithms mentioned above are of exclusive type; an example for a non-exclusive algorithm is Fuzzy clustering [21].

Clustering versus Partitioning. The clustering algorithms described above do not impose any constraint on cluster sizes. Partitioning algorithms as used in the fields of parallel computing or VLSI design typically demand homogeneous cluster sizes. Examples for partitioning algorithms can be found in [10,8].

Λ-maximization and MAJORCLUST can be classified as non-hierarchical and exclusive. MAJORCLUST finds a fast, but possibly suboptimal solution for the problem of Λ-maximization. I. e., unlike most optimization approaches, Λ-maximization as performed by MAJORCLUST does not rely on slow optimization techniques and can be used for large graphs.

The clustering quality of the Λ criterion and the MAJORCLUST algorithm will be illustrated by the following two comparisons with well-known clustering techniques as well as by different applications in Section 5.

4.1 Clustering Based on the Minimum Cut

MAJORCLUST is a divisive approach, that recursively subdivides a graph at its smallest cut. The following theorem relates Min-cut-clustering to clustering by means of Λ-maximization.

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

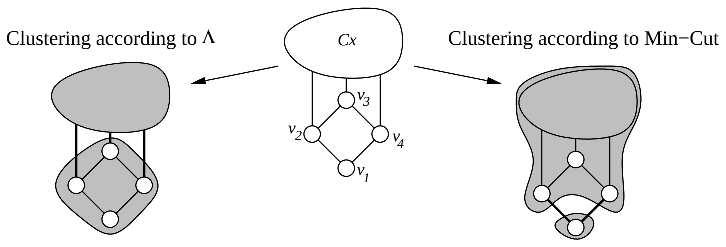


Fig. 6. Weighted partial connectivity (Λ-) maximization versus Min-Cut-clustering.

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 6 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.

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 Clustering Based on Nearest-Neighbor Strategies

Nearest-Neighbor clustering is an agglomerative approach that iteratively merges the two closest clusters. Its widespread use results in several variations [3,4,16,6]. The following qualitative comparison to MAJORCLUST does not take all existing variations into consideration, but we claim that Λ-maximization and MAJORCLUST respectively can indeed overcome typical problems inherent to Nearest-Neighbor clustering concepts.

1. Nearest-Neighbor clustering, like all hierarchical algorithms, does not define the (optimal) number of clusters. Λ-maximization (as well as MAJORCLUST) implicitly defines both number and sizes of the clusters.
2. The greedy nature of Nearest-Neighbor methods (unlike as in MAJORCLUST, nodes are never reassigned to another cluster) leads to the so-called chaining effect [3], as illustrated in Figure 7.
3. The step of transforming the tree as created by a Nearest-Neighbor algorithm into a unique clustering often depends on extra parameters such as the minimum cluster or vertex distance. This results in difficulties if clusters have strongly varying point densities or inter-cluster distances. Λ-maximization and MAJORCLUST do not depend on additional parameters and behave more sensible in such clustering situations.

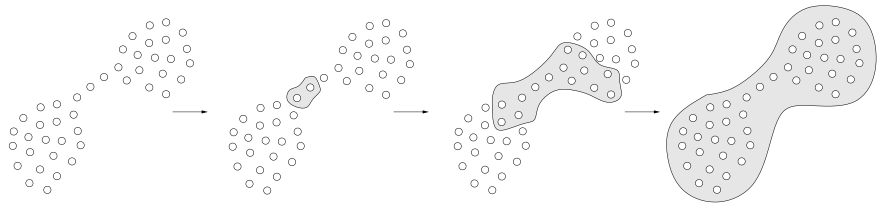


Fig. 7. The (undesired) chaining behavior of Nearest-Neighbor methods.

4. Nearest-Neighbor methods rely on distance information only. They thus disregard connectivity information. For weighted graphs this may lead to clusters which lack the human sense of esthetics, for unweighted graphs this behavior may result in a failure to find any clusters.

5 Application

This section outlines three applications for structure identifications.[2]

5.1 Monitoring Computer Networks

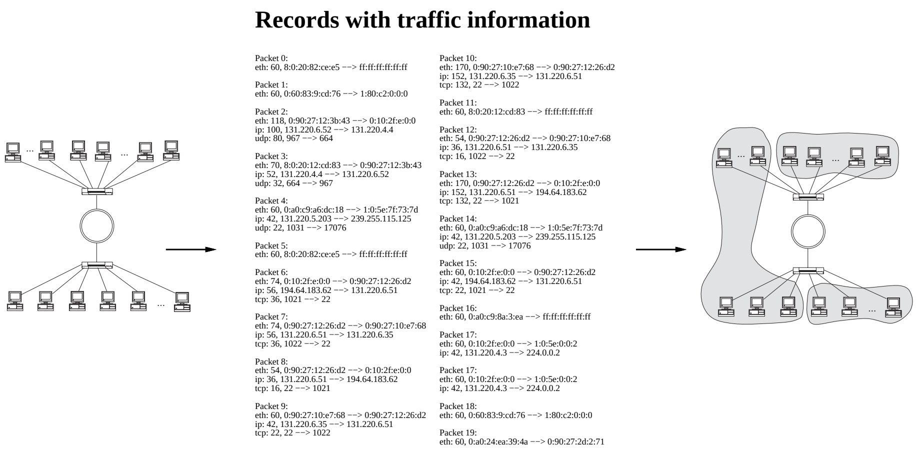
Monitoring traffic, i. e. recording inter-computer communications, is substantial for administrating and analyzing computer networks. The amount of traffic between all pairs of computers in the network is recorded in the so-called traffic matrix. By interpreting the traffic matrix as the adjacency matrix of a weighted graph, cluster identification techniques can be applied to the problem of traffic analysis. Figure 8 illustrates the procedure.

Being faced with rather large traffic matrices ($>$ 400 nodes, $>$ 800 edges), human experts need to fall back upon computer support for cluster identification. However, this clustering problem is difficult to solve since no features about communication structures are known beforehand, which makes this problem an ideal testbed for MAJORCLUST.

Cooperations with network experts allowed the application of MAJORCLUST under realistic conditions. MAJORCLUST revealed several interesting structures in traffic matrixes: (*i*) subnets were subdivided according to main applications, (*ii*) project member in different subnets were identified, and (*iii*) computer serving similar purposes were clustered together.

These insights are helpful in several ways. Firstly, they render a general understanding of traffic structure possible, e. g. clusters combining computers in different subnets mean high traffic on the backbone. Secondly, they provide additional information for planning tasks in the form of modification hints for the network architecture.

[2] Aside from the presented applications, structure identification has been investigated for the preprocessing of configuration knowledge bases and the topological analysis of fluidic systems [5,18].



```
Packet 0:
eth: 60, 8:0:20:82:ce:e5 --> ff:ff:ff:ff:ff:ff

Packet 1:
eth: 60, 0:60:83:9:cd:76 --> 1:80:c2:0:0:0

Packet 2:
eth: 118, 0:90:27:12:3b:43 --> 0:10:2f:e:0:0
ip: 100, 131.220.6.52 --> 131.220.4.4
udp: 80, 967 --> 664

Packet 3:
eth: 70, 8:0:20:12:cd:83 --> 0:90:27:12:3b:43
ip: 52, 131.220.4.4 --> 131.220.6.52
udp: 32, 664 --> 967

Packet 4:
eth: 60, 0:a0:c9:a6:dc:18 --> 1:0:5e:7f:73:7d
ip: 42, 131.220.5.203 --> 239.255.115.125
udp: 22, 1031 --> 17076

Packet 5:
eth: 60, 8:0:20:82:ce:e5 --> ff:ff:ff:ff:ff:ff

Packet 6:
eth: 74, 0:10:2f:e:0:0 --> 0:90:27:12:26:d2
ip: 56, 194.64.183.62 --> 131.220.6.51
tcp: 36, 1021 --> 22

Packet 7:
eth: 74, 0:90:27:12:26:d2 --> 0:90:27:10:e7:68
ip: 56, 131.220.6.51 --> 131.220.6.35
tcp: 36, 1022 --> 22

Packet 8:
eth: 54, 0:90:27:12:26:d2 --> 0:10:2f:e:0:0
ip: 36, 131.220.6.51 --> 194.64.183.62
tcp: 16, 22 --> 1021

Packet 9:
eth: 60, 0:90:27:10:e7:68 --> 0:90:27:12:26:d2
ip: 42, 131.220.6.35 --> 131.220.6.51
tcp: 22, 22 --> 1022

Packet 10:
eth: 170, 0:90:27:10:e7:68 --> 0:90:27:12:26:d2
ip: 152, 131.220.6.35 --> 131.220.6.51
tcp: 132, 22 --> 1022

Packet 11:
eth: 60, 8:0:20:12:cd:83 --> ff:ff:ff:ff:ff:ff

Packet 12:
eth: 54, 0:90:27:12:26:d2 --> 0:90:27:10:e7:68
ip: 36, 131.220.6.51 --> 131.220.6.35
tcp: 16, 1022 --> 22

Packet 13:
eth: 170, 0:90:27:12:26:d2 --> 0:10:2f:e:0:0
ip: 152, 131.220.6.51 --> 194.64.183.62
tcp: 132, 22 --> 1021

Packet 14:
eth: 60, 0:a0:c9:a6:dc:18 --> 1:0:5e:7f:73:7d
ip: 42, 131.220.5.203 --> 239.255.115.125
udp: 22, 1031 --> 17076

Packet 15:
eth: 60, 0:10:2f:e:0:0 --> 0:90:27:12:26:d2
ip: 42, 194.64.183.62 --> 131.220.6.51
tcp: 22, 1021 --> 22

Packet 16:
eth: 60, 0:a0:c9:8a:3:ea --> ff:ff:ff:ff:ff:ff

Packet 17:
eth: 60, 0:10:2f:e:0:0 --> 1:0:5e:0:0:2
ip: 42, 131.220.4.3 --> 224.0.0.2

Packet 17:
eth: 60, 0:10:2f:e:0:0 --> 1:0:5e:0:0:2
ip: 42, 131.220.4.3 --> 224.0.0.2

Packet 18:
eth: 60, 0:60:83:9:cd:76 --> 1:80:c2:0:0:0

Packet 19:
eth: 60, 0:a0:24:ea:39:4a --> 0:90:27:2d:2:71
```

Fig. 8. Communication clusters can be identified in the traffic matrix of a network.

5.2 Visualizing Knowledge Bases

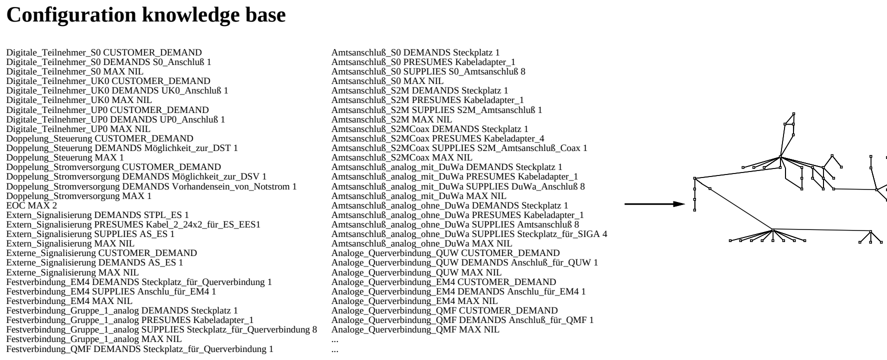
Automatic graph visualization is a key problem when supporting human understanding of complex data structures. To reduce the complexity of the visualization problem, one strategy is to apply a Divide-and-Conquer approach [2,14].

The role of clustering in this connection is to tackle the divide task, i. e., to break down a large graph into useful subgraphs. By a second step the resulting clusters are arranged on a grid, and by a third step the nodes within each cluster are positioned.

We have operationalized and applied this concept for the analysis and visualization of resource-based configuration knowledge bases. A resource-based knowledge base textually describes the configuration objects by tuples comprising an object's supplied and demanded functions. From such a description a global overview can be created that envisions the closely connected modules and their functional interplay. Figure 9 shows a part of a visualized knowledge-base for the resource-based configuration of telecommunication system. Details and related information can be found in [12,17,19,15,2].

5.3 Clustering Metric Data

Data for clustering is often given as positions in a metric space, which can canonically be transfered into a graph. Based on this graph, the structure measure Λ, operationalized in the form of MAJORCLUST, can be applied for clustering. Figure 10 shows a set of points and the identified clusters. Recall that MAJORCLUST did not need meta information about the number and the size of the clusters. Input for MAJORCLUST was the totally connected graph of points. However, instead of connecting all pairs of vertices, connecting a vertex solely to its nth closest



Configuration knowledge base

```
Digitale_Teilnehmer_S0 CUSTOMER_DEMAND
Digitale_Teilnehmer_S0 DEMANDS S0_Anschluß 1
Digitale_Teilnehmer_S0 MAX NIL
Digitale_Teilnehmer_UK0 CUSTOMER_DEMAND
Digitale_Teilnehmer_UK0 DEMANDS UK0_Anschluß 1
Digitale_Teilnehmer_UK0 MAX NIL
Digitale_Teilnehmer_UP0 CUSTOMER_DEMAND
Digitale_Teilnehmer_UP0 DEMANDS UP0_Anschluß 1
Digitale_Teilnehmer_UP0 MAX NIL
Doppelung_Steuerung CUSTOMER_DEMAND
Doppelung_Steuerung DEMANDS Möglichkeit_zur_DST 1
Doppelung_Steuerung MAX 1
Doppelung_Stromversorgung CUSTOMER_DEMAND
Doppelung_Stromversorgung DEMANDS Möglichkeit_zur_DSV 1
Doppelung_Stromversorgung DEMANDS Vorhandensein_von_Notstrom 1
Doppelung_Stromversorgung MAX 1
EOC MAX 2
Extern_Signalisierung DEMANDS STPL_ES 1
Extern_Signalisierung PRESUMES Kabel_2_24x2_für_ES_EES1
Extern_Signalisierung SUPPLIES AS_ES 1
Extern_Signalisierung MAX NIL
Externe_Signalisierung CUSTOMER_DEMAND
Externe_Signalisierung DEMANDS AS_ES 1
Externe_Signalisierung MAX NIL
Festverbindung_EM4 DEMANDS Steckplatz_für_Querverbindung 1
Festverbindung_EM4 SUPPLIES Anschlu_für_EM4 1
Festverbindung_EM4 MAX NIL
Festverbindung_Gruppe_1_analog DEMANDS Steckplatz 1
Festverbindung_Gruppe_1_analog PRESUMES Kabeladapter_1
Festverbindung_Gruppe_1_analog SUPPLIES Steckplatz_für_Querverbindung 8
Festverbindung_Gruppe_1_analog MAX NIL
Festverbindung_QMF DEMANDS Steckplatz_für_Querverbindung 1
Amtsanschluß_S0 DEMANDS Steckplatz 1
Amtsanschluß_S0 PRESUMES Kabeladapter_1
Amtsanschluß_S0 SUPPLIES S0_Amtsanschluß 8
Amtsanschluß_S0 MAX NIL
Amtsanschluß_S2M DEMANDS Steckplatz 1
Amtsanschluß_S2M PRESUMES Kabeladapter_1
Amtsanschluß_S2M SUPPLIES S2M_Amtsanschluß 1
Amtsanschluß_S2M MAX NIL
Amtsanschluß_S2MCoax DEMANDS Steckplatz 1
Amtsanschluß_S2MCoax PRESUMES Kabeladapter_4
Amtsanschluß_S2MCoax SUPPLIES S2M_Amtsanschluß_Coax 1
Amtsanschluß_S2MCoax MAX NIL
Amtsanschluß_analog_mit_DuWa DEMANDS Steckplatz 1
Amtsanschluß_analog_mit_DuWa PRESUMES Kabeladapter_1
Amtsanschluß_analog_mit_DuWa SUPPLIES DuWa_Anschluß 8
Amtsanschluß_analog_mit_DuWa MAX NIL
Amtsanschluß_analog_ohne_DuWa DEMANDS Steckplatz 1
Amtsanschluß_analog_ohne_DuWa PRESUMES Kabeladapter_1
Amtsanschluß_analog_ohne_DuWa SUPPLIES Amtsanschluß 8
Amtsanschluß_analog_ohne_DuWa SUPPLIES Steckplatz_für_SIGA 4
Amtsanschluß_analog_ohne_DuWa MAX NIL
Analoge_Querverbindung_QUW CUSTOMER_DEMAND
Analoge_Querverbindung_QUW DEMANDS Anschluß_für_QUW 1
Analoge_Querverbindung_QUW MAX NIL
Analoge_Querverbindung_EM4 CUSTOMER_DEMAND
Analoge_Querverbindung_EM4 DEMANDS Anschlu_für_EM4 1
Analoge_Querverbindung_EM4 MAX NIL
Analoge_Querverbindung_QMF CUSTOMER_DEMAND
Analoge_Querverbindung_QMF DEMANDS Anschluß_für_QMF 1
Analoge_Querverbindung_QMF MAX NIL
...
...
```

Fig. 9. Part of a configuration knowledge base with analyzed and visualized structure.

neighbors improves the performance. Both the quality of the clusters found and the runtime needed by MAJORCLUST have been examined using our VisioDat tool.

6 Summary

The paper presented a new approach to quantify the structure of graphs. 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 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, the fast algorithm MAJORCLUST operationalizing Λ-maximization has been developed. Applications from the fields

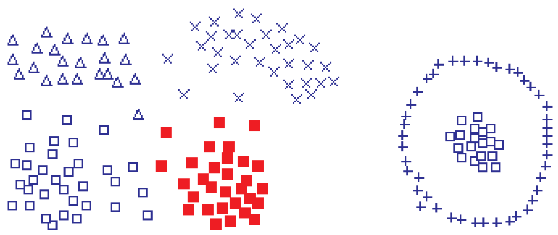


Fig. 10. Clusters in a set of points in a metric space.

of monitoring, visualization, and configuration 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. 128
2. 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. 131, 131
3. B. S. Everitt. Cluster analysis. *Edward Arnold, a division of Hodder & Stoughton*, 1992. 129, 129
4. K. Florek, J. Lukaszewiez, J. Perkal, H. Steinhaus and S. Zubrzchi. Sur la liason et la division des points d'un ensemble fini. *Colloqium Mathematicum*, 1951. 128, 129
5. 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. 130
6. S. C. Johnson. Hierarchical clustering schemes. *Psychometrika 32*, 1967. 128, 129
7. D. Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI Wissenschaftsverlag, 1990. 124
8. B. Kernighan and S. Lin. Partitioning graphs. *Bell Laboratories Record*, January 1970. 128
9. T. Kohonen. Self Organizing and Associate Memory. *Springer-Verlag*, 1990. 128
10. T. Lengauer. *Combinatorical algorithms for integrated circuit layout*. Applicable Theory in Computer Science. Teubner-Wiley, 1990. 124, 127, 128
11. P. MacNaughton-Smith, W.T. Williams, M.B. Dale and L.G. Mockett. Dissimilarity analysis. *Nature 202*, 1964. 128
12. 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. 131
13. T. Roxborough and Arunabha. Graph Clustering using Multiway Ratio Cut. In S. North, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer Verlag, 1996. 128, 128
14. R. Sablowski and A. Frick. Automatic Graph Clustering. In S. North, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer Verlag, 1996. 128, 131
15. G. Sander. Graph Layout through the VCG Tool. Technical Report A/03/94, 1994. 131
16. P. H. A. Sneath. The application of computers to taxonomy. *J. Gen. Microbiol. 17*, 1957. 128, 129
17. B. Stein. *Functional Models in Configuration Systems*. Dissertation, University of Paderborn, Department of Mathematics and Computer Science, 1995. 131
18. B. Stein and E. Vier. Computer-aided Control Systems Design for Hydraulic Drives. *Proc. CACSD 97, Gent*, Apr. 1997. 130
19. K. Sugiyama, S. Tagawa, and M. Toda. Methods for Visual Understandig of Hierarchical System Structures. *IEEE Transactions on Systems, Man, and Cybernectics, Vol. SMC-11, No. 2*, 1981. 131

20. 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. 127
21. J.-T. Yan and P.-Y. Hsiao. A fuzzy clustering algorithm for graph bisection. *Information Processing Letters 52*, 1994. 128
22. C. T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on computers Vol. C-20, No. 1*, 1971. 128