The Dynamic Graph Clustering Problem - ILP-Based Approaches Balancing Optimality and the Mental Map

Thesis

submitted by

Florian Hübner

INSTITUT FÜR THEORETISCHE INFORMATIK
UNIVERSITÄT KARLSRUHE (TH)
2008

Supervised by

Dipl.-Math. techn. Robert Görke
Prof. Dr. Dorothea Wagner

Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet.

Karlsruhe, 13. Mai 2008

Zusammenfassung

Clustering ist ein etabliertes Verfahren zur Analyse von Netzwerken bzw. netzwerkartigen Daten. Die Unterteilung des Graphen eines Netzwerks in so genannte *Cluster* soll seine Funktion verdeutlichen, sowie Gemeinsamkeiten von Mengen von Knoten und Eigenschaften von einzelnen Knoten aufzeigen. Methodisch geht es beim Clustering eines Graphen darum, seine Knotenmenge derart zu partitionieren, daß die Knoten jeder Partition viele Verbindungen untereinander jedoch wenige Verbindungen zu anderen Partitionen besitzen.

Obwohl viele natürliche Netzwerke stetiger Veränderung unterliegen, wurde das Clustering von *dynamischen Graphen* (d.h. Graphen, deren Knoten- und Kantenmenge sich im Laufe der Zeit verändert) bisher nicht detailliert betrachtet. Ein dynamischer Graph $\mathcal{G}$ wird in der vorliegenden Arbeit als eine geordnete Menge von gewöhnlichen Graphen $\{G_1, \ldots, G_n\}$ betrachtet. Das *Dynamic Graph Clustering Problem* besteht darin, zu jedem dieser gewöhnlichen Graphen G_i ein Clustering $\mathcal{C}_{G_i}$ zu finden, welches von maximaler Qualität ist und gleichzeitig maximale Ähnlichkeit mit dem Clustering $\mathcal{C}_{G_{i-1}}$ des vorhergehenden Graphen G_{i-1} aufweist. Dies soll die *Mental Map* des Clusterings erhalten und eine Analyse der strukturellen Veränderungen im Verlauf der Zeit ermöglichen. Die Qualität eines Clusterings wird dabei mittels der bekannten Indizes *Modularity* und *Significance Performance* geprüft, als Maß für die Ähnlichkeit zweier Clusterings werden ebenfalls bekannte Abstandsmaße verwendet.

Im Zuge dieser Arbeit wird eine existierende Formulierung eines ganzzahligen linearen Programms (ILP) zur Berechnung von Clusterings mit optimalen Modularity Werten analysiert und in Bezug auf die zur Lösung benötigte Zeit optimiert. Außerdem wird ein analoges ILP für den Significance Performance Index erstellt und mit dem Original verglichen. Dabei wird die Entscheidungsäquivalenz der beiden Indizes gezeigt.

Zur Berechnung von dynamischen Clusterings welche die Problemstellung erfüllen, wird ein bi-kriterielles ILP entworfen, welches in der Lage ist, eine gewichtete Kombination aus Qualitätsindex und Abstandsmaß gleichzeitig zu optimieren. Somit werden beide Kriterien explizit berücksichtigt. Um die Möglichkeiten dieses Ansatzes zu untersuchen, wird ein entsprechendes ILP basierend auf Modularity und dem *Rand Abstandsmaß* formuliert und mit einem echten dynamischen Graphen getestet. Dabei werden unterschiedliche Gewichtungen bezüglich der beiden Kriterien berücksichtigt, sowie die Auswirkungen unterschiedlich großer Veränderungen im Graphen getestet. Es wird der Zusammenhang zwischen Qualität eines Clusterings und Abstand zu seinem Vorgänger verdeutlicht, sowie die Möglichkeiten des Anwenders, eine Balance zwischen den Kriterien zu steuern.

Ein zweiter Ansatz zielt darauf ab, die Qualität eines Clusterings explizit und den Abstand zu einem vorherigen Clustering implizit zu optimieren. Die Formulierung eines partiellen ILPs basiert auf der Annahme, daß minimale Veränderungen in einem Graphen semantisch hauptsächlich von lokaler Bedeutung sind. In Folge einer atomaren Modifikation des Graphen G_{i-1} zu G_i wird eine mittels verschiedener Strategien gefundene Umgebung aus $\mathcal{C}_{G_{i-1}}$ extrahiert und mit Hilfe des mathematischen Programms unter Berücksichtigung der noch bestehenden Cluster neu strukturiert. Die nicht aufgelösten Teile des Clusterings bleiben unverändert. Der Vorteil des Verfahrens liegt darin, daß die Komplexität des ILPs weitgehend unabhängig von der Größe des zu clusternden Graphen ist. Die Clustering Qualität der durch das partielle ILP gefundenen dynamischen Clusterings wird mit Ergebnissen von Newman's Greedy Algorithmus verglichen. Es wird demonstriert, daß das partielle ILP stets Clusterings findet, die eine mindestens gleichwertige Qualität aufweisen. Dazu werden sowohl statische, generierte Clustergraphen sowie Clustergraphen mit bekannten optimalen Indexwerten pseudodynamisch aufgebaut. Außerdem wird anhand eines echten dynamischen Graphen die Performance des partiellen ILP in Bezug auf die Ähnlichkeit zwischen aufeinander folgenden Clusterings gezeigt, sowie seine Fähigkeit, mit allen möglichen Veränderungen im Graphen (Einfügen und Löschen von Knoten und Kanten, Erhöhung und Verringerung von Kantengewichten) umzugehen.

Abstract

Clustering is an established tool for the analysis of networks or network-like data. The partitioning of the graph of a network into so-called *clusters* is meant to yield insights into its function, and to reveal common properties amongst nodes, as well as properties of individual nodes. A cluster is understood to be a subset of the nodes of a network with large density of links amongst them and sparse connectivity to other clusters.

Though lots of natural networks are subject to changes, the clustering of *dynamic graphs* (i.e. graphs, whose set of nodes and set of edges change over time) has not been studied in detail yet. In this thesis, a dynamic graph is understood to consist of a sequence of ordinary graphs $\{G_1, \ldots, G_n\}$. In brief, the *Dynamic Graph Clustering Problem* is to find a clustering $\mathcal{C}_{G_i}$ for each of those graphs G_i, such that the clustering's quality is maximized and the distance to the clustering $\mathcal{C}_{G_{i-1}}$ of the previous graph G_{i-1} is minimized. The structural proximity to its predecessor is meant to conserve the *mental map* of the clustering, and thus to allow for an analysis of the structural changes in the *dynamic clustering* (i.e. a sequence of clusterings) over time. The quality of a clustering is examined by the application of the well known clustering indices *modularity* and *significance performance*. The structural distance between consecutive clusterings is evaluated by also well known distance measures for clusterings.

During the course of this thesis, an existing integer linear program (ILP) formulation for finding optimal modularity clusterings is analyzed and optimized, in order to allow for the faster processing of larger graphs. Furthermore, an ILP formulation of the significance performance index is developed and compared to the modularity version. It is shown that both rank clusterings of a graph identically and thus result in the same optimal results.

A bi-criterial ILP for solving the Dynamic Graph Clustering Problem is designed, which allows for the optimization of a weighted combination of a quality index and a distance measure at the same time. Both criteria are considered explicitly. In order to study the capabilities of the approach, a proof of concept employing the modularity index and the Rand distance measure is formulated and tested with a real dynamic graph. The influence of different weighting factors and different numbers of atomary modifications in the graph's structure is examined. The relationship between a clustering's quality and the distance to its predecessor is shown, as well as the user's ability to balance their optimization.

In addition to that, a complementary partial ILP is designed, which explicitly optimizes the clusterings' quality while minimizing the distance between consecutive clusterings implicitly. The mathematical program is based on the assumption that minimal changes in the graph affect its semantics primarily locally. As a consequence of the modifications between two graphs G_{i-1} and G_i, a neighborhood of nodes is extracted from the clusters of $\mathcal{C}_{G_{i-1}}$ and re-assigned by the partial ILP according to the new structure of the graph. The preserved clusters stay unchanged. With respect to the remaining parts of the previous clustering and the modularity index, the result is optimal. The complexity of the constructed ILP is mostly independent from the size of the underlying graph, therefore graphs with arbitrary size can potentially be processed. The quality of the found clusterings is compared to the quality of clusterings found by Newman's greedy algorithm, and it is demonstrated, that the partial ILP's clusterings' quality is at least as good. This is confirmed by clustering generated graphs as well as common example graphs with known optimal modularity values incrementally. Additionally, the performance with respect to the distance between consecutive clusterings is shown by calculating a dynamic clustering for a real dynamic graph. The ability of the partial ILP to cope with all possible sorts of modifications (i.e. the insertion and deletion of nodes and edges, changes in existing edges weights) is demonstrated.

Contents

1 Introduction

1.1 Motivation

Clustering is a frequently used tool in the analysis of networks or network-like data. For example, within Figure 1.1 the nodes of the network represent members of a karate club and the edges represent the social interactions amongst them. The data was compiled over the course of two years in the late 1970s by Wayne Zachary ([34]). A dispute arose between the club's administrator and its principal karate teacher over whether to raise club fees or not. The conflict lead to a split of the club and two groups around the administrator and the karate teacher formed. The grouping of the network shows this split. Without knowing details about the actual event, the usage of clustering techniques allows for the analysis of the social contacts and thus the prediction of the new formation of club members. When applying the well known modularity index (see Section 2.3.1) to find a decomposition of the network into smaller subgroups with strong interaction between their members, four groups are identified. These groups are marked by colors and correspond to the actual split.

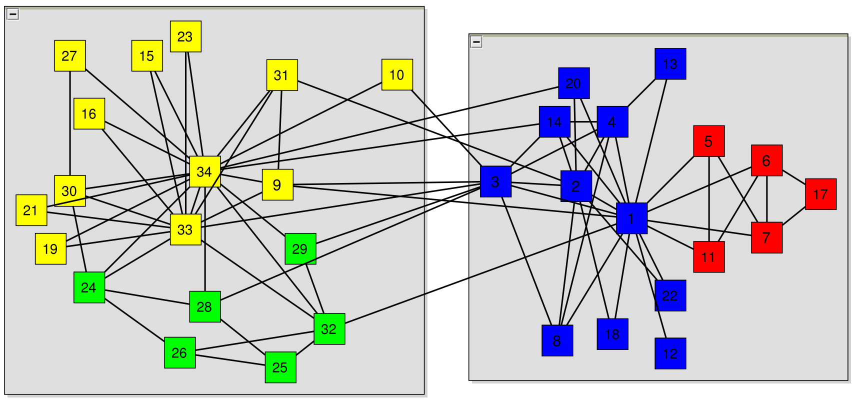

Figure 1.1: The network shows the social interactions amongst members of a Karate Club. The grouping corresponds to a split that happened due to a dispute between the club's administrator and its principal karate teacher. The coloring depicts a clustering with optimal modularity.

In general such networks consist of a set of objects and a relation between them. They span over many disciplines. In biology the clustering of proteins is used to indentify structural similarities (e.g. [1]), it helps optimizing transportation schedules and networks (e.g. [17]), in communication networks efficient routing is achieved by identifying inter-cluster connections (e.g. [22]), and of course the analysis of social relations is supported by decomposing social networks into clusters as shown in the example. A cluster is understood to be a subset of the nodes of a network with large density of links amongst them and sparse connectivity to other clusters. The detection of clusters is meant to yield insights into the function of a network, and to reveal common properties amongst nodes, as well as properties of individual nodes. It has proven itself fundamental for uncovering semantics within large and complex data sets.

Clustering techniques have been studied for some time now and application driven invention of algorithms on the one hand and academic interest on the other hand have led to a vast set of tools. While it is possible to efficiently find clusterings, measure their quality (or significance), and to evaluate their structure on static snapshots of networks, their evolution over time has not yet been considered in depth. However, lots of natural networks are subject to changes.

As an example, note the constantly changing link structure within social networks like Facebook ([8]), orkut ([15]) or the German studiVZ ([31]), that are becoming more and more popular in everyday life. A clustering from a month ago adequately decomposing a snapshot into groups of friends might not be significant today, though parts of it (e.g. single cliques) are still valid and could even be transferred to a current clustering. Calculations made and measures taken according to the analysis of a partitioning might lead to the requirement that a clustering of a future state of such an evolving network should resemble the current, at least to some extent. Additionally, the observation of developments in the clustering structure over the course of time is supposed to foster further understandings. A smooth transition between successive clusterings allows the preservation of a so called *mental map* of the network, which is of particular importance within large scale graphs, that humans can only capture by clustering them in the first place. Therefore, a massive change within the partitioning would lead to a loss of information and the perception of a new graph.

In addition to that, clustering evolving networks needs to be studied from an algorithmic point of view. It is desireable to incorporate existing knowledge about significant partitionings of previous snapshots of an evolving network into the detection of partitionings for current instances, in order to reuse results of previous calculations. This is supposed to lower the complexity of the problem and to save running time.

It is the goal of this thesis to explore and develop applicable techniques, that enable the analysis of dynamic networks by evaluating clustering structures and their evolution over time. A focus is put on methods making use of mathematical programming, which allow the definition of exact properties of their results on the one hand, but are hard to solve on the other hand. Besides clustering quality and smooth progression, computability will be a third aspect when reviewing found approaches.

1.2 Structure

Chapter 2: Fundamentals

The fundamental concepts and algorithms used within this thesis are explained. This includes the definition of the main datastructures, applied techniques to measure clustering quality and to compare clusterings to one another, as well as some established algorithms to generate cluster graphs and to detect clusterings of graphs.

Chapter 3: Problem Definition

The problem definition for the Dynamic Graph Clustering Problem is given. Its main criteria are highlighted and the difficulties in optimizing those are explained. Additionally, the scope of the problem instances with regard to the availability of data is restricted to an online problem version.

Chapter 4: Index Maximization via Mathematical Programming

An established integer linear program formulation used to solve modularity based clustering optimization is analyzed. It is shown how it can be optimized with regard to its complexity, in order to speed up the calculation of optimal solutions. The concept is then transferred to another known index. Finally the two are compared.

Chapter 5: Bi-Criterial Mathematical Programming

A generic framework explicitly balancing the optimization of the two opposing criteria (i.e. clustering quality and stability) of the Dynamic Graph Clustering Problem by solving an integer linear program is developed. It is evaluated by solving a problem instance based on real world data.

Chapter 6: Partial Mathematical Programming

It is explained, how an integer linear program can be used to explicitly optimize index-based clustering quality while implicitly preserving a dynamic clustering's structure. The approach is tested on generated problem instances and the quality achieved is compared against known results. Also, a large problem instance derived from real world data is solved and evaluated.

Chapter 7: Conclusion

The thesis is concluded with a summarizing overview of the found results. Additonally, a brief outlook on possibly following work is given.

2 Fundamentals

2.1 Dynamic Graphs

2.1.1 Graphs

The concept of a dynamic (i.e. evolving) graph as it is used in this thesis is based on the common notation of a *graph* as a set of *vertices* and a set of *edges* that serve as links between those vertices. While edges are not considered to be directed, their weight will be of particular interest later on.

Definition 2.1 *An undirected graph $G = (V, E)$ consists of a set of vertices V and a set of edges $E \subseteq \{\{u, v\} | u, v \in V\}$ between them. A weighted Graph $G_w = (V, E, w)$ additionally has a function $w : E \to \mathbb{R}$, that denotes the weight of each edge. The number of nodes of G is $n = |V|$, and the number of edges of G is $m = |E|$. Within a weighted graph, $w(E) = \sum_{e \in E} w(e)$ is the total edge weight.*

2.1.2 Extension to Dynamic Graphs

As an extension of a regular graph representing static data, dynamic graphs represent dynamic information and therefore have to adapt to changes in the underlying data over time. These changes could result in the introduction of new nodes or edges, the deletion of existing ones, or changes in the weights of edges in the weighted graph's case. Because of trying to find a description of a dynamic environment that is able to reflect every single modification in a graph, it seems inevitable to be able to treat them as distinct steps. The straightforward implementation of these requirements leads to a dynamic graph being an ordered set of non-dynamic graphs.

Definition 2.2 *A dynamic graph $\mathcal{G} = \{G_0, \ldots, G_n\}$ consists of an ordered set of graphs $G_1, \ldots, G_n$. Each graph G_i represents the underlying data at time t_i. The length of the dynamic graph $\mathcal{G}$ is n.*

While the changes in a graph between two consecutive points in time do not have to be bounded in any way (i.e. G_i and G_{i+1} do not have to be alike at all), the smallest changes in its structure include the insertion, deletion or modification of a single node or edge. For reasons described later on (see Section 2.3) only changes involving edges and therefore node degrees are influencing the given problem. This means that the introduction of a node (that must have degree 0) or the deletion of a node without any adjacent edges is not relevant. For the purpose of this thesis it is necessary to observe the introduction or deletion of single edges or changes to single edges' weights respectively.

2.2 Dynamic Clusterings

2.2.1 Clusterings

A clustering is a decomposition of the vertices of a graph into usually smaller but non-empty subsets. From a semantic point of view, it intends to break up the graph into groups of nodes sharing common traits, in order to clarify some information contained in the graph's structure. Such single groups are called clusters.

Definition 2.3 *A clustering $\mathcal{C}_G = \{C_1, \ldots, C_n\}$ of a graph $G = (V, E)$ consists of a set of clusters $C_1, \ldots, C_n$ with $\forall i, j \in \{1, \ldots, n\}$ $C_i \subseteq V$ and $i \neq j \Leftrightarrow C_i \cap C_j = \emptyset$ and $C_1 \cup \ldots \cup C_n = V$.*

A cluster that contains only one single node is called a *singleton*. A clustering is called trivial, if each node is in a separate singleton (*singleton-clustering*), i.e there are $n = |V|$ clusters, or if it consists of one single cluster only (*1-clustering*).

Finding a clustering is usually achieved by decomposing the set of vertices into subsets, such that the number of edges within clusters (*intra-cluster edges*) is large in comparison to the number of edges between clusters (*inter-cluster edges*). While the number of nodes within a cluster C_i will be denoted by $|C_i|$, the number of edges between two clusters C_i, C_j is given by $E(C_i, C_j)$, with $E(C_i) = E(C_i, C_i)$ being shorthand for the number of edges within C_i and $E(C_i)^c = \sum_{C_j \in \mathcal{C} \setminus \{C_i\}} E(C_i, C_j)$ being shorthand for the number of edges between C_i and all other clusters. Additionally, the number of non-adjacent node pairs between two clusters is $\overline{E}(C_i, C_j)$, $\overline{E}(C_i)$ is shorthand for the number of non-adjacent pairs within C_i and $\overline{E}(C_i)^c$ results in the number of non-adjacent pairs between C_i and all remaining clusters. The cluster a node $v \in V$ belongs to shall be given by $clust(v)$. In case of a weighted graph, the number of edges is substituted by their weight within all terms.

2.2.2 Dynamic Clusterings

The straightforward extension of the concept of a clustering for a static graph to a clustering for a dynamic graph is realized by assigning each of the single per-timestep graphs of a dynamic graph a single clustering.

Definition 2.4 *A dynamic clustering $\mathcal{C}_{\mathcal{G}} = \{\mathcal{C}_1, \ldots, \mathcal{C}_l\}$ of a dynamic graph $\mathcal{G}$ with length l consists of a set of clusterings $\mathcal{C}_1, \ldots, \mathcal{C}_l$ where $\mathcal{C}_i$ is a clustering of the graph G_i.*

2.3 Clustering Indices as Quality Measures

Clustering indices are the means for quantitatively formalizing the yet imprecise paradigm of searching for dense subgroups within a graph. Serving as a mathematical notion of how good a clustering is, they allow the optimization of it towards a certain index's optimal value as well as the comparison of two different clusterings with regard to them. For more general information about the concept of quality measures for clusterings see [16].

A clustering index is a function $index(G, \mathcal{C}_G) \rightarrow \mathbb{R}$ that uses a graph G and a clustering $\mathcal{C}_G$ as the input and produces a single value describing how well the clustering fits the index's demands. By normalizing any index against the number of nodes and edges considered, its value is bounded by a certain constant. In most cases the maximal value is 1 and the minimal value is 0. As this creates the impression that cross-graph comparison of clustering quality is possible, it has to be pointed out, that for most scenarios it is not possible to find a clustering with the index's maximum value. A clustering with a lower value for one graph might be optimal, while it yields a higher value for a graph with a different edge set and yet is obviously bad. Therefore, a clustering achieving a relatively high index value for a specific graph may only be considered significant in comparison to other clusterings of the same graph with lower index values.

The following clustering indices will be the basis for the algorithms outlined in this thesis, though the framework itself allows to substitute them by any other index adequately fulfilling the stated demands.

2.3.1 Modularity

The *modularity* index has been introduced by Newman et al. in [27]. It is based on the *coverage* index, that measures the fraction of edges (or edges' weight) that are in clusters. The coverage index is given as follows.

Definition 2.5 *The coverage* $cov(G, \mathcal{C}_G)$ *of a weighted graph* $G = (V, E, w)$ *and a clustering* $\mathcal{C}_G$ *for this graph is defined as*

$$cov(G, \mathcal{C}_G) = \sum_{C \in \mathcal{C}_G} \frac{E(C)}{w(E)}$$

The basic idea behind modularity is to normalize the coverage index's value by subtracting the expected coverage value, i.e. the number or weight of edges within clusters of a graph with the same degree distribution but random edges. Therefore, the probability to have nodes u, v linked by an edge is set to $w(u) \cdot w(v)/(2 \cdot w(E))$. In the appendix of [12] Wagner et al. state that this leads to an expected sum of edge weights being unequal to $w(E)$, with an error tending to zero as the total edge weight increases. The formulation of the modularity index is given as follows, note that its minimal value is -0.5.

Definition 2.6 *The modularity* $mod(G, \mathcal{C}_G)$ *of a weighted graph* $G = (V, E, w)$ *and a clustering* $\mathcal{C}_G$ *for this graph is defined as*

$$\begin{aligned} mod(G, \mathcal{C}_G) &= \sum_{C \in \mathcal{C}_G} \left(\frac{E(C)}{w(E)} - \left(\frac{E(C) + \sum_{C' \in \mathcal{C}} E(C, C')}{2 \cdot w(E)} \right)^2 \right) \\ &= \sum_{C \in \mathcal{C}_G} \left(\frac{E(C)}{w(E)} - \left(\frac{\sum_{v \in C} w(v)}{2 \cdot w(E)} \right)^2 \right) \end{aligned}$$

where $w(v), v \in V$ *denotes the weight of node* v *and is the sum of all adjacent edges' weights.*

In case of an unweighted graph, let $w(e)$ equal $1 \, \forall e \in E$ so that $w(E)$ equals $|E|$, resulting in an unchanged definition.

The consideration of the expected value is meant to establish a link between the absolute index value and the significance of a clustering. In this case, a given clustering will be considered significant, if it achieves a high index value compared to a random clustering. Figure 2.1 shows two clustered graphs taken from [12], a random split on the one hand and a meaningful partitioning of a six-sided tube on the other hand. While the split achieves a higher coverage value than the tube's clustering, the low significance of its random clustering due to the high expected value leads to lower modularity.

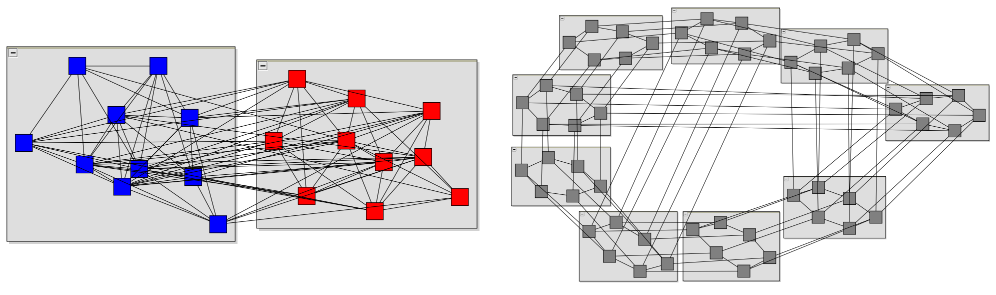

Figure 2.1: Figure (a) shows a random split of a graph with 20 nodes. It yields a coverage of 0.13 and has modularity 0.32. Figure (b) depicts the meaningful clustering of a six-sided tube, yielding a coverage of 0.43 and modularity of 0.32. Both graphs are taken from [12].

2.3.2 Significance-Performance

Inspired by the fact that modularity is based on the common coverage index and its expected value, in [12] Wagner et al. develop a framework for the creation of other indices incorporating the significance of a clustering. The proposed *significance paradigm* compares graphs to a null hypothesis with regard to certain properties, not necessarily describing the complete graph but a family sharing properties, e.g. degree sequence, number of nodes or edges, or degree distribution. In order to demonstrate its capabilities, the also well known *performance* index, that counts the number of correctly classified node pairs (i.e. the number (or weight) of intra-cluster-edges and the number of non-linked (or weakly linked) pairs between clusters), gets extended analogously to modularity and becomes the *significance-performance* index. Again, the classification of graph families is based on the degree distribution, as is the calculation of the expected number of correctly clustered pairs.

The following notations are based on those used in [12] and are supposed to ease the definition of the significance-performance index.

The number of all intra-cluster edges in a clustering $\mathcal{C}$ is:

$$m(\mathcal{C}) = \sum_{C \in \mathcal{C}} E(C)$$

The number of all inter-cluster edges within a clustering $\mathcal{C}$ is:

$$\overline{m}(\mathcal{C}) = \sum_{\{C,C'\} \in \mathcal{C}^2, C \neq C'} E(C, C')$$

The number of all non-connected node pairs residing in different clusters is:

$$\overline{m}(\mathcal{C})^c = \sum_{\{C,C'\} \in \mathcal{C}^2, C \neq C'} \overline{E}(C, C')$$

The weight of all intra-cluster edges within a clustering $\mathcal{C}$ is:

$$w(\mathcal{C}) = \sum_{\{u,v\} \in E,\ \text{clust}(u)=\text{clust}(v)} w(u, v)$$

The weight of all inter-cluster edges within a clustering $\mathcal{C}$ is:

$$\overline{w}(\mathcal{C}) = \sum_{\{u,v\} \in E,\ \text{clust}(u)\neq\text{clust}(v)} w(u, v)$$

With these notations at hand, a formulation of the complete index comes down to the formulation of performance itself, as well as its expected value.

The Performance Index

Definition 2.7 *The performance value perf$(G, \mathcal{C}_G)$ of a weighted graph $G = (V, E, w)$ and a clustering $\mathcal{C}_G$ for this graph is defined as*

$$\textit{perf}(G, \mathcal{C}_G) = \frac{w(\mathcal{C}_G) + w_{max} \cdot \overline{m}(\mathcal{C}_G)^c + w_{max} \cdot \overline{m}(\mathcal{C}_G) - \overline{w}(\mathcal{C}_G)}{0.5 \cdot n(n-1) \cdot w_{max}}$$

where the normalizing factor $1/(0.5 \cdot n(n-1))$ represents the number of node pairs and w_{max} is the maximum weight of all edges within the graph.

All non-adjacent node pairs between clusters add the same value to the performance index as an edge of maximum weight within a cluster. Inter-cluster edges e are also considered, they contribute $w_{max} - w(e)$, hence favoring weak connections between clusters over others.

The Expected Value of the Performance Index

Definition 2.8 *The expected performance value exp-perf$(G, \mathcal{C}_G)$ of a weighted graph $G = (V, E, w)$ and a clustering $\mathcal{C}_G$ for this graph is defined as*

$$\begin{aligned}
\textit{exp-perf}(G, \mathcal{C}_G) =& \frac{\frac{1}{2 \cdot w(E)} \cdot \sum_{C \in \mathcal{C}} \sum_{(u,v) \in C^2} w(u) \cdot w(v)}{0.5 \cdot n(n-1) \cdot w_{max}} \\
&+ \frac{\sum_{C \in \mathcal{C}} \sum_{C' \in \mathcal{C} \setminus C} \sum_{(u,v) \in C \times C'} (w_{max} - \frac{w(u) \cdot w(v)}{2 \cdot w(E)})}{0.5 \cdot n(n-1) \cdot w_{max}} \\
=& \frac{\frac{1}{w(E)} \sum_{C \in \mathcal{C}_G} (\sum_{v \in C} w(v))^2 + w_{max}(n^2 - \sum_{C \in \mathcal{C}_G} |C|^2) - 2 \cdot w(E)}{n(n-1) \cdot w_{max}}
\end{aligned}$$

where the normalizing factor $1/(0.5 \cdot n(n-1))$ represents the number of node pairs and w_{max} is the maximum weight amongst all edges within the graph.

The calculation of the expected performance value of a class of graphs with the same degree distribution is rather complicated in comparison to modularity. Again, the underlying assumption is that the probability to have an edge connecting nodes $u, v \in V$ is expressed by $w(u) \cdot w(v)/(2 \cdot w(E))$. As a probability value of 0 reflects the fact that nodes will not be connected, the straightforward implementation again considers light edges to be more likely between clusters.

The Significance-Performance Index

Definition 2.9 *The significance performance value sigperf$(G, \mathcal{C}_G)$ of a weighted graph $G = (V, E, w)$ and a clustering $\mathcal{C}_G$ for this graph is defined as*

$$\textit{sigperf}(G, \mathcal{C}_G) = \textit{perf}(G, \mathcal{C}_G) - \textit{exp-perf}(G, \mathcal{C}_G)$$

Remarks

As described for modularity, an unweighted version of the given indices and expected values is equivalent to that of a weighted graph where $w(e) = 1 \, \forall e \in E$.

Besides the two variants presented so far, [12] also investigates the behavior of relative versions of significance-coverage (i.e. modularity) and significance-performance. Instead of subtracting expected values, the index values are then divided by them. For the purpose of this work, the relative versions are of no concern.

2.4 Distance Measures for Clusterings

So far the question has been answered which of two given clusterings $\mathcal{C}_G, \mathcal{C}'_G$ of a graph G better suits the demands of a given index $index(G, \mathcal{C}_G)$. The answer simply derives from the calculation and comparison of the index values, the clustering achieving the higher result being superior. However, another problem that will be faced throughout this work is to find a quantification of the difference between two clusterings, not only in quality but also structure. It seems there are many ways to solve this task, but then again the various techniques' outputs often lack intuitive behavior and their interpretation is difficult. A subset of such distance measures for clusterings will be discussed in this section.

2.4.1 Types of Distance Measures for Clusterings

In [6] Delling evaluates distance measures systematically. He first distinguishes between quality based distance measures, node structural distance measures, and graph structural distance measures. The first ones compare index values (as described in Section 2.3), while the latter two actually consider the partitioning induced by a clustering of a graph. Graph structural measures additionally use information about the graph itself, enriching the analysis by knowledge about links between nodes. Therefore they are influenced by the clustering's structure and its quality. The two classes of structural measures can be subdivided into pair counting measures, matching measures, and entropy measures. Examples for these classes will be given in Section 2.4.2. So far, besides Delling's work the literature only deals with node structural distance measures.

Delling tries to shed light on how all measures express minimum and maximum distances between two clusterings. Minimum distance (i.e. equality) is easy to determine for all types, but saying what two clusterings are of maximal difference becomes hard for the structural case. It is also stated that all measures lack *quality sensing* abilities, meaning that an atomary change to a significant clustering should result in a more noteable difference than the same change to a less significant one.

He concludes that node structural methods tend to give the best results, emphasizing his observation that amongst those entropy versions show the most intuitive behavior. Thus, the node structural measures will be considered in this thesis.

2.4.2 Node Structural Distance Measures

As clustering quality in terms of index values and clustering distances will be optimized separately, it is enough to pay attention to node structural distance measures, that are widely spread in the literature and show the most promising results being studied in [6]. This section will give an overview and introduce all techniques applied within this thesis.

Rand Measure

This measure by Rand (see [29]) does a comparison based on single node pairs and how they are assigned in each clustering. It counts the number of pairs that are either in the same cluster or in different clusters within both partitionings and normalizes by the total number of pairs.

The set of pairs that are clustered together within both clusterings $\mathcal{C}, \mathcal{C}'$ of the same graph G is denoted with S_{11}, pairs that are clustered in different clusters in both clustering belong to S_{00} (the nomenclature follows that of Delling):

$$S_{11} = \{u, v | u, v \in V, clust_{\mathcal{C}}(u) = clust_{\mathcal{C}}(v), clust_{\mathcal{C}'}(u) = clust_{\mathcal{C}'}(v)\}$$

$$S_{00} = \{u, v | u, v \in V, clust_{\mathcal{C}}(u) \neq clust_{\mathcal{C}}(v), clust_{\mathcal{C}'}(u) \neq clust_{\mathcal{C}'}(v)\}$$

Definition 2.10 *The Rand distance measure rand$(\mathcal{C}, \mathcal{C}')$ between to clusterings $\mathcal{C}, \mathcal{C}'$ is defined by*

$$rand(\mathcal{C}, \mathcal{C}') = 1 - \frac{2 \cdot (|S_{11}| + |S_{00}|)}{n \cdot (n-1)} \quad .$$

By concept, the minimal value that can be achieved is 0. It allows the conclusion that the two clusterings are identical. Two clusterings will have maximum distance, if one is a singleton clustering and the other one is the 1-clustering. For any other clustering that has more than 1 and less than n clusters it is impossible to find a counterpart that results in a distance of 1. Delling describes the main disadvantage of the Rand measure to be its dependence on the number of clusters, i.e. for random clusterings the distance tends to a threshold of 0 for a rising number of clusters. While he claims that this leads to the method being useless for the general comparison of clusterings, real world clusterings as well as modularity-optimized clusterings of real world data tend to have a somewhat bounded number of clusters. Therefore, the observation of Rand values is still justified.

Jaccard Measure

The Jaccard measure (or Sørensen measure, see [32]) is motivated by clustering problems in biology and geology. It is similar to the Rand measure, but concentrates on node pairs that are clustered together. Therefore it only considers S_{11} and leaves S_{00} aside.

Definition 2.11 *The Jaccard distance measure jacc$(\mathcal{C}, \mathcal{C}')$ between two clusterings $\mathcal{C}, \mathcal{C}'$ is defined by*

$$jacc(\mathcal{C}, \mathcal{C}') = \begin{cases} 1 - \frac{2 \cdot |S_{11}|}{n \cdot (n-1) - 2 \cdot S_{00}} & , \textit{for } n \cdot (n-1) - 2 \cdot S_{00} > 0 \\ 0 & , \textit{otherwise} \end{cases} \quad .$$

The second case will only occur, if two singleton clusterings are compared.

The Jaccard distance measure registers the singleton clustering and the 1-clustering to have the maximum distance of 1, as pointed out by Delling. While the comparison of two singleton clusterings results in a value of 0, any other clustering compared to a singleton clustering will result in a distance of 1. This proofs to be unintuitive as a singleton clustering compared to a clustering with $|\mathcal{C}| = n - 1$ (i.e. only one cluster holds two nodes) should be rated to be nearly alike. Unfortunately, as long as a singleton clustering is involved $S_{11} = \emptyset$ holds.

Fowlkes-Mallows Measure

The Fowlkes-Mallows distance measure as defined in [10] makes use of the *matching matrix* (see the Maximum Match measure). In [6] it is shown that the same measure can be formulated by the usage of the sets defined in 2.4.2 plus the number of pairs that are clustered together within one clustering and are separated within the other (and the other way round). In addition to the known sets, S_{10} and S_{01} are used as follows. Again, the nomenclature is taken from [6].

$$S_{10} = \{u, v | u, v \in V, clust_{\mathcal{C}}(u) = clust_{\mathcal{C}}(v), clust_{\mathcal{C}'}(u) \neq clust_{\mathcal{C}'}(v)\}$$

$$S_{01} = \{u, v | u, v \in V, clust_{\mathcal{C}}(u) \neq clust_{\mathcal{C}}(v), clust_{\mathcal{C}'}(u) = clust_{\mathcal{C}'}(v)\}$$

Definition 2.12 *The Fowlkes-Mallows distance measure foma$(\mathcal{C}, \mathcal{C}')$ between two clusterings $\mathcal{C}, \mathcal{C}'$ is defined by*

$$\textit{foma}(\mathcal{C}, \mathcal{C}') = \begin{cases} 1 - \frac{|S_{11}|}{\sqrt{(|S_{11}|)+|S_{10}|)\cdot(|S_{11}|+|S_{01}|)}} & \textit{, for } |S_{01}|, |S_{10}| > 0 \vee |S_{11}| > 0 \\ 1 & \textit{, for } |S_{11}|, |S_{10}| = 0 \vee |S_{11}|, |S_{01}| = 0 \\ 0 & \textit{, otherwise} \end{cases}$$

The second case guarantees, that all comparisons where only one argument is a singleton clustering result in the maximal distance of 1.

Maximum Match Measure

The Maximum Match distance measure is the first and only matching measure used within this thesis. Delling introduces it as a symmetric generalization of the classification accuracy measure by Meilă and Heckerman in [24]. Like all matching measures it is based on the so called matching matrix, i.e. a $|\mathcal{C}|$-by-$|\mathcal{C}'|$-matrix M whose entry $M[i, j]$ holds the amount of nodes within $C_i \in \mathcal{C} \cap C_j \in \mathcal{C}'$:

$$M \in \mathbb{N}^{|\mathcal{C}| \times |\mathcal{C}'|} = m_{ij} \text{ where } m_{ij} := C_i \in \mathcal{C} \cap C_j \in \mathcal{C}'$$

In [6] the Maximum Match value is computed in a greedy process by finding the maximum entry within M, deleting its row and column within the matrix, and finally summing over all found entries. The implementation of a function MaximumMatch$(\mathcal{C}, \mathcal{C}')$ is given as pseudo code in Algorithm 1. As a maximum of n entries could have been classified the same, the measure is normalized by $1/n$. It is pointed out that whole clusters are ignored in the case of $|\mathcal{C}| \neq |\mathcal{C}'|$. As this does not necessarily result in the best pairing, it might be wortwhile to utilize a known algorithm to find the best possible matching between the clusters of both clusterings, e.g. the *Hungarian Algorithm* (see [20]) with a running time in $O(n^3)$).

Definition 2.13 *The Maximum Match distance measure maxmatch$(\mathcal{C}, \mathcal{C}')$ between two clusterings $\mathcal{C}, \mathcal{C}'$ is defined by*

$$\textit{maxmatch}(\mathcal{C}, \mathcal{C}') = 1 - \frac{1}{n} \cdot \textit{MaximumMatch}(\mathcal{C}, \mathcal{C}')$$

where MaximumMatch$(\mathcal{C}, \mathcal{C}')$ denotes the application of Algorithm 1.

Fred and Jain Measure

The last distance measure employed within this thesis is taken from the class of entropy measures, which Delling considers to give the most intuitive results for clustering comparison. *Information entropy* itself (as defined by Shannon in [30]) is a measure for the uncertainty associated with a decision or the information contained in it respectively and is counted in bits. For

Algorithm 1: MAXIMUM MATCH

```
Input: 2 Clusterings $\mathcal{C}, \mathcal{C}'$
Output: Maximum Match value $result$
1 Set $result = 0$
2 $initMatchingMatrix(M, \mathcal{C}, \mathcal{C}')$
3 while $min(rows(M), cols(M)) > 0$ do
4     $(i, j) = findMaxEntry(M)$
5     $result = result + M[i, j]$
6     $deleteRowsCols(M, i, j)$
7 return $result$
```

clusterings entropy can be understood as the information within the assignment of a node to a specific cluster. Therefore its definition is

$$\mathcal{H}(\mathcal{C}) = -\sum_{C \in \mathcal{C}} \frac{|C|}{n} \log_2\left(\frac{|C|}{n}\right)$$

where $P(C) = |C|/n$ represents the probability of a random node to be in the cluster C. The information entropy of the 1-clustering is 0 and that of the singleton clustering is $\log_2(n)$, these are the minimum and maximum values. In [23] this approach is extended to *correlation information* between two clusterings, quantifying the uncertainty of the assignment decision for one clustering if the other one is known in advance. With the probability of a node $v \in V$ being in $C \in \mathcal{C}$ and in $C' \in \mathcal{C}'$ set to $P(C, C') = |C \cap C'|/n$, the correlation information is given by the following equation. Its maximum value is bounded by the maximum values of $\mathcal{H}(C)$ and $\mathcal{H}(C')$.

$$\mathcal{I}(\mathcal{C}, \mathcal{C}') = \sum_{C \in \mathcal{C}} \sum_{C' \in \mathcal{C}'} \frac{|C \cap C'|}{n} \cdot \log_2\left(\frac{|C \cap C'| \cdot n}{|C| \cdot |C'|}\right)$$

Amongst the entropy measures, Delling favors the measure of Fred and Jain introduced in [11], as it delivers the most differentiate results and needs a special case only in case both clusterings are singleton clusterings and therefore identical.

Definition 2.14 *The Fred and Jain distance measure faj$(\mathcal{C}, \mathcal{C}')$ between two clusterings $\mathcal{C}, \mathcal{C}'$ is defined by*

$$\mathit{faj}(\mathcal{C}, \mathcal{C}') = \begin{cases} 1 - \frac{2 \cdot \mathcal{I}(\mathcal{C}, \mathcal{C}')}{\mathcal{H}(\mathcal{C}) + \mathcal{H}(\mathcal{C}')} & \text{, for } \mathcal{H}(\mathcal{C}) + \mathcal{H}(\mathcal{C}') \neq 0 \\ 0 & \text{, otherwise} \end{cases}$$

2.4.3 Distance Measures and Dynamic Clusterings

So far, all distance measures are defined for the comparison of clusterings for static graphs and therefore two clusterings partitioning the same set of nodes V. For a dynamic clustering $\mathcal{C}_\mathcal{G}$ of a dynamic graph $\mathcal{G}$ it is of interest, to compare clusterings for two graphs G, G' (not necessarily consecutive within $\mathcal{G}$) that neither need to have an identical set of nodes V nor of edges E. For the presented structural distance measures it has do be decided, how the deletion and creation of nodes within the dynamic graph are treated. Clearly, a varying nodebase should be reflected in the result and no two clusterings based on different node sets should feature a distance of 0.

For two graphs $G = (V, E), G' = (V', E')$ and two clusterings $\mathcal{C}(G), \mathcal{C}'(G')$ the extension of the distance measures works as follows. First, the node sets will be united to $V_u = V \cup V'$, $|V_u| = n_u$ will be used for all normalization purposes. All nodes only present within one clustering will be considered to be unclustered within the other one.

For the pair counting measures this means, that a node v within $V \backslash (V' \cap V)$ will affect S_{00} by the number of relations with nodes in other clusters $(n_u - 1 - \binom{|clust_{\mathcal{C}}(v)|}{2})$ and S_{10} by the number of relations with nodes in the same cluster $\binom{|clust_{\mathcal{C}}(v)|}{2}$. This sums up to $n_u - 1$ as v will be compared to all other nodes within V_u.

The Maximum Match distance measure naturally copes with the difference in node sets. The maximum result of Algorithm 1 can be $|V' \cap V|$, but the normalization is based on $|V' \cup V|$.

The information entropy of a clustering is affected by the addition of nodes as well. On the one hand the entropy value of a single cluster (being $-(|C|/n)\log_2(|C|/n)$) shrinks for clusters holding less than (n/e) nodes. On the other hand the information contributed by unclustered nodes has to be considered as well. An analysis shows that this is done best by treating unclustered nodes like singleton clusters for the clustering entropy as well as correlation information. This way, all semantics are conserved and no additional implications regarding the comparison of two clusterings are made. See Appendix A for further information.

2.5 Algorithms

2.5.1 Clustering Generators

In order to evaluate the performance of the developed algorithms, it is of interest to be able to guarantee certain properties for the graphs that they are tested with. In [3] the description of a Gaussian cluster graph generator is given, that takes a suggested number of nodes n, the probability p_I for intra-cluster edges and the probability p_O for inter-cluster edge as the input to generate a graph $G = (V, E)$ with a clustering $\mathcal{C}$. The number of clusters $|\mathcal{C}|$ is randomly chosen from the interval $[log_{10}(n), \sqrt{n}]$. The cluster sizes are chosen around the expected value $\mathbb{E}_{size} = n/|\mathcal{C}|$ and vary by a maximum deviation of $\mathbb{E}_{size}/4$. The final number of nodes will be in the interval $[n - |\mathcal{C}| \cdot \mathbb{E}_{size}, n + |\mathcal{C}| \cdot \mathbb{E}_{size}]$.

For all node pairs within each cluster an intra-cluster edge will be generated with probability p_I. The number of intra-cluster edges within one cluster thus grows quadratically in its size while their overall number is linked to the number of clusters itself. An inter-cluster edge between two nodes from different clusters will be generated with a probability of p_O. The number of inter-cluster edges grows quadratically in the number of nodes within the graph. As the number of intra-cluster edges does not grow as fast as the number of inter-cluster edges due to a growing number of clusters, larger generated graphs with the same probability values tend to be less significant. In general, an appropriate choice of p_I and p_O with respect to the other parameters is necessary in order to achieve good results.

A more intuitive usage is realized by automatically determining p_O by giving a ratio r between intra- and inter-cluster edges. The probability p_O is calculated by dividing the expected number of intra-cluster edges by the number of inter-cluster node pairs times the ratio r. For a fixed number of nodes this leads to the following behavior when varying the number of clusters. A smaller number of clusters leads to more intra-cluster pairs and a larger number of expected intra-cluster edges. The expected number of inter-cluster edges is a constant fraction of this number. Those have to be distributed amongst fewer inter-cluster node pairs. The probability for inter-cluster edges grows and the significance of the clustering shrinks. Therefore, the implementation of the ratio r does not help to overcome the need to manually adapt the given parameters, in order to generate clustergraphs with similar significance independently from the size of the graph and the number of clusters.

So far there are no established generators for dynamic clustergraphs, though there is a need for dynamic networks with parametrized properties. There are several ongoing projects, but unfortunately none of those are ready to produce graph material usable for this thesis yet.

2.5.2 Maximizing Modularity via Mathematical Programming

In order to maximize modulartity the clustering problem can be cast into a set of constraints and an objective function to be solved as an integer linear program (ILP). Several similar formulations guaranteeing consistency of a clustering are available, the one used in this work is adapted from [2].

Constraints and Objective Function

For any graph $G = (V, E)$, n^2 decision variables are defined, each representing the relation between to single nodes. For two nodes $u, v, \in V$ the variable $X_{uv} \in \{0, 1\}$ equals 1 iff u and v belong to the same cluster. According to this, constraints for reflexivity, symmetry and transitivity are defined as follows.

$$\begin{aligned}
\forall u \in V &: X_{uu} = 1 \\
\forall \{u, v\} \in V^2 &: X_{uv} = X_{vu} \\
\forall \{u, v, w\} \in V^3 &: \begin{cases} X_{uv} + X_{vw} - 2 \cdot X_{uw} \leq 1 \\ X_{uv} + X_{uw} - 2 \cdot X_{vw} \leq 1 \\ X_{uw} + X_{vw} - 2 \cdot X_{uv} \leq 1 \end{cases}
\end{aligned}$$

The necessary objective function directly derives from the modularity function given in Section 2.3. It is rewritten to sum over all node pairs. The ILP needs to find a consistent assignment of the decision variables, that maximizes the objective function by summing over the scalars of all variables that equal 1. The function is given as follows.

$$\begin{aligned}\operatorname{mod}(G,\mathcal{C}_G) &= \sum_{C in \mathcal{C}_G}\left(\frac{E(C)}{w(E)} - \left(\frac{\sum_{v\in C} w(v)}{2\cdot w(E)}\right)^2\right)\\ &= \frac{1}{2\cdot w(E)}\cdot\sum_{(u,v)\in V^2}\left(w(u,v) - \frac{w(u)\cdot w(v)}{2\cdot w(E)}\right)X_{uv}\end{aligned}$$

As previously defined $w(v)$ denotes the weight of node v (i.e. the sum of all weights of all edges adjacent to v), $w(v,u)$ is shorthand for $w(e), e = (v,u) \in E$.

Running Time

In general, solving integer programs is $\mathcal{NP}$-hard, while a solution of the corresponding linear program can be computed in polynomial time. The actual time needed to find a solution for the modularity ILP depends on the number of constraints and variables used, as the branch and bound method employed by most ILP solvers corresponds to a complete enumeration of all solutions by climbing down a (binary) tree. During this process, branches leading to impossible results are bounded as early as possible, in order to shrink the space of solutions. How early and how often such branches can be bounded depends on the particular instance of a problem and hence the underlying graph's structure. Given the LP solution, it can be seen that many decision variables feature results close to 0 and 1, while others are close to 0.5. For the latter ones, a decision is hard to make and thus the solver cannot decide to bisect the decision tree due to their branches, leading to more solutions that need to be considered.

In practice, problem instances for graphs with less than 50 nodes can normally be solved within seconds or few minutes, while graphs with more than 100 nodes and a non-trivial structure are likely to consume hours or even days of calculations. For even larger graphs, finding the optimal ILP solution might not be feasible at all. Therefore, finding optimal modularity via solving the ILP can be discarded as a possible standard solution.

Rounding the LP

Instead of solving the ILP to maximize modularity, several attempts have been made to round a relaxed solution of the respective linear program, which also gives an upper bound on the possible maximal index value itself. In [19] and [18] different approaches to use the LP results are suggested. For all node-node relations the LP solver assigns values close to 0 or 1 to, the integer decision is likely to be accordant. In [19] good index values are achieved by first merging all nodes with a decision variable set close to 1 and then greedily merging clusters leading to the highest increase in modularity (compare Section 2.5.3). In [18] a technique essentially identical to the one used by Charikar et al. for the rounding of correlation clustering instances ([5]) is applied by using a threshold to determine the merging process. Unfortunately, the guarantees calculated in [5] are not transferable to optimizing modularity.

Note that rounding the linear program requires solving it first. Especially for large graphs or badly structured graphs (with many 0.5-variables) this is still very time consuming, so far making it unpractical for highly dynamic evolution and the implicated constant recalculation of problem instances.

2.5.3 A Greedy Algorithm for Modularity

In [26] Newman also proposes a greedy algorithm based on modularity. Its approach is to start out with a singleton-clustering and to keep on merging the clusters that yield the highest increase or the least decrease in modularity. The algorithm produces a *dendrogram* representing each step taken, and the implied clusterings' modularity values. The clustering with the highest value is chosen to be the output of the algorithm.

To find the next step, the algorithm maintains a symmetric $|\mathcal{C}|$-by-$|\mathcal{C}|$ matrix M, where each entry $M[i,j]$ represents the clustering's modularity in case the clusters C_i and C_j are merged. The pseudo code of this greedy algorithm is given in Algorithm 2.

As it will never increase the index value to merge two unconnected clusters there are at most $m = |E|$ possible updates. The change in modularity upon joining two clusters can be calculated in constant time and therefore the matrix M can be computed in time $O(m)$. Then again the update after the best merge has been found only influences matrix entries in the rows and columns holding $M[i,j]$, their number is in $O(n)$. In total there are $n-1$ joins possible and this results in the worst case running time of greedy modularity being in $O((m+n)\cdot m)$.

Algorithm 2: GREEDY MODULARITY

```
  Input: Graph $G = (V, E, w)$
  Output: Clustering $\mathcal{C}_G$ of $G$
1 $resetToSingletons(\mathcal{C}_G)$
2 $initMergingMatrix(M, \mathcal{C}_G)$
3 while $dim(M) > 1$ do
4     $(i,j) = findMaxEntry(M)$
5     $mergeClusters(i,j)$
6     $updateMatrix(M,i,j)$
7 return maximum modularity clustering
```

While exploring the significance paradigm, Wagner et al. also show the straightforward generalization of the greedy algorithm for modularity proposed by Newman et al., in order to produce the same type of clusterings for other indices (such as significance-performance).

3 Problem Definition

This chapter will give an introduction to the origin of the Dynamic Graph Clustering Problem and its classification amongst graph clustering problems in general. As it has not yet been the topic of publicly available studies, it will also be explained what framework will be the basis for all algorithms and what reasons led to this design.

3.1 Overview

As dynamic graphs extend normal graphs by the possibility of representing change within the underlying information over time, dynamic clusterings allow the adaptation of a clustering to the changing structure of their respective dynamic graph (see Section 2.1 and Section 2.2). A dynamic graph can basically be seen as a sequence of ordinary graphs and a dynamic clustering as a sequence of ordinary clusterings linked to those. Therefore it is possible to treat each timestep of a dynamic graph as a single clustering problem (i.e. find a clustering for each timestep's graph), and to create a dynamic clustering by solving all of those. Of course this completely ignores any (semantic) link between graphs of different timesteps.

In Section 2.1.2 it has been stated, that modifications during the evolution of a dynamic graph are not bounded in any way. Then again it is intended to understand this evolution, leading to the assumption that consecutive timesteps' graphs are similar (i.e. only differ in a small number of changes to nodes and links). Unfortunately this does not necessarily hold for consecutive (optimal) clusterings, as clustering indices (such as modularity and significance-performance, see Section 2.3) show *non-local behavior*. A slight change in the graph from one timestep to the next might result in a complete change of the partitioning induced by a related optimal clustering. Figure 3.1 illustrates this phenomenon. The colors represent the optimal clusterings for both graphs. Though only the lower clusters are manipulated, the whole clustering is affected and the non-adjacent upper clusters are merged (this particular effect is called the *resolution limit* as optimal clusters in a small graph might be combined in an optimal clustering of a larger graph, see [9] for more information). As a direct result it is not feasible to only manipulate a given clustering for the original graph locally while guaranteeing global optimality with regard to an index for the updated graph. On the other hand in [2] it has been shown by Wagner et al. that modularity optimization is $\mathcal{NP}$-complete, making it desirable to reuse the information of an optimized clustering for one timestep to solve the clustering problem for the next one, in order to lower the time complexity of the calculation.

Another aspect of non-local differences between consecutive timesteps' clusterings derives from the application of graph clusterings itself. It may not only rely on clustering quality but structural properties as well. In [13] Wagner et al. remark that besides an index value the distance between consecutive clusterings has to be regarded as an additional criterion. The sim-

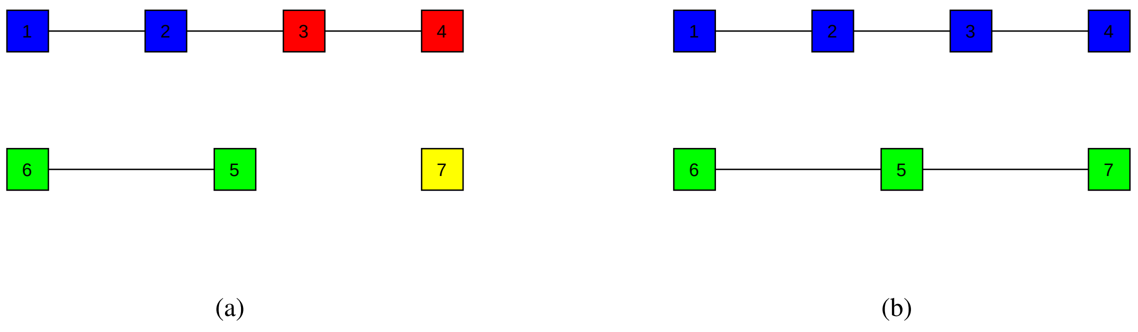

Figure 3.1: Consecutive timesteps' graphs illustrating the non-local behavior of modularity optimization. Colors represent the optimal clustering of each graph. Though only the lower clusters are manipulated, the whole clustering is affected.

ilarity of each clustering to its predecessor in time allows the transfer of insights derived from previous observations and enables an understanding and analysis of the structural development of the clustering over time. This technique is called preservation of the mental map and relates to similar tasks in the field of graph layouts (see [7],[25]). Especially when analyzing large graphs, a person is not able to detect and remember the massive amount of information. Thus an abstraction is made and a clustering-like structure is captured in case it is not given explicitly. Therefore a given clustering is vital for the perception of the graph and a change in its structure between timesteps might lead to the perception of a completely different network.

3.2 The Dynamic Graph Clustering Problem

The two main criteria for the Dynamic Graph Clustering Problem (informally given in Section 3.1) are to find a clustering for each timestep that yields a high index value, as well as to guarantee similarity between consecutive timesteps' clusterings. There are several formal methods available to measure the clustering quality with respect to certain properties, some being introduced in Section 2.3. The most common and applicable algorithm to create good clusterings with regard to a specific index is the greedy algorithm, originally used by Newman (see Section 2.5.3). The structural distance between clusterings can also be measured formally, a subset of well known approaches is presented in Section 2.4. Following that, a general definition of the Dynamic Graph Clustering Problem is given as follows.

Definition 3.1 *Given a dynamic graph $\mathcal{G} = (G_0, \dots, G_n)$, a clustering $\mathcal{C}_{G_0}$ for its first timestep, a clustering index $q(G_i, \mathcal{C}_{G_i})$, and a distance measure $dist(C_i, C_j)$. Find a high quality clustering with regard to the given index for each of the graphs $G_1, \dots, G_n$ so that the distance between consecutive clusterings $dist(\mathcal{C}_{G_i}, \mathcal{C}_{G_{i+1}})$ is small.*

It has already been stated, that the distance between consecutive optimal clusterings can be arbitrarily large. Here it becomes clear, that the optimizations of the parameters clustering quality and distance can be conflicting. The quantification of how to optimize a combination of index value and structural distance to a previous clustering is complicated and no optimal ratio is known in advance. This is due to the fact that neither a generally optimal index value (and its corresponding clustering) is available, nor a precise formulation of what distance measure value is acceptable in order to guarantee a certain similarity between consecutive clusterings. How both are related is also unknown. It is the main task in solving the Dynamic Graph Clustering Problem to find the balance between both criteria.

Given the optimal index value $q_{opt}(G_i)$ and the optimal distance value $dist_{opt}(G_i, \mathcal{C}_{G_{i-1}})$ for each graph G_i, possible stronger formulations include:

- Given a weighting factor $b \leq 1.0$. For each G_i find the clustering $\mathcal{C}_{G_i}$ with

  $$q(G_i, \mathcal{C}_{G_i}) \geq b \cdot q_{opt}(G_i)$$

  that yields the smallest distance value. For $b = 1.0$ this corresponds to finding the clustering with the best distance value amongst all clusterings that achieve the optimal index value. Instead of the relative threshold, an absolute version can be formulated analogously.

- Given a constant $d \geq 0$. For each G_i find the clustering $\mathcal{C}_{G_i}$ with

  $$dist(\mathcal{C}_{G_i}, \mathcal{C}_{G_{i-1}}) \leq dist_{opt}(G_i, \mathcal{C}_{G_{i-1}}) + d$$

  that yields the highest index value. For $d = 0$ this corresponds to finding the clustering with the best index value amongst all clusterings with minimal distance to the predecessor $\mathcal{C}_{G_{i-1}}$. Instead of the absolute threshold, a relative version can be formulated analogously.

- Given a weighting function $f(G_i, \mathcal{C}_{G_i}, \mathcal{C}_{G_{i-1}})$. For each graph G_i find a clustering $\mathcal{C}_{G_i}$ that maximizes f with respect to its predecessor $\mathcal{C}_{G_{i-1}}$.

3.3 Restriction to the Online Problem Definition

When dealing with a sequence of graphs or data in general, there are two concepts available that describe different aspects and are motivated differently as well. When developing algorithms for the Dynamic Graph Clustering Problem it is of importance to define what data is available at what time. The main versions are an *offline problem* definition and an *online problem* definition. The offline version assumes that all graphs are available at the time of solving the task. This means that the evolution of a dynamic graph is known in advance, thus not only the structure of one or more previous but also of all later timesteps can be taken into consideration when calculating a current optimal clustering. Common techniques for layouts but also clustering problems even interconnect the timesteps and compute on one large dataset. In contrast to that, the online problem definition assumes that for any timestep t_i all graphs and possibly also clusterings $G_j, \mathcal{C}_{G_j}$ with $j < i$ are known and can be incorporated, but none that are later in time. Thus it is unknown how the graph will evolve. Incrementally arriving data has to be considered at a time. Therefore an offline approach should at least perform as well as an online one. This thesis only deals with algorithms for the online problem version.

4 Index Maximization via Mathematical Programming

In this chapter it will be explained, how a mathematical program for modularity can be optimized. This is achieved by reducing its number of decision variables and constraints. Using the same model to describe a clustering that has been introduced in Section 2.5.2, it will be shown what parts of the original program formulation are redundant and how previously discovered knowledge about the behavior of the index that gets optimized can be exploited in order to improve the performance of the algorithm. In consideration of the improvements made, an objective function to maximize the significance performance index gets set up and compared to that for the modularity index.

4.1 Optimizing Variables and Constraints for the Modularity ILP

So far, for every (directed) pair of nodes (u, v) there is one variable X_{uv}, resulting in $vars = n^2 = |V|^2$ variables for the constructed ILP. The number d of constraints is strictly linked to the number of variables itself. It the sum of the number of constraints necessary to ensure reflexivity, symmetry and transitivity constraints and is given as follows. As the running time is exponential in d, reducing it promises a significant gain in actual performance.

$$d = \underbrace{n}_{reflexivity} + \underbrace{\frac{n(n-1)}{2}}_{symmetry} + \underbrace{3 \cdot \binom{n}{3}}_{transitivity}$$

4.1.1 Pruning Reflexivity

As a first step, note that the reflexivity constraints force a number of n variables to hold the constant value 1. By concept a node will always be in one cluster with itself. Together with the fact that the objective function of the ILP sums over all node pairs, it can easily be seen that the fraction of the index value contributed by loop edges of a node u is constant within all possible clusterings as it is linked to the constant decision variable X_{uu} only.

Lemma 4.1 *Given an ILP ilp_1 with the set of constraints $\mathcal{D} = \{D_1, \cdots, D_d\}$ and the objective function $f_{obj}(x_1, \cdots, x_n) = f'_{obj}(x_1, \cdots, x_n) + a$, with $x_1, \cdots, x_n$ being decision variables and a being constant. Another ILP ilp_2 with the same set of constraints $\mathcal{D}$ and the objective function $f'_{obj}(x_1, \cdots, x_n)$ will be maximal (minimal) with regard to the same set of decision variables iff ilp_1 is maximal (minimal).*

Therefore reflexive decision variables and reflexivity constraints are of no influence to the maximization problem, making it possible to neglect all reflexive pairs' variables completely. In this case the value of the objective function will be maximal iff the clustering has the maximum modularity possible for the particular graph, but it will no longer be equal to the modularity value itself.

4.1.2 Directed vs. Undirected Pairs

For every node pair u, v the two relations expressed by X_{uv} and X_{vu} have to be decided separately, though they are interlinked and actually equalized by the symmetry constraints. The modularity objective function sums over directed pairs, as it derives from the original modularity function, which calculates the expected coverage value $\mathbb{E}[cov](C)$ of a cluster C as follows. After pruning unnecessary terms of the equation it will no longer be equivalent to the actual expected value, thus it is called *exp-cov*(C).

$$\begin{aligned}
\textit{exp-cov}(C) &= \left(\frac{w(C)}{2 \cdot w(E)}\right)^2 \\
&= \left(\frac{\sum_{v \in C} w(v)}{2 \cdot w(E)}\right)^2 \\
&= \frac{(w(v_1) + \ldots + w(v_{|C|}))^2}{4 \cdot w(E)^2} \\
&= \sum_{(u,v) \in C^2} \frac{w(u) \cdot w(v)}{4 \cdot w(E)^2}
\end{aligned}$$

The summands for the pairs (u, v) and (v, u) are equal. If u is in the same cluster as v then, by concept, v is in the same cluster as u, and as the graph is undirected $w(u, v) = w(v, u)$. A general comparison of sets of undirected and directed node pairs leads to the following results.

Lemma 4.2 *Given a symmetric function $f(x, y) : \mathbb{R} \to \mathbb{R}$ with $f(x, y) = f(y, x)$ and a finite set of parameters $P \subset \mathbb{R}$.*

$$\sum_{(x,y) \in P} f(x, y) = 2 \cdot \sum_{\{x,y\}} f(x, y) + \sum_{x \in P} f(x, x)$$

In Section 4.1.1 it has been shown that reflexive node pairs do not need to be considered. Therefore the transition from directed node pairs to undirected node pairs within the objective function can be done as follows. The number of variables after this reduction is $vars = n \cdot (n - 1)/2$ and the constraints for symmetry can be dropped.

$$\begin{aligned}\text{obj}(G, \mathcal{C}_G) &= \frac{1}{2 \cdot w(E)} \cdot \sum_{(u,v) \in V^2, u \neq v} \left(w(u,v) - \frac{w(u) \cdot w(v)}{2 \cdot w(E)} \right) X_{uv} \\ &= \frac{1}{w(E)} \cdot \sum_{\{u,v\} \in V^2, u \neq v} \left(w(u,v) - \frac{w(u) \cdot w(v)}{2 \cdot w(E)} \right) X_{uv}\end{aligned}$$

4.1.3 Removing Constant Factors

In order to clarify the behavior of the ILP's objective function, constant factors within it can be removed by the means of a similar argument as given in Lemma 4.1.

Lemma 4.3 *Given an ILP ilp_1 with a set of constraints $\mathcal{D} = \{D_1, \cdots, D_d\}$ and the objective function $f_{obj}(x_1, \cdots, x_n) = a \cdot f'_{obj}(x_1, \cdots, x_n)$, with $x_1, \cdots, x_n$ being its decision variables and a being constant. Another ILP ilp_2 with the same set of constraints $\mathcal{D}$ and the objective function $f'_{obj}(x_1, \cdots, x_n)$ will be maximal (minimal) with regard to the same set of decision variables iff ilp_1 is maximal (minimal).*

The constant factor normalizing the index value by the total edge weight within the underlying graph can be removed and hence an equivalent expression of the original function is given as follows:

$$\text{obj}'(G, \mathcal{C}_G) = \sum_{\{u,v\} \in V^2, u \neq v} \left(w(u,v) - \frac{w(u) \cdot w(v)}{2 \cdot w(E)} \right) X_{uv}$$

4.1.4 Optimizing Modularity for Connected Components

In [2] it has already been stated that there is always a clustering with maximum modularity, in which no cluster induces a disconnected subgraph. If there was a disconnected cluster, splitting it into two single clusters would not alter the number of intra-cluster edges but might lower the penalty term deriving from the expected coverage value. According to this finding, the number of variables within the ILP can be further reduced, as no two nodes $u, v \in V$ that belong to different connected components of the graph $G = (V, E)$ need to be regarded. The variable $X_{u,v}$ would have to be set to 0 and hence it adds a constant term to the total modularity value, that can be ignored due to Lemma 4.1. With $\mathcal{H} = \{H_1, \ldots, H_k\}$ being the set of connected components of G, $H_i \cap H_j = \emptyset$ for $i \neq j$, $H_1 \cup \ldots \cup H_k = V$, and for $u \in H_i, v \in H_j$ $\{u, v\} \notin E$, the amount of variables and constraints is given as follows.

$$\begin{aligned} vars &= \sum_{H \in \mathcal{H}} \frac{|H| \cdot (|H| - 1)}{2} \\ constraints &= \sum_{H \in \mathcal{H}} \binom{0.5 \cdot |H| \cdot (|H| - 1)}{3} \end{aligned}$$

In fact, the optimization of each component can be done separately, as they do not have any other effect on each other than increasing the constants used for normalization. Thus the problem can be split up into the solution of $|\mathcal{H}|$ ILPs with even less decision variables and constraints, allowing the calculation of optimal modularity for arbitrarily sized graphs with bounded component sizes.

4.1.5 Satellite Constraints

Wagner et al. also point out the fact that any node with a degree of 1 will be in one cluster with the one node it is linked to, though they limit their statement to the unweighted case. The extra intra-cluster edge always adds more to the coverage value than the increase of the cluster's degree to the expected value. Therefore another type of decision variables has a constant value known in advance. For every node pair u, v with $\{u, v\} \in E$ and $deg(u) = 1 \vee deg(v) = 1$ the corresponding variable X_{uv} will have a final value of 1. For the weighted case, given a cluster C with intra-cluster edges weighing w_{intra} and inter-cluster connections weighing w_{inter}, and a singleton cluster (i.e. a single node) v with $deg(v) = 1$ connected to it, the contribution to the modularity value made by these two clusters is given by

$$\frac{w(C)}{w(E)} - \frac{(2 \cdot w_{intra} + w_{inter})^2 + w(v)^2}{4 \cdot w(E)^2} \quad .$$

Merging the two clusters changes this contribution to

$$\begin{aligned}
&\frac{w(C) + w(v)}{w(E)} - \frac{(2 \cdot w_{intra} + w_{inter} + w(v))^2}{4 \cdot w(E)^2} \\
=&\frac{w(C) + w(v)}{w(E)} - \frac{(2 \cdot w_{intra} + w_{inter})^2 + 2 \cdot (2 \cdot w_{intra} + w_{inter}) \cdot w(v) + w(v)^2}{4 \cdot w(E)^2} \quad .
\end{aligned}$$

The expected value of the new cluster rises due to the increase of the intra-cluster degree. The delta is given as follows:

$$\begin{aligned}
\frac{(2 \cdot w_{intra} + w_{inter}) \cdot w(v)}{2 \cdot w(E)^2} &< \frac{w(E) \cdot w(v)}{2 \cdot w(E)^2} \\
&= \frac{1}{2} \cdot \frac{w(v)}{w(E)} \\
&< \frac{w(v)}{w(E)}
\end{aligned}$$

Therefore the change in modularity due to merging the singleton and the cluster is

$$\frac{w(v)}{w(E)} - \frac{(2 \cdot w_{intra} + w_{inter}) \cdot w(v)}{2 \cdot w(E)^2} + \frac{w(v}{4 \cdot w(E)^2} > 0$$

and the two clusters will always be merged in an optimal clustering. Unfortunately the decision variable X_{uv} with the known value 1 cannot be pruned easily, as the information that two nodes are clustered together is needed by the other nodes of the same connected component when applying the transitivity constraints. From an engineering perspective, it is a possible solution to virtually combine the two nodes u, v to one that has a loop edge with weight $w(v)$ and to drop X_{uv} and all constraints referring to v from the ILP. Without altering the ILP's graph one still has the possibility to add another constraint forcing the assignment of X_{uv}. While this increases the theoretical complexity of the problem, ILP solvers will be able to identify the branch limitation and act accordingly.

4.2 An ILP Formulation of the Significance-Performance Index

The significance performance index as presented in Section 2.3.2 so far has only been applied in combination with a greedy algorithm analogue to Newman's greedy algorithm for modularity. In this section an integer linear program that optimizes its value will be developed. Its basis will be the same constraints as used for the original modularity ILP. Later on it will be discussed how the improvements described in Section 4.1 can be used for this ILP as well.

4.2.1 Objective Function for the Performance Index

In addition to the intra-cluster edges, the performance index also counts the non-existent inter-cluster edges (or even weakly linked node pairs for the weighted case). For the ILP formulation it is important to distinguish four possible cases. Two nodes can be linked and clustered together, linked and separated, disconnected but clustered together, and disconnected and in different clusters. For a node pair u, v, $E_{uv} : V \times V \to \{0, 1\}$ is 1 iff $\{u, v\} \in E$ (then $w(u, v)$ is the weight of the edge). Table 4.1 shows how to calculate the fraction of the performance index for the particular pair subject to their relations in the clustering and in the graph. It can be observed, that only two different terms are needed for the four cases and that only one of these can be non-zero at a time.

X_{uv}	E_{uv}	performance		ilp formulation
0	0	w_{max}	$=$	$(1 - X_{uv}) \cdot (w_{max} - w(u, v))$
0	1	$w_{max} - w(u, v)$	$=$	$(1 - X_{uv}) \cdot (w_{max} - w(u, v))$
1	0	0	$=$	$X_{uv} \cdot w(u, v)$
1	1	$w(u, v)$	$=$	$X_{uv} \cdot w(u, v)$

Table 4.1: The fraction of the performance value contributed by two nodes u, v and an equivalent ILP term subject to their relations X_{uv} and E_{uv}. Normalizing factors have been left out.

For a graph G and a clustering $\mathcal{C}_G$ an ILP formulation with respect to the known constraints has to sum over all node pairs as well and directly derives from the given terms and the original performance function. The reduction of constant factors cannot be done before composing the complete significance-performance function. Constant terms will be deleted later on.

$$\begin{aligned}
\textit{perf}(G,\mathcal{C}_G) &= \frac{w(\mathcal{C}_G) + w_{max}\cdot \overline{m}(\mathcal{C}_G)^c + w_{max}\cdot \overline{m}(\mathcal{C}_G) - \overline{w}(\mathcal{C}_G)}{0.5\cdot n(n-1)\cdot w_{max}} \\
&= \frac{2}{n(n-1)\cdot w_{max}} \sum_{\{u,v\}\in V^2} \Big((1-X_{uv})\cdot(w_{max}-w(u,v)) + X_{uv}\cdot w(u,v)\Big) \\
&= \frac{2}{n(n-1)\cdot w_{max}} \sum_{\{u,v\}\in V^2} \Big(X_{uv}\cdot(2\cdot w(u,v) - w_{max}) + w_{max} - w(u,v)\Big)
\end{aligned}$$

4.2.2 ILP Formulation of the Expected Performance Value

The original formulation of the expected value of the performance index as seen in Section 2.3.2 has to be modified in such a way, that the ILP does not have to sum over all intra- and inter-cluster edges. It has to apply the same term to all node pairs instead, a distinction between those in the same cluster and in different clusters has to be done via the decision variable itself. Again, the weighted case considers light edges e between clusters with $w_{max} - w(e)$. As for the performance index's formulation, constant terms and factors will be dealt with when composing the ILP's complete objective function.

$$\begin{aligned}
&\textit{exp-perf}(G,\mathcal{C}_G) \\
=&\frac{\frac{1}{2\cdot w(E)}\cdot\sum_{C\in\mathcal{C}}\sum_{(u,v)\in C^2} w(u)\cdot w(v)}{n(n-1)\cdot w_{max}} + \frac{\sum_{C\in\mathcal{C}}\sum_{C'\in\mathcal{C}\setminus C}\sum_{(u,v)\in C\times C'}(w_{max} - \frac{w(u)\cdot w(v)}{2\cdot w(E)})}{n(n-1)\cdot w_{max}} \\
=&\sum_{(u,v)\in V^2}\Bigg(\frac{w(u)\cdot w(v)}{2\cdot w(E)\cdot n(n-1)\cdot w_{max}}\cdot X_{uv} + \\
&\left(w_{max} - \frac{w(u)\cdot w(v)}{2\cdot w(E)\cdot n(n-1)\cdot w_{max}}\right)\cdot(1-X_{uv})\Bigg) \\
=&\frac{1}{n(n-1)\cdot w_{max}}\cdot\sum_{(u,v)\in V^2}\left(X_{uv}\cdot\left(\frac{w(u)\cdot w(v)}{w(E)} - w_{max}\right) + w_{max} - \frac{w(u)\cdot w(v)}{2\cdot w(E)}\right)
\end{aligned}$$

4.2.3 Objective Function for the Significance-Performance ILP

As in Section 2.3.2, the objective function for the significance-performance index is equal to the difference of the objective function for the performance index and that for the expected value. They differ in the pairs that are considered as one uses directed node pairs $(u,v)\in V^2$ and the other one uses undirected pairs $\{u,v\}\in V^2$. The approaches shown in Section 4.1 will be applied in order to ease the equation.

$$
\begin{aligned}
&\operatorname{perf}(G, \mathcal{C}_G) - \operatorname{exp-perf}(G, \mathcal{C}_G) \\
=&\frac{2}{n(n-1) \cdot w_{max}} \sum_{\{u,v\} \in V^2} \Big(X_{uv} \cdot (2 \cdot w(u,v) - w_{max}) + w_{max} - w(u,v) \Big) \\
&- \frac{1}{n(n-1) \cdot w_{max}} \cdot \sum_{(u,v) \in V^2} \left(X_{uv} \cdot \left(\frac{w(u) \cdot w(v)}{w(E)} - w_{max} \right) + w_{max} - \frac{w(u) \cdot w(v)}{2 \cdot w(E)} \right)
\end{aligned}
$$

Constant terms within the sums (that are independent of the decision variables) can be removed according to lemma 4.1.

$$
\begin{aligned}
=&\frac{2}{n(n-1) \cdot w_{max}} \sum_{\{u,v\} \in V^2} \Big(X_{uv} \cdot (2 \cdot w(u,v) - w_{max}) \Big) \\
&- \frac{1}{n(n-1) \cdot w_{max}} \cdot \sum_{(u,v) \in V^2} \left(X_{uv} \cdot \left(\frac{w(u) \cdot w(v)}{w(E)} - w_{max} \right) \right)
\end{aligned}
$$

As seen for the modularity objective function, all reflexive decision variables X_{uu} are of the constant value 1. Therefore their contributions are constant, making it possible to rewrite the sum of directed pairs for the expected performance value according to Lemma 4.2 without adding extra summands. Reflexive summands for the performance value can be pruned as well.

$$
\begin{aligned}
=&\frac{2}{n(n-1) \cdot w_{max}} \sum_{\{u,v\} \in V^2, u \neq v} \Big(X_{uv} \cdot (2 \cdot w(u,v) - w_{max}) \Big) \\
&- \frac{2}{n(n-1) \cdot w_{max}} \cdot \sum_{\{u,v\} \in V^2, u \neq v} \left(X_{uv} \cdot \left(\frac{w(u) \cdot w(v)}{w(E)} - w_{max} \right) \right)
\end{aligned}
$$

A further reduction of constant factors in accordance to Lemma 4.3 results in the final objective function for the significance performance ILP. It is given as follows.

$$\begin{aligned}
obj(G,\mathcal{C}_G) &= \sum_{\{u,v\}\in V^2, u\neq v} \left(X_{uv}\cdot(2\cdot w(u,v) - w_{max})\right) \\
&\quad - \sum_{\{u,v\}\in V^2, u\neq v} \left(X_{uv}\cdot\left(\frac{w(u)\cdot w(v)}{w(E)} - w_{max}\right)\right) \\
&= \sum_{\{u,v\}\in V^2, u\neq v} \left(X_{uv}\cdot\left(2\cdot w(u,v) - w_{max} - (\frac{w(u)\cdot w(v)}{w(E)} - w_{max})\right)\right) \\
&= \sum_{\{u,v\}\in V^2, u\neq v} X_{uv}\cdot\left(2\cdot w(u,v) - \frac{w(u)\cdot w(v)}{w(E)}\right)
\end{aligned}$$

4.2.4 Optimizing the Significance Performance ILP

The significance performance ILP can also be optimized with respect to the amount of constraints and variables needed. The reduction to undirected node pairs is a first step. An optimal clustering for the significance performance index also does not allow disconnected clusters, therefore the same modifications considering single connected components as seen for modularity can be applied. The satellite constraints behave identically and the same tradeoff between the temporary modification of the graph and the number of constraints and variables has to be made. The final number of variables and constraints needed is identical to that for the modularity index.

4.2.5 Comparison with the Modularity ILP

Comparing the objective functions for the modularity ILP (given in Section 4.1.3) and for the significance performance ILP (given in Section 4.2.3), it can easily be seen that even in case of optimizing the index values of a weighted graph the calculation is equivalent. For any given graph G and two clusterings $\mathcal{C}_1, \mathcal{C}_2$ the following equation holds.

$$mod(G,\mathcal{C}_1) > mod(G,\mathcal{C}_2) \Leftrightarrow perf(G,\mathcal{C}_1) > perf(G,\mathcal{C}_2)$$

In particular, it means that a clustering with optimal modularity for any graph will also yield an optimal significance performance value. Still, both indices are not equivalent in total. As the actual values differ so might approximation schemes and especially approximation thresholds.

As a consequence of this finding, whenever mathematical programming for one of the indices is applied, the other one does not need to be looked at. Within this thesis only modularity will be considered, as it is more widely studied.

5 Bi-Criterial Mathematical Programming

In this chapter it will be explained, how a single integer linear program can be used to solve a single step of the online version of the Dynamic Graph Clustering Problem, i.e. to find a clustering with a high modularity value and a small distance to the partitioning of the previous timestep. Two competing criteria will be incorporated into the ILP, in order to consider index quality on the one hand and clustering stability on the other hand.

5.1 General Approach

To evaluate known algorithms for clustering dynamic graphs, in [4] Chakrabarti et al. suggest a simple framework, composed of a measure for the clustering's *snapshot quality* $sq(C_t)$ and a function calculating the *history cost* $hc(C_t, C_{t-1})$ with regard to a previous snapshot's clustering. A good dynamic clustering is supposed to have high quality for all timesteps (i.e. snapshots) and small distance between consecutive timesteps, according to Chakrabarti and also the problem given in Section 3.2. The difference between clustering quality and distance is supposed to be high resulting in the need to maximize the following equation.

$$\text{quality}(\mathcal{C}_{\mathcal{G}}) = \sum_{t=1}^{n} \Big(sq(C_t) - b \cdot hc(C_t, C_{t-1}) \Big)$$

The snapshot quality function $sq(C)$ generalizes the index measures introduced in Section 2.3 and the history cost takes the place of the distance measures for clusterings as seen in Section 2.4. The two parameters tend to be opposing, the *weighting factor* b allows to emphasize either one of them. Chakrabarti et al. compare evolutionary versions of traditional heuristics for the graph clustering problem and extend them with the formulation of specific history cost.

It is an even more general approach to apply the framework to independent measures, allowing a free combination of quality and distance functions. This creates the need for an algorithm to optimize those features. In Chapter 4 it has not only been shown how to optimize indices' values by integer linear programming, but also how to optimize different features simultaneously (i.e. index value and expected index value) by combining their objective functions. Extending the ILP objective function for e.g. the modularity index by an ILP formulation for a clustering distance function in a subtractive manner directly corresponds to the suggested method and allows to explicitly optimize the stability of the mental map as well. This approach will be examined within the following sections.

5.2 ILP Formulations for Clustering Distance Measures

The most important restriction to the use of a combination of clustering indices' ILPs and distance measures' formulations follows from the constraints model guaranteeing correct clusterings. The intuitive use of pairwise node handling allows the easy observation of node-node relations but lacks a proper representation of single clusters. While sums of cluster sizes and even sums of matching nodes between all clusters still can easily be calculated, more complicated operations like intersecting single clusters needed for matching measures and entropy measures are hard to implement. To demonstrate the capabilities of the approach, a formulation of the Rand measure (see Section 2.4.2) will be developed and used within the framework.

5.2.1 An ILP Using the Rand Distance Measure

The Rand measure as described in Section 2.4.2 is a counting pair measure. It counts the number of node pairs that were either clustered together or clustered apart within both of the clusterings. Within a dynamic clustering these clusterings are ordered with respect to a timeline. Comparing two clusterings $\mathcal{C}_t, \mathcal{C}_{t-1}$ for the two consecutive timesteps $t, t-1$ and the same underlying graph G containing the nodes u, v, let X_{uv} be the decision variable for $\mathcal{C}_t$ and Y_{uv} be the decision variable for $\mathcal{C}_{t-1}$. The non-normalized term deriving from each node pair's relation can be seen in Table 5.1.

X_{uv}	Y_{uv}	fraction		ILP formulation
0	0	1	$=$	$(1 - X_{uv}) \cdot (1 - Y_{uv})$
0	1	0	$=$	0
1	0	0	$=$	0
1	1	1	$=$	$X_{uv} \cdot Y_{uv}$

Table 5.1: The fraction of the Rand measure value contributed by two nodes u, v and an equivalent ILP term subject to their relations X_{uv} and Y_{uv}. Normalizing factors have been left out.

It can be seen that an additive combination of all ILP formulations is possible, as in all cases but the one to be applied, one of the factors will be 0. Therefore the straightforward ILP formulation of the Rand distance measure is given as follows.

$$\begin{aligned}
rand_{ILP}(\mathcal{C}_{t-1}) &= 1 - \frac{2}{n\cdot(n-1)}\cdot \sum_{\{u,v\}\in V^2, u\neq v} \Big(X_{uv}\cdot Y_{uv} + (1-X_{uv})\cdot(1-Y_{uv})\Big)\\
&= 1 - \frac{2}{n\cdot(n-1)}\cdot \sum_{\{u,v\}\in V^2, u\neq v} \Big(2\cdot X_{uv}\cdot Y_{uv} - X_{uv} - Y_{uv} + 1\Big)\\
&= -\frac{2}{n\cdot(n-1)}\cdot \sum_{\{u,v\}\in V^2, u\neq v} \Big(2\cdot X_{uv}\cdot Y_{uv} - X_{uv}\Big)\\
&= -\sum_{\{u,v\}\in V^2, u\neq v} \Big((2\cdot Y_{uv}-1)\cdot X_{uv}\Big)
\end{aligned}$$

As for modularity, reflexive node relations are constant and of no influence to the optimization. Additionally the last line of the equation shows the expression without the additive terms that do not depend on X_{uv} and the constant normalization factor. This includes the Y_{uv} decision variables, as they refer to the previous timestep and are supposed to be known in advance. The combination of modularity and the Rand distance measure can be done by subtracting the respective ILP objective functions. It has to be taken into consideration that constant factors stripped from the function either need to be regarded again or compensated by the weighting factor b. The objective function for the combination of the modularity index and the Rand distance measure is given as follows.

$$\begin{aligned}
&bicrit(G_t, \mathcal{C}_{t-1})\\
&= mod_{ILP}(G_t) - b\cdot rand_{ILP}(\mathcal{C}_{t-1})\\
&= \sum_{\{u,v\}\in V^2, u\neq v} \Big(w(u,v) - \frac{w(u)\cdot w(v)}{2\cdot w(E)}\Big) X_{uv} + b\cdot \sum_{\{u,v\}\in V^2, u\neq v} (2\cdot Y_{uv}-1)\cdot X_{uv}\\
&= \sum_{\{u,v\}\in V^2, u\neq v} \Big(w(u,v) - \frac{w(u)\cdot w(v)}{2\cdot w(E)} + 2\cdot b\cdot Y_{uv} - b\Big)\cdot X_{uv}
\end{aligned}$$

5.2.2 Formulating Other Distance Measures

In order to formulate any of the other given distance measures for clusterings within the proposed ILP clustering model, it needs to be rewritten so that either single node-node relations or calculations over the complete nodeset are performed. Within the objective function of the ILP only one decision variable can be used per additive term, therefore all demands have to be casted into its scalar factors. This is achievable for the pair counting measures (Jaccard measure and Fowlkes-Mallows measure) analogously to the Rand measure's implementation.

The calculations of a more algorithmic approach as used for the MaximumMatch measure, that computes single cluster relations and matches single clusters, is more demanding. It involves the identification of single clusters' sizes as well as a memory what clusters have been used during the calculation. In its current form it cannot be incorporated into the ILP. The entropy measures like the Fred and Jain measure also need to calculate terms including the sizes of single clusters as well as matchings between those. They suffer from the same effects.

5.3 Scaling Between Index and Distance Measure

In Section 5.2.1 the weighting factor b has been introduced, making it possible to emphasize either one of the framework's parts, the snapshot quality measure or the history cost function. In order to use the overall approach for automated graph calculations, this weighting factor either has to be fix or there must be a way to determine it systematically.

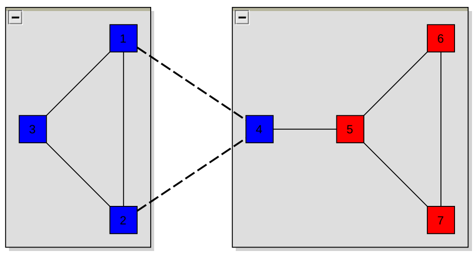

Figure 5.1: A graph showing two possible clusterings. The grouping is optimal without considering the dashed edges, otherwise the coloring depicts the optimal decomposition with regard to modularity. Inserting the dashed edges only leads to a switch between the clusterings, if $b < 0.3409$ holds.

As an example, Figure 5.1 shows a graph in two possible states. The grouping marks the optimal clustering with regard to the modularity index if the dashed edges between the clusters are not yet inserted. The coloring on the other hand depicts the optimal clustering if the new edges are regarded. The index then yields an optimal value of 0.3642 while the old clustering would only achieve 0.2716, this is a delta of 0.0926. Switching from one clustering to the other would lead to a Rand distance of 0.2857. Node 4 will switch clusters only if the delta in modularity is bigger than the implied distance. Thus the switch will only occur if $0.0926 > b \cdot 0.2716 \Leftrightarrow b < 0.3409$ is met.

The same setting is shown again in Figure 5.2. The graph has been extended by the green cluster spanning four additional nodes. As those have influence on the modularity index (see resolution limit, Figure 3.1), they do have influence on the distance measure as well. The optimal clustering without the dashed edges is marked by the grouping of the nodes and has an index value of 0.66. On the other hand, if the edges are considered, the coloring of the graph marks the optimal clustering with an index value of 0.559 while the grouping only achieves 0.4861. This results in the delta being 0.0729. The Rand distance between the two clusterings is 0.1091. Therefore, analogously to the calculations made for the graph in Figure 5.1, the switch between the clusterings will take place, if $0.0729 > b \cdot 0.1091 \Leftrightarrow b < 0.6682$ holds. It can be seen that the weighting factor b can be significantly higher, resulting in exactly the same change. The overall graph stays more similar to its original clustering, small changes in the structure do not have as much influence.

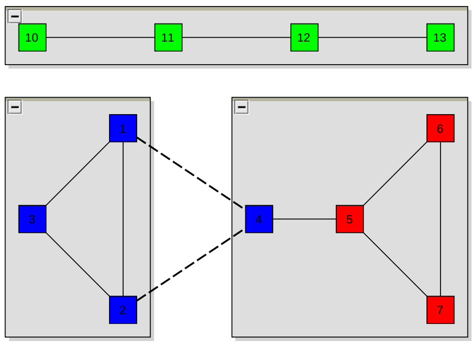

Figure 5.2: A graph showing two possible clusterings. The grouping is optimal without considering the dashed edges, otherwise the coloring depicts the optimal decomposition with regard to modularity. Inserting the dashed edges only leads to a switch between the clusterings, if $b < 0.6682$ holds.

An important observation here is, that the modularity index and the distance measure do not scale evenly. In both cases changes of few nodes between clusters will lead to a smaller change if the graph gets bigger. In case of modularity this is linked to the assumption that edges are evenly split amongst clusters and that the (connected) clusters' sizes do not vary proportionally to the number of the graph's nodes. If the induced change to the number of intra- and inter-cluster edges is small, then intra-cluster degrees do not change much either. Having said that, the significance paradigm itself dampens the effect. Coming from an optimal clustering the change in coverage induced by few additional edges minus the expected value of the new optimal clustering tends to be smaller for bigger clusterings as well. Therefore the scaling within the index itself results in a similar modularity delta for the two given examples. As the same amount of node-node relations (neighbors in the node's old cluster and the node's new cluster) is affected by the switch but the overall amount of relations grows quadratically in the total number of nodes, the distance measure's value shrinks significantly. Thus, choosing a fixed weighting factor allows more changes in bigger graphs.

Note that nodes within small clusters are generally more likely to change to another (small) cluster due to new connections, than nodes within large clusters. This is based on the fact, that the overall amount of affected relations is relatively smaller and the fraction of intra-cluster edges gained tends to be high in comparison to the fraction of inter-cluster edges created.

5.4 Experimentally Evaluating Bi-Criterial Mathematical Programming

5.4.1 General Setup

In order to evaluate the presented bi-criterial clustering method, a portion of the email graph is clustered. The email graph is a real dynamic network showing the email communication between a group of people. Sending an email results in a connection being established between sender and recipients (the nodes of the graph) represented by creating an edge between them or increasing the weight of an already existing one. The contact due to email communication is subject to a timeout that has to be defined per experiment. For further information see the explanation in Section 6.8.1.

For the experiments with the bi-criterial approach and the email graph, several parameters can be modified. Within the framework, it has to be decided, what weighting factor b, influencing the ratio of modularity and Rand distance measure, shall be applied. Besides, the number of single edge modifications (i.e. sent emails, timed out connections) before the graph is re-optimized as well as a lifetime for established connections has to be chosen. For the following evaluation, the email graph for September 2007 has been clustered with weighting factors $b \in \{0.1, 0.2, \cdots, 4.9, 5.0\}$, for different step sizes $s \in \{5, 10, 20\}$, and with a connection lifetime of 24 hours. The size of the graph is around 50 nodes, a snapshot is shown in Figure 5.3. For the sake of completeness, further figures showing the complete evolution during the first three weeks of the month and for the different parameters are given in Appendix D.2.

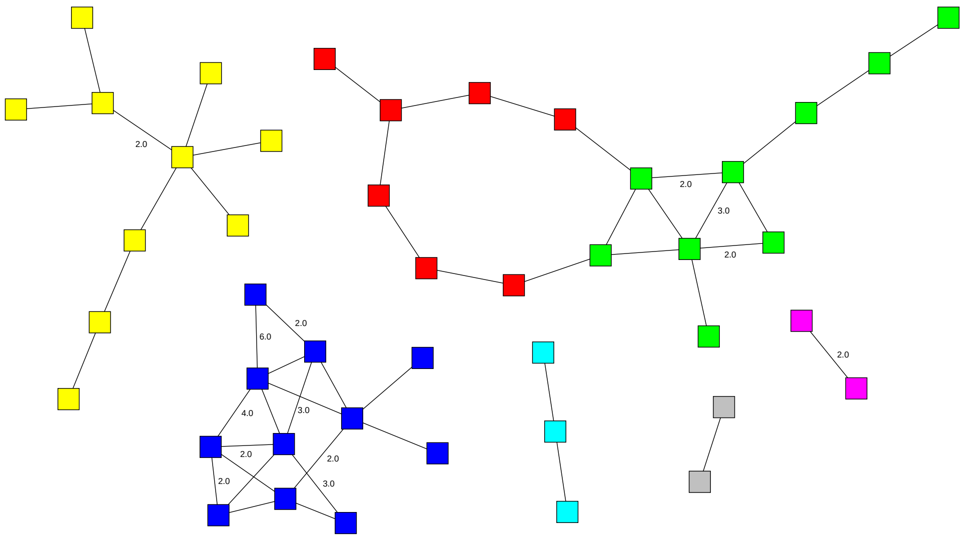

Figure 5.3: A snapshot of the email graph for September 2007 containing 3 professor's chairs. The connection lifetime is 24 hours. 160 atomary modifications haven taken place.

5.4.2 Analysis

The analysis of the conducted experiments focuses on the influence of the two parameters that are of concern to the framework but not the data. It will be analyzed, what influence the weighting factor b and the step size s have on the clustering quality and the distance measures.

Influence of the Weighting Factor

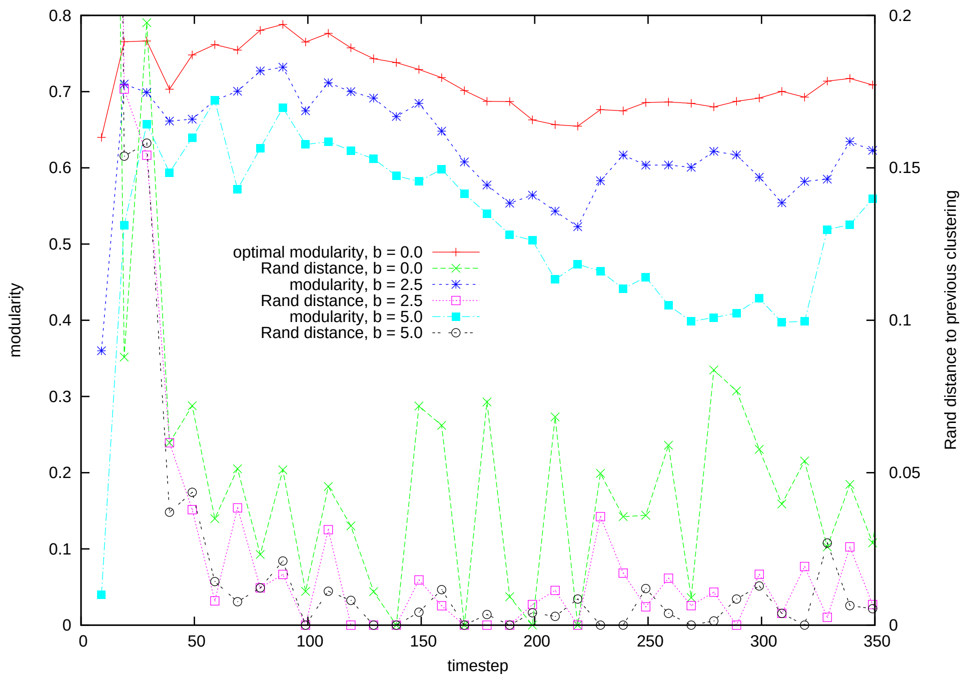

Figure 5.4: The graphs show the modularity and Rand distance measure for the clustering of a fraction of the email graph with a connection lifetime of 24 hours. Different weighting factors have been applied, the step size is 10.

Figure 5.4 shows the modularity curves and the distance measure values (with regard to the previously clustered timestep) for the first days of September 2007 and the weighting factors $0.0, 2.5, 5.0$. The index curve for b set to 0.0 (red) shows the optimal modularity values, thus the corresponding Rand distance (green) has not been optimized explicitly. By concept, the best index values as well as the highest distance measure values are achieved. The modularity values for b set to 2.5 and 5.0 shrink significantly, as do their corresponding distance measures values. It can very easily be seen, that a higher weighting of the Rand distance leads to less distance measured to previous clusterings, but also lowers the modularity value. This is explained by the fact that necessary optimizations in order to construct optimal modularity partitionings cannot be executed, as minimizing the history cost is favored and hence nodes are anchored to their current clusters. This underlines the assumption, that the two criteria are opposing. Also, it shows that the balancing by the weighting factor has the anticipated effect and allows to explicitly decide, what quality at the cost of what fluctuation in the clustering structure shall be targeted. Note that the general characteristics of the curves stay similar.

Influence of the Step Size

Figure 5.5 and Figure 5.6 show the same type of graph as seen in Figure 5.6 for the step sizes of 5 and 20. The sampling points for the optimal modularity curves (red) that are included in more than one plot stay unchanged, thus the curves are nearly congruent with one another. In contrast to that, the corresponding distances to previous clusterings are noticeably higher for the larger step size of 20 and lower for the smaller step size of 5. Due to more modifications within 20 timesteps, the next snapshot of the dynamic graph will most likely exhibit more differences to its predecessor than a snapshot taken after only 5 updates. Therefore, a higher Rand distance is measured though the same quality values are achieved. It is clearly visible, that the curve for a weighting factor b of 2.5 and a step size s of 5 features several local maxima, which the curve for a step size s of 20 lacks. This leads to the observation, that the influence of the weighting factor b grows if it is combined with a larger step size s.

On the other hand, it can easily be observed, that the curves for the highest weighting factor of $b = 5.0$ do not vary as much, though the distance measures' values differ significantly. This phenomenon is explained by the observation that structural differences due to insertion and deletion of nodes cannot be compensated by anchoring. Even if the algorithm solely tries to optimize the distance to the previous clustering, it cannot achieve the optimal Rand distance value of 0. For arbitrarily high values of b, the only optimizations with regard to the quality index are accomplished by assigning new nodes to existing or new clusters.

Note that a smaller step size s is not able to guarantee higher modularity values. For example, the end of the modularity curve for $b = 5.0$ for the step size $s = 20$ in Figure 5.6 yields a higher value than the modularity curve for the step size $s = 5$ depicted in Figure 5.5. This can happen due to smaller modifications being necessary in order to optimize modularity, e.g. if the change in structure after 20 modifications reverts the effects of a change after only 10.

Additionally, the influence of the step size is subject to the overall size of the graph. Within small networks, the possible distance to a previous snapshot after a fixed number of updates is bigger than in large networks.

5.4.3 Applicability

The implemented framework allows to explicitly optimize the two criteria influencing the Dynamic Graph Clustering Problem. Nevertheless, the fairly good and predictable results in quality and history cost come at the price of applicability to large graphs. The information contained within existing clusterings is not exploited in order to reduce the complexity of a problem instance and the ILP used is an extension of the one given in Chapter 4, that derived from the original version described in Section 2.5.2. Thus, solving the problem is bounded to the same conditions. It is $\mathcal{NP}$-hard and ILP solvers will not be able to compute the problem for connected components bigger than 50 to 80 nodes without enormous demands to runtime.

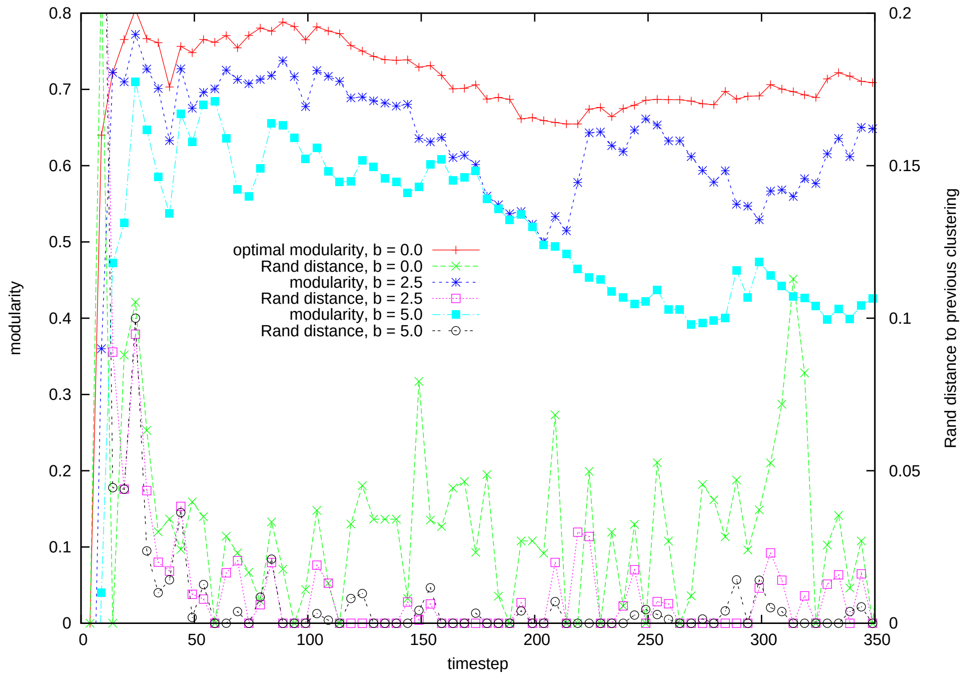

Figure 5.5: The graphs show the modularity and Rand distance measure for the clustering of a fraction of the email graph with a connection lifetime of 24 hours. Different weighting factors have been applied, the step size is 5.

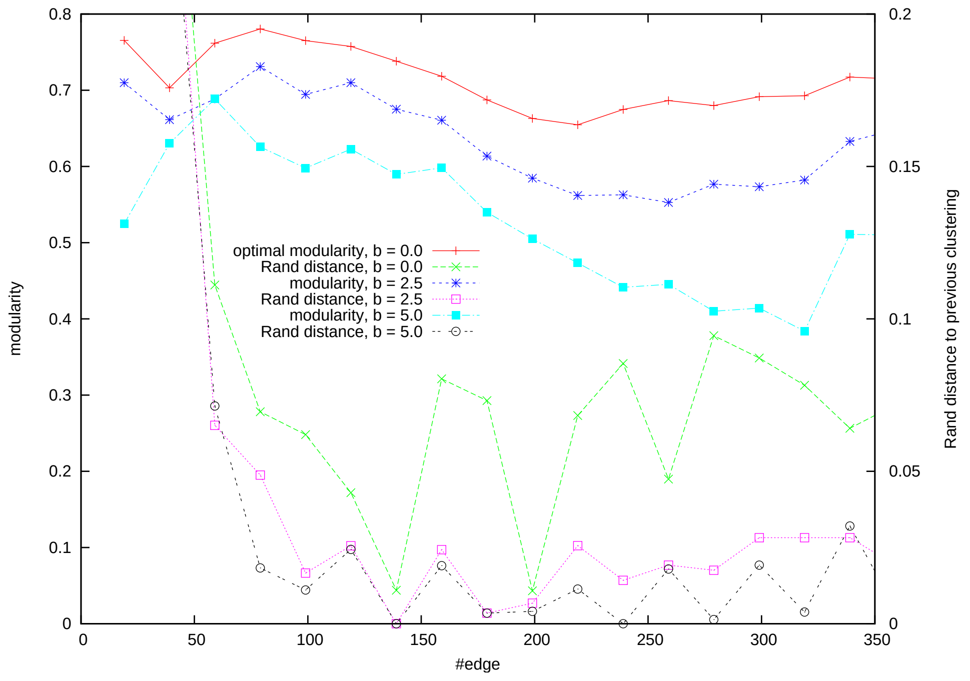

Figure 5.6: The graphs show the modularity and Rand distance measure for the clustering of a fraction of the email graph with a connection lifetime of 24 hours. Different weighting factors have been applied, the step size is 20.

6 Partial Mathematical Programming

In this chapter another approach to solve the online version of the Dynamic Graph Clustering Problem is presented. Again, the ILP constraints model describing clusterings (as introduced in Section 2.5.2) will be used, in order to formulate an objective function, that tries to optimize two contradicting criteria at the same time. It is designed to locally re-optimize a clustering with respect to an index due to modifications in the graph. Like this, the quality criterion will be considered explicitly while the distance function is kept low implicitly.

6.1 General Approach

The key idea behind the approach of partial mathematical programming is the local optimization after a single modification in the graph. Such a modification is atomic with regard to modularity if it involves one single edge only and does not modify its weight by more than 1. This modification still might lead to non-local changes in the optimal clustering. But as those changes are due to rippling effects (i.e. a node is reassigned to a different cluster, therefore a node from that cluster is assigned to another cluster and so on) or the resolution limit, their effect on the index value itself is supposed to be marginal. Of course the insertion, deletion or manipulation of an edge has direct consequences for the nodes it connects. On the one hand intra-cluster density might be enhanced, on the other hand yet unclustered nodes might be connected (closer) and changed node-node distances could therefore influence the semantics of the underlying graph. Partial mathematical programming is based on the assumption that this semantic change is primarily local.

After a modification of an edge it has to be decided how it influences the clustering around it. Therefore a predefined type of neighborhood of nodes around that edge is examined (several possible strategies to choose this subset of V will be given in Section 6.4). All nodes within this neighborhood are extracted from their respective clusters, they are then considered to be unclustered. Similar to the ILP described in Section 2.5.2, ILP constraints and a new objective function are constructed. Within this ILP it is decided how unclustered nodes are reassigned to the remaining clusters (i.e. by clustering them with all contained nodes) or grouped amongst each other. Therefore, all nodes of an existing cluster are represented by the same decision variables forming node-cluster relations. By definition it is impossible for existing clusters to merge. With respect to the conserved clusters and the modularity index, the resulting clustering will be optimal. The global mental map of the clustering is preserved implicitly, as all components beyond the freed fraction of the clustering stay unchanged.

Figure 6.1 shows a small graph and the manipulation of its clustering due to the insertion of an additional edge between nodes 0 and 3. The grouping marks the current clustering within each step, while the node colors preserve the initial clustering. At the beginning both are optimal.

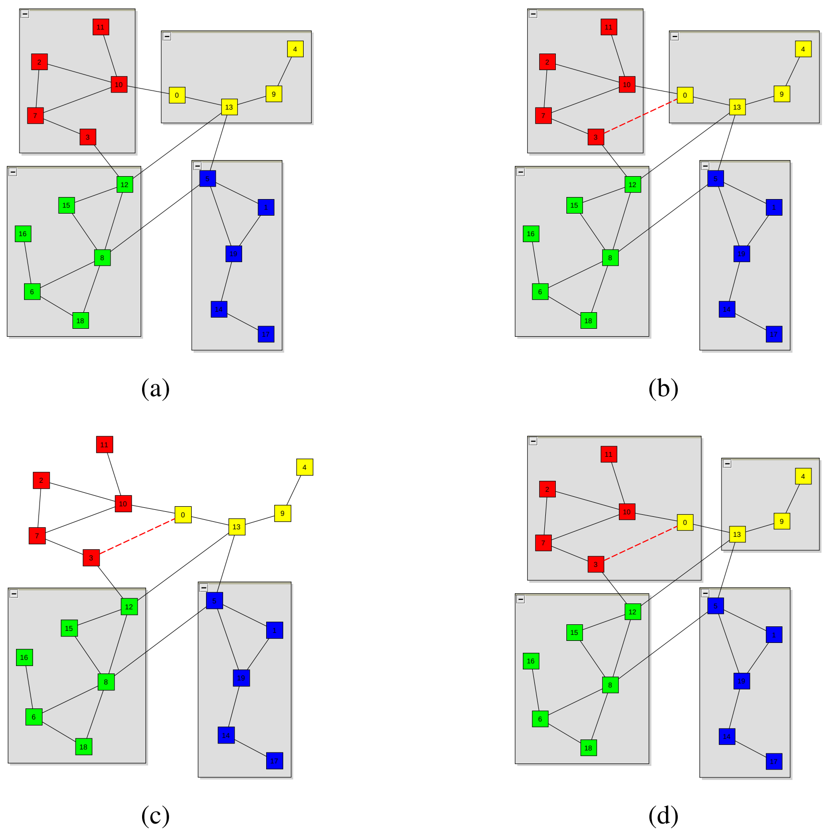

Figure 6.1: A graph gets locally optimized via partial mathematical programming after the insertion of an additional edge. The coloring always marks the initial clustering, while the grouping marks the current clustering of each step.

In this example the clusters of both nodes adjacent to the new edge get dissolved resulting in 9 unclustered nodes and two clusters that will remain after the update. Figure 6.1(c) shows the current clustering before the calculation of the partial ILP. The fourth graph shows the new clustering in which node 0 switched clusters. The resulting clustering actually is globally optimal in this case.

6.2 Formulating a Partial ILP

The formulation of a partial ILP to optimally (re-)assign a fraction of the graph's nodes to a given partial clustering involves the construction of transitivity constraints as well as an objective function spanning decision variables for node-node and also node-cluster relations. For a given graph $G = (V, E)$ with a given partial clustering $\mathcal{C} = \{C_1, \cdots, C_n\}$ with $C_1 \cup \ldots \cup C_n = V_{\mathcal{C}} \subset V$, a non-empty set $V_u \subset V$ of unclustered nodes and with $V_{\mathcal{C}} \cup V_u = V$, these decision variables can be divided into the two sets $\mathcal{X}_{NN}, \mathcal{X}_{NC}$. For every tuple within $V_u \times V_u$ there will be a decision variable within $\mathcal{X}_{NN}$, for every tuple within $V_u \times \mathcal{C}$ there will be one in $\mathcal{X}_{NC}$. For all triplets within V_u^3 the known transitivity constraints are maintained, guaranteeing a correct clustering amongst all momentarily unclustered nodes. Analogously to those, for all triplets within $V_u^2 \times \mathcal{C}$ and $V_u \times \mathcal{C}^2$ there are constraints to guarantee that nodes that are clustered together get assigned to at most one existing cluster that has to be identical for both. For the partial ILP, there will be the following constraints.

$$\forall \{u,v,w\} \in V_u^3 : \begin{cases} X_{uv} + X_{vw} - 2 \cdot X_{uw} \leq 1 \\ X_{uv} + X_{uw} - 2 \cdot X_{vw} \leq 1 \\ X_{uw} + X_{vw} - 2 \cdot X_{uv} \leq 1 \end{cases}$$

$$\forall \{u,v,C\} \in V_u^2 \times \mathcal{C} : \begin{cases} X_{uv} + X_{vC} - 2 \cdot X_{uC} \leq 1 \\ X_{uv} + X_{uC} - 2 \cdot X_{vC} \leq 1 \\ X_{uC} + X_{vC} - 2 \cdot X_{uv} \leq 1 \end{cases}$$

$$\forall \{u,C_1,C_2\} \in V_u \times \mathcal{C}^2 : X_{uC_1} + X_{uC_2} \leq 1$$

The extension of the modularity ILP's objective function has to consider the consequences of adding a node to an existing cluster. While the scalars for unclustered nodes are identical to those used for the objective function defined in Chapter 4, the scalars for decision variables within $\mathcal{X}_{NC}$ have to sum over all relations between an unclustered node and all nodes within a particular cluster. This reflects the fact that an unclustered node is either clustered with all nodes within one cluster or with none. By definition it is not possible for two existing clusters to merge. The objective function for the partial modularity ILP is given as follows.

$$\begin{aligned} obj_{pILP}(G,\mathcal{C}) = & \sum_{X_{uv} \in \mathcal{X}_{NN}} \left(w(u,v) - \frac{w(u) \cdot w(v)}{2 \cdot w(E)} \right) X_{uv} \\ & + \sum_{X_{uC} \in \mathcal{X}_{NC}} \left(\sum_{c \in C} \left(w(u,c) - \frac{w(u) \cdot w(c)}{2 \cdot w(E)} \right) \right) X_{uC} \end{aligned}$$

In order to reduce the number of variables, the techniques described in Chapter 4 can be used. The number x of variables needed for the partial ILP depends on the number of unclustered nodes $|V_u|$ and existing clusters $|\mathcal{C}|$. The number of constraints d depends on the number of variables. Additionally, only clusters that have an edge to any of the unclustered nodes, need to be considered. It is not possible, that a node gets assigned to another cluster, as this would results in its members being disconnected. Given the number of neighboring clusters c_n, the maximum numbers of constraints and variables are given as follows.

$$
\begin{aligned}
x =& \underbrace{0.5 \cdot |V_u| \cdot (|V_u| - 1)}_{|\mathcal{X}_{NN}|} + \underbrace{|V_u| \cdot c_n}_{|\mathcal{X}_{NC}|} \\
\leq& 0.5 \cdot |V| \cdot (|V| - 1)
\end{aligned}
$$

$$
\begin{aligned}
d =& \underbrace{3 \cdot \binom{|V_u|}{3}}_{NNN-transitivity} + \underbrace{3 \cdot \binom{|V_u|}{2} \cdot c_n}_{NNC-transitivity} + \underbrace{|V_u| \cdot \binom{c_n}{2}}_{single\ cluster\ association} \\
\leq& 3 \cdot \binom{n}{3}
\end{aligned}
$$

6.3 Complexity of the ILP

Solving the modularity ILP is $\mathcal{NP}$-hard, and enumerating possible solutions is exponential in the number of decision variables needed. By deciding which nodes will be extracted from their clusters before a re-optimization of a clustering is calculated with the ILP given in Section 6.2, the partial ILP approach allows to directly influence the needed number of decision variables. Assuming that the number of clusters adjacent to unclustered nodes is bounded by a constant c_n for graphs to be clustered, the overall number x of variables to be considered in order to solve the problem is constant as well. In [21] an algorithm to solve ILPs with a fixed number of variables in polynomial time is given (the polynomial is of a degree exponential in the number of variables). For the partial ILP this possibility becomes clear, as a constant number of binary decision variables x leads to a constant number of 2^x potentially optimal solutions for any given size of the underlying graph. Therefore, the partial ILP approach allows to cluster dynamic graphs with arbitrary size, as long as the number of decision variables $|\mathcal{X}_{NN}| + |\mathcal{X}_{NC}|$ does not rise above the capabilities of the employed solver.

6.4 Freeing Nodes

After modifying an edge $e = \{u, v\}$ of the graph it has to be decided what nodes of the existing clustering could have been influenced the most. The key assumption is, that especially nodes close to the event might change and therefore all suggested strategies incorporate this proximity.

As used within the example shown in Figure 6.1, one possible approach is to dissolve complete clusters. Of course this should include clusters of both nodes u, v. Depending on the allowed size of the set of unclustered nodes V_u, other clusters around those could be dissolved as well. As V_u heavily influences the number of decision variables for the ILP and therefore the time needed to find and optimal solution, its scaling is vital for the performance of the algorithm. Cluster sizes vary with the overall size of the graph and its components. Therefore even dissolving a single cluster could free more nodes than feasible.

Another disadvantage when freeing complete clusters is the difference of proximity of single nodes within and between clusters. The hop distance from u to a node p within u's cluster can be significantly higher than the hop distance to a node q in another cluster. Figure 6.2 shows a graph being optimally clustered in (a). The newly inserted edge $\{u, v\}$ in (b) would not lead to a change within the clustering, if all nodes from u's and v's clusters were freed. Graph (c) shows what nodes would be extracted from their respective clusters in case of freeing a 2-hop neighborhood around the affected nodes. This time, also nodes within the blue cluster are considered and thus the clustering can be changed accordingly.

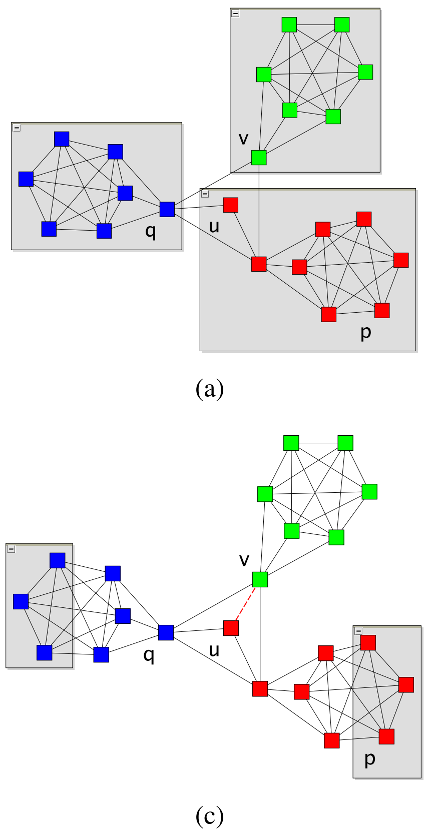

Figure 6.2: Update of a graph's clustering due to the insertion of the red dashed edge. The coloring marks the initially optimal clustering while the grouping depicts the current clustering. A 2-hop neighborhood around the affected nodes u, v gets extracted from their clusterings before the partial ILP is applied. The new clustering is optimal. If the affected clusters were freed instead, the clustering would stay unchanged.

Freeing neighbors allows to scale the number of extracted nodes. Considering the capabilities of the used linear program solver, the possible size of V_u can be preset. Then the freed neighborhood can be enlarged incrementally by increasing the hop-distance from u and v until the maximal size is reached. For a growing number of hops h, it can be assumed that the size of each h-hop neighborhood increases exponentially in h. It is likely that the last hop will free more nodes than allowed and therefore amongst the nodes within the h_{max}-hop neighborhood a second criterion has to be applied. Two possible strategies are to either free nodes with a high degree first or to free nodes with a low degree first, and for each strategy points can be made. The influence of the change within the graph on a node k with a high degree is stronger than the influence on a node l with a low degree, as the scalars to X_{uk} and X_{ul} incorporate the multiplication of the nodes' degrees within the expected coverage term. On the other hand nodes with a low degree tend to be freed less often as they will not be in as many neighborhoods. Therefore it might be more valuable to optimize their position within a clustering if possible. Both strategies will be evaluated.

For the neighborhood freeing strategy that considers nodes of high degrees first, the following disadvantageous effect can be observed. Figure 6.3 shows a 25-node clique with antennas. Those antenna-nodes will not be extracted unless the neighborhood size is bigger than the clique size. In the figure, the coloring marks the optimal clustering with regard to modularity. If two of the antenna nodes had been clustered together at a timestep without such density within the ball, they would never have been separated again. Besides using another freeing approach (e.g. freeing nodes with low degree first), the partial ILP algorithm can come up against this phenomenon by explicitly investigating clusters with small size and connections to other clusters. However, for reasons of compatibility of the different strategies and comparability of the experiment's results for different strategies, this is not done so far.

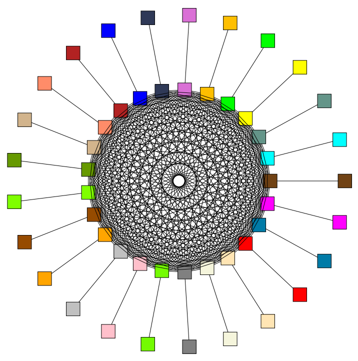

Figure 6.3: The graph shows a clique with 25 nodes and an additional antenna attached to all of those. The coloring marks the optimal clustering.

6.5 The Experiments' Environment

6.5.1 Comparing to Greedy Modularity

While conducting experiments to test the partial ILP algorithm, the two criteria of interest are the modularity value of each timesteps' clustering and the distance between the found clustering and its predecessor in time. While the optimal distance to the previous clustering is known to be zero, the optimal modularity value of a possible clustering is unknown. It is necessary to use another algorithm to compute it. Unfortunately, all techniques known to calculate optimal clusterings lack the performance necessary to be applied after each modification of a dynamic graph. Newman's greedy modularity algorithm is known to perform quite well in terms of index quality and even excellent in running time compared to other algorithms and especially mathematical program solvers. For this reason it seems adequate to use it as an indication for the index performance of the partial ILP approach. Within each timestep of all experiments the greedy algorithm is applied to the current snapshot of the dynamic graph. As it does not consider any previous clustering information but calculates a clustering globally and focuses on a high modularity value, it is not expected to result in low distance measure values. This bears the opportunity to observe at what cost, in terms of non-local changes in the clustering's structure, the index quality can further be optimized.

6.5.2 Incrementally Constructing Static Graphs

In order to use the partial ILP with static cluster graphs, their construction is stretched over time. Therefore, a graph $G = (V, E)$ with $|E| = m$ gets decomposed and built up edge by edge. Starting with an empty set of edges $E_0 = \emptyset$ and a pool $E_p = E$ of edges to add at time t_0, in every timestep t_i a random edge e gets extracted from E_p and is added to $E_{i+1} = E_i \cup \{e\}$ until $E_m = E$ at time t_m. Inter- and intra-cluster edges are added with the same rate they exist in a respective clustering, thus reflecting its significance in each timestep. In case of a weighted graph, each edge e with $w(e) > 1$ will be added within $w(e)$ steps that do not need to be consecutive in time. The process lacks true dynamic properties as no edge will ever be deleted or lowered in weight.

To be able to repeat the experiments and compare the results of different algorithms applied on such incremental graphs, the incremental assembly has to be repeatable as well. Hence the random number generator of the implementation is set up with a static seed.

6.5.3 Schematic Outline of the Experiments

As a first step of any experiment, the underlying dynamic graph $\mathcal{G}$ with the length n has to be instantiated. For its first timestep's graph G_0 the optimal modularity clustering C_0 is calculated. Note that the the number of nodes in G_0 is small, thus the ILP introduced in Chapter 4 can be applied. C_0 will be used as an input to the first partial ILP and to compare its results against an initial clustering.

Then $\mathcal{G}$ is advanced step by step and the following procedure is executed n times. For each $G_i \in \mathcal{G}$ the partial ILP algorithm is executed. First, nodes are extracted from $\mathcal{C}_{i-1}$ and marked unclustered according to the modification made and the chosen freeing strategy. Then $\mathcal{C}_i$ is calculated by solving the partial ILP. In parallel another clustering $\mathcal{C}'_i$ is constructed by Newman's greedy algorithm (see Section 6.5.1). Both clusterings are then compared to their predecessors $\mathcal{C}_{i-1}$ and $\mathcal{C}'_{i-1}$ by applying all distance measures introduced in Section 2.4.

This method allows to observe the clustering quality (with respect to clustering structure, number of nodes and number of edges) of the incremental partial ILP algorithm and to compare it against the globally optimizing greedy algorithm at all times. Also, the overall stability of the dynamic clustering is shown by how often and how much the structure changes.

Nodes	Edges	Clusters	p_I	p_O	Generated	Partial ILP	Newman	$\frac{\text{Partial ILP}}{\text{Newman}}$
182	1481	5	0.35	0.02	0.5937	0.5937	0.5937	1.0000
230	1451	5	0.20	0.02	0.5010	0.5016	0.4661	1.0762
214	1191	10	0.35	0.02	0.5427	0.5448	0.5174	1.0523
217	1641	10	0.50	0.02	0.6203	0.6203	0.5862	1.0582
222	2314	10	0.50	0.05	0.4222	0.4182	0.3457	1.2097
179	532	20	0.50	0.01	0.6412	0.6533	0.6343	1.0331
196	805	20	0.50	0.02	0.5025	0.5120	0.4547	1.1260

Table 6.1: The clustering results of the Partial ILP algorithm and Newman's greedy algorithm for similar sized, generated cluster graphs (around 200 nodes) with different cluster counts and probability values.

6.6 Experiments on Generated Graphs

6.6.1 General Setup

To get an idea of the behavior of the partial ILP approach, it is tested with static generated cluster graphs. These are incrementally constructed as described in Section 6.5.2, in order to simulate dynamic behavior. The partial ILP approach is dependent on the number of nodes (per cluster and component) and clusters (within components), if the hop-neighborhood strategy is selected to choose the nodes that get extracted from their clusters within a single step. Several instances of a graph holding around 200 nodes (subject to the standard deviation of cluster sizes) are generated with the Gaussian method described in Section 2.5.1. Their parameters are varied, they have different cluster counts and hence different probability values for inter- and intra-cluster connections, in order to maintain a certain level of significance.

6.6.2 Analysis

Table 6.1 shows some properties of the final clusterings. The index values achieved by the partial ILP are generally better than those of the greedy algorithm. Especially for less significant networks (i.e. low intra-cluster edge probability p_I with respect to the inter-cluster probability p_O and the number of intra-cluster and inter-cluster node pairs, compare Section 2.5.1) a significant difference in clustering quality is seen. This is despite the fact, that the Newman algorithm optimizes the clustering for the final snapshot of the artificially dynamic graph only, while the partial ILP stabilizes the restructuring between single timesteps and thus dampens the possible fluctuation (and gain) in modularity as well. While the dynamic clustering by the Newman algorithm is not expected to care for smooth transitions between consecutive timesteps' clusterings other than due to significance in the best found partitioning itself, it is worthwhile to evaluate the influence of the stability maintained by the partial ILP.

Figure 6.4 shows the development of the modularity index values of found clusterings by both algorithms during the incremental construction of the graph with 10 clusters, p_I set to 0.50 and p_O set to 0.01. The gap between the two curves depicting the index values grows with the insertion of nodes into the graph. While the Newman values are visibly unstable, the partial ILP's curve runs flatly, pointing to more similarity amongst its clusterings. This assumption is underlined by the graph given in Figure 6.5 showing the Jaccard distance (see Section 2.4) between each timestep's current and previous clustering. Only for the partial ILP's clusterings values equal to zero have been pruned. With a growing link structure between the nodes of the graph, it becomes more and more stable. It is easily visible that it outperforms the Newman algorithm's results, that always change the clustering structure between consecutive timesteps.

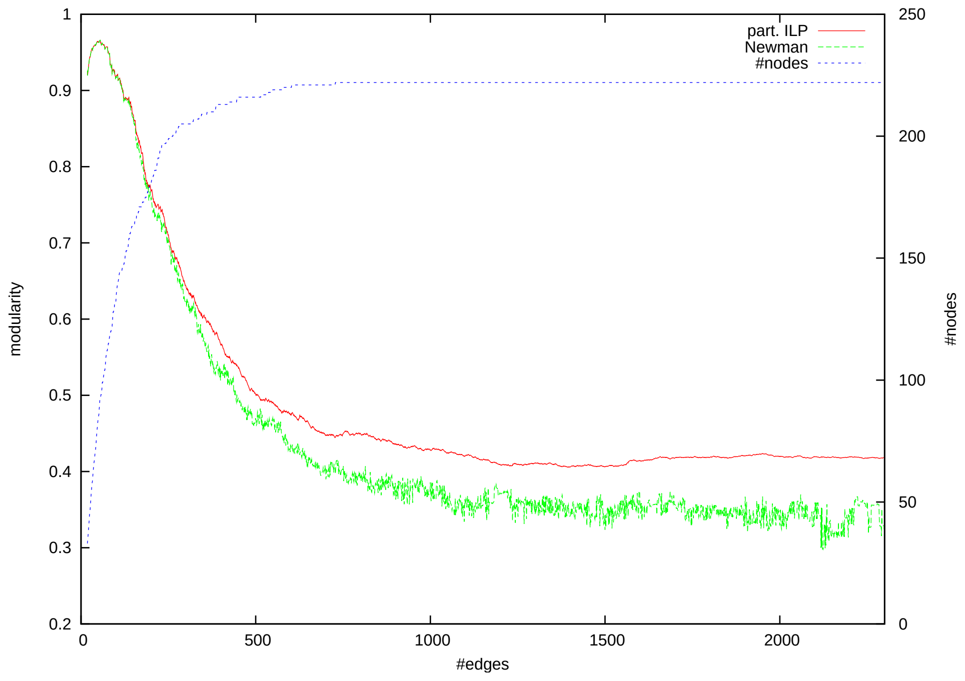

Figure 6.4: The graph shows the modularity value of each timestep's clustering (and hence its development over time) for the incremental clustering of a static, generated cluster graph. The partial ILP (red) incorporates previous clusterings, while Newman's greedy algorithm (green) works on the current graph only. Additionally, the number of nodes within each timestep's graph is shown (blue).

The combination of a better index value and less distance between consecutive timesteps' clusterings (and thus smoother evolution of the dynamic clustering) clearly expresses the applicability of the partial ILP algorithm to compute solutions to the Dynamic Graph Clustering Problem. In addition, the question arises, whether the approach could be used to cluster static graphs as well, as the final index values often exceed those found by Newman's algorithm. Here it has to be considered that the greedy algorithm does not need to compute all timesteps but the last while the partial ILP has to go over all edges incrementally. Still, the time needed to solve the small linear programs is shorter than solving an (I)LP for the complete graph and hence it comes down to a tradeoff between running time and quality.

In order to evaluate the running time needed to compute single timesteps' clusterings for larger graphs, several generated cluster graphs with up to 500 nodes have been processed by the partial ILP algorithm. Figure 6.6 shows the running time of each clustering step of an incrementally constructed graph with 500 nodes, $p_I = 0.25$, $p_O = 0.005$, and a generated

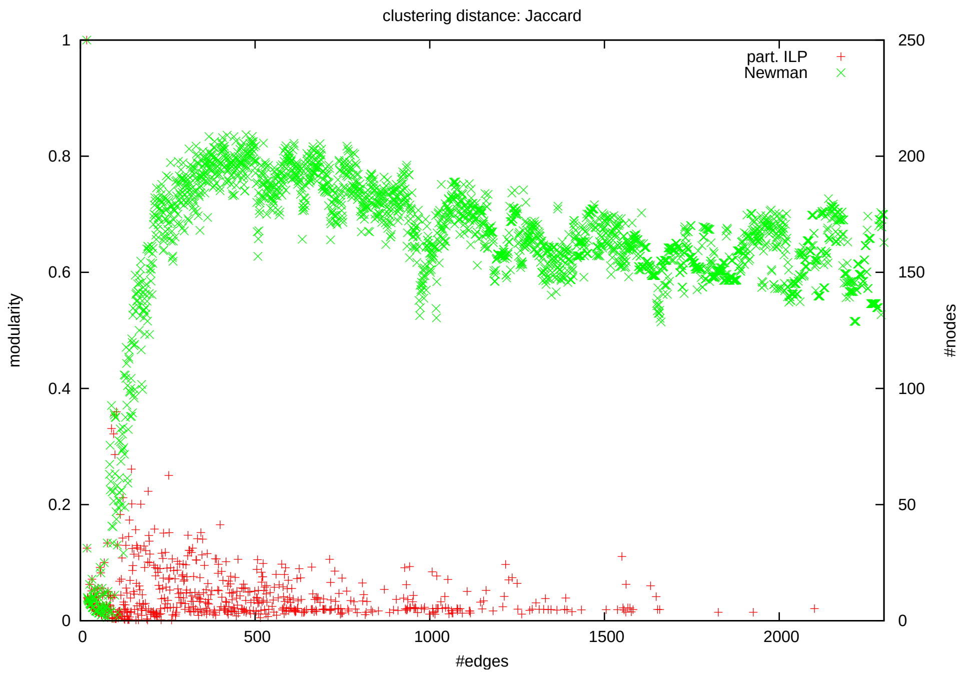

Figure 6.5: The graph shows the Jaccard distance between current and previous timesteps' clusterings for the incremental clustering of a static, generated cluster graph. For the partial ILP's clusterings (red), the majority of the values are equal to zero. They have been pruned for readability reasons.

clustering with 25 clusters and a modularity value of 0.6338. It can easily be observed, that most of the snapshots are optimized within around 2 seconds, despite the number of nodes in the graph. Several maxima with no more than 223 seconds running time occur due to the large number of clusters possibly leading to 50 decision variables as the neighborhood size was set to 25 nodes. The bound on the number of clusters mentioned in Section 6.3 is critical for graphs generated with the Gaussian generator. Though p_O can be lowered in order to have a significant ratio of intra- and inter-cluster edges, the created inter-cluster edges will link randomly between clusters. Their overall high amount therefore leads to all clusters being considered again. For real dynamic graphs, one would expect single clusters preferably linking to a subgroup of all other clusters.

For the sake of completeness, the modularity curve for the large generated dynamic clustering is shown in Figure 6.7. The partial ILP algorithm's results exceed those of the greedy algorithm. The final quality of 0.6378 is also higher than that of the generated clustering. The smoothness of the partial ILP's curve in comparison to Newman's algorithm's curve points to less fluctuation within the clustering structure.

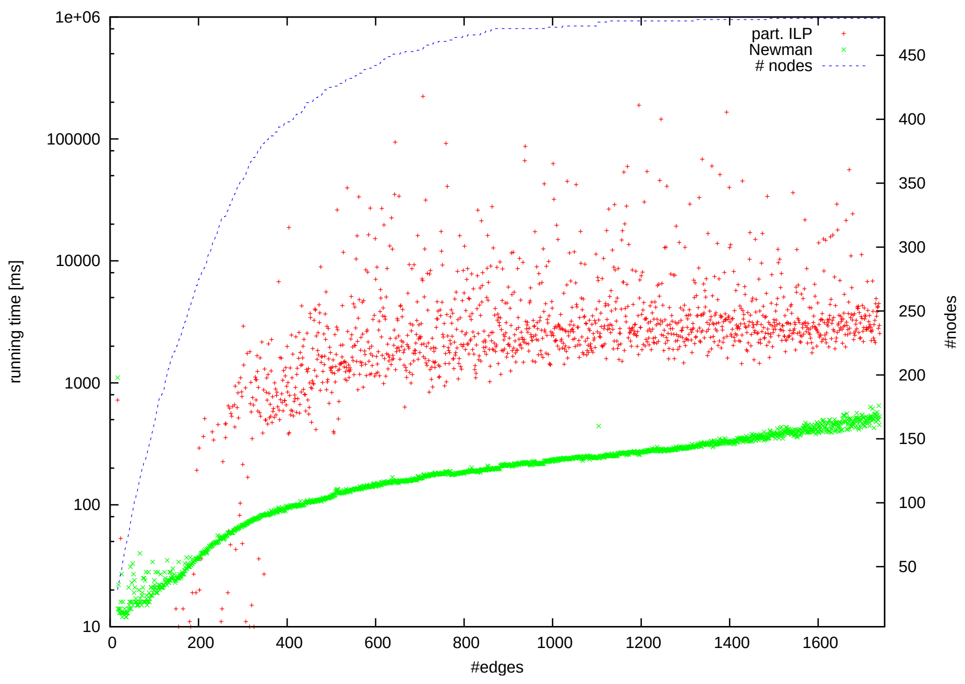

Figure 6.6: The graph shows the running time of the clustering algorithm for a large generated graph (500 nodes).

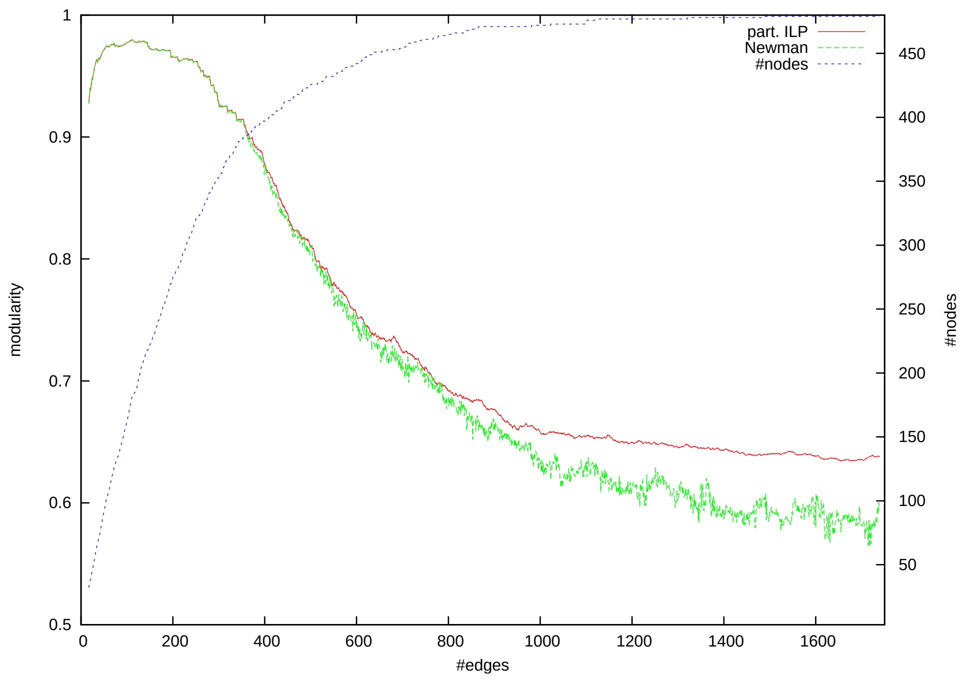

Figure 6.7: The graph shows the modularity of the dynamic clustering found by the partial ILP and Newman's greedy algorithm for a large generated graph (500 nodes).

6.7 Clustering Known Graphs

6.7.1 Graphs and General Results

To get an idea of the performance of the partial ILP with respect to the clustering quality, the algorithm has been tested on several graphs that are often used test cases taken from respective publications by Newman et al. For those graphs, index values of optimal clusterings are known. As described in Section 6.5.2, they are incrementally built up in order to be clustered by the partial ILP algorithm.

Graph	Nodes	Edges	Optimal	Partial ILP	Newman	$\frac{\text{Partial ILP}}{\text{Newman}}$
Chesapeake Lake	37	155	0.2766	0.2766	0.2714	1.0192
Dolphins	64	159	0.5285	0.5277	0.4955	1.0650
Football	128	723	0.6046	0.5077	0.4474	1.1348
Polbooks	105	441	0.5272	0.5253	0.5019	1.0466
Zachary's Karate Club	38	77	0.4308	0.4308	0.3974	1.0840

Table 6.2: Graphs with known optimal modularity clusterings and index values are clustered with the incremental partial ILP approach on the one hand and the greedy algorithm by Newman on the other hand. The 0.6046 value for the football graph is taken from [18] and has been found by rounding an LP.

While one would expect Newman's greedy algorithm to outperform the incremental approach, Table 6.2 shows, that the partial ILP's clusterings achieve good modularity values. For all graphs it is better than the greedy algorithm. In fact, it finds a nearly optimal clustering in all cases but the football graph, which is also the one for which it exceeds over the greedy algorithm the most.

6.7.2 Analyzing the Football Graph

The football graph has first been used by Girvan and Newman in [14]. It represents the schedule of games between American college football teams in one single season and has a known community structure as it divides the teams into conferences of around 8–12 teams each. Usually there are more matches played within conferences than between teams of different conferences, thus the data is supposed to contain a significant clustering structure. While a comparison of the detected clusterings to the known partitioning is useful to judge on how well an algorithm reflects the real life scenario, in this experiment it is of interest to analyze the index quality of any clustering found over the course of the incremental construction of the graph.

Figure 6.8 shows the development of the modularity value of each timestep's current clustering. The modularity graph for the dynamic clustering generated by the partial ILP algorithm (red) is constantly better than the graph depicting the modularity of the dynamic clustering generated by the greedy algorithm (green). Both lose significance over the course of time, probably due to the increasing interconnection of different components and the general loss of significance within bigger graphs. The latter assumption is backed by the observation that the decrease in modularity slows down dramatically after all nodes have been inserted into the graph (blue). Figure 6.9 shows the relative delta in modularity of the two approaches. It can easily be seen that the partial ILP algorithm performs better at all times, mostly around 8 percent and up to 18.6 percent at step 509.

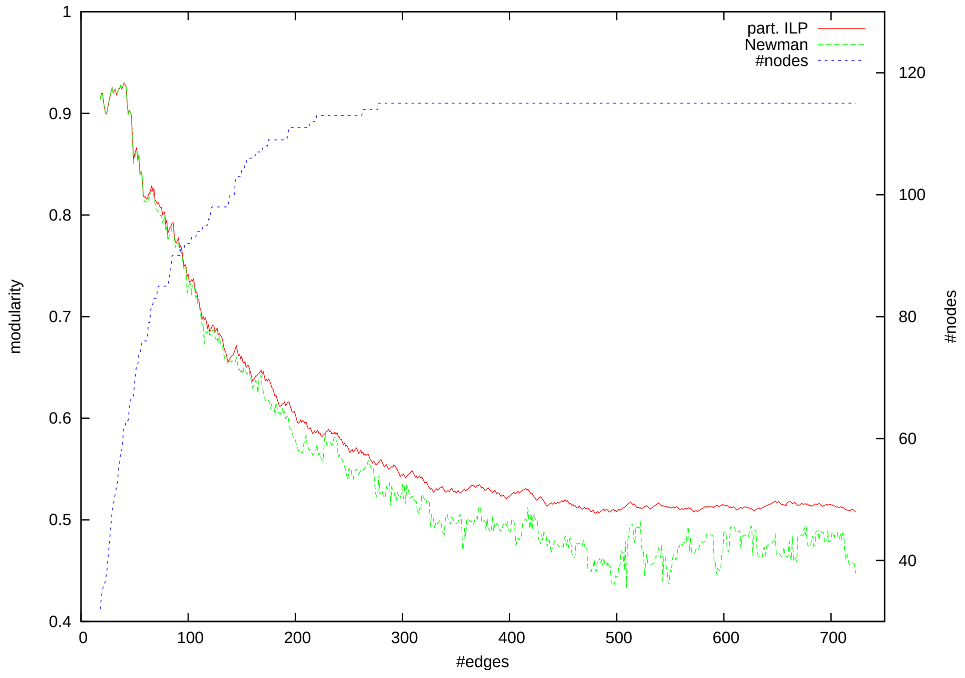

Figure 6.8: The graph shows the modularity value of each timestep's clustering, and hence its development over time. The partial ILP (red) incorporates previous clusterings, while Newman's greedy algorithm (green) considers the current graph only. Additionally, the number of nodes within each timestep's graph is shown (blue).

The unsteady curve for the modularity of the greedy clustering hints at the assumption that the structure of the partitioning is subject to constant change, as a single modification without altering the current clustering could only lead to small changes in the index value. Figure 6.10 shows the Jaccard distance between the current and the previous clustering of both algorithms and for all timesteps. While the clustering of the partial ILP is unchanged most of the time and never reaches a value larger than 0.41, the Newman clusterings' instability can easily be seen. As some changes have to take place, a better comparison is done by evaluating the delta of the distance measure values. Therefore, Figure 6.11 shows the absolute difference between the distance of the current and the previous Newman clustering and the distance between the current and the previous partial ILP's clustering

$$jacc(\mathcal{C}_{t-1}^{Newman}, \mathcal{C}_{t}^{Newman}) - jacc(\mathcal{C}_{t-1}^{ILP}, \mathcal{C}_{t}^{ILP})$$

for all timesteps of the dynamic clusterings. Only once there is a bigger change within the partial ILP clustering than in the Newman clustering.

The running time of each clustering step is shown in Figure 6.12. The scale is logarithmic and there are dimensions between the time needed to calculate the fast greedy algorithm and to solve the partial ILP. Nevertheless, the running time of the greedy approach grows exponentially in the number of edges in the underlying graph (in contrast to the worst case bounds given in Section 2.5.3), while the running time of the partial ILP stays more or less constant at around 3 seconds as expected.

For the sake of completeness, more information about the analysis of the football network is given in Appendix C.

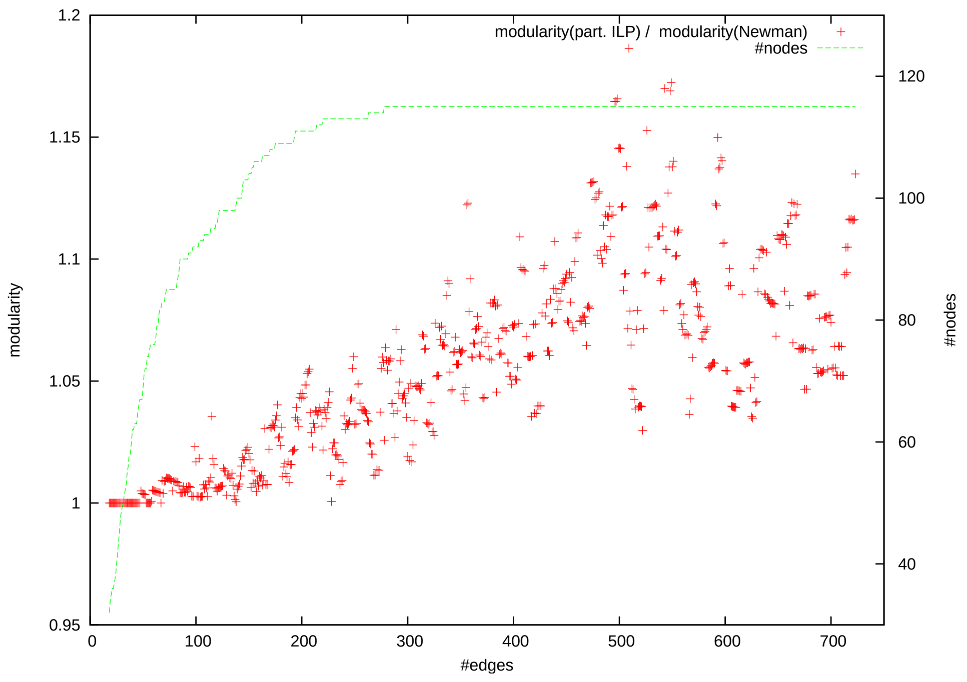

Figure 6.9: The graph shows the relative delta of modularity between the partial ILP's clustering and Newmans's greedy clustering by dividing one by the other.

6.8 Experiments on the Email Graph

6.8.1 General Setup

The email graph is a real world example of a dynamic graph. It is generated from the log files of the email server of the Faculty of Informatics (Fakultät für Informatik) of the Universität Karlsruhe. Whenever an email is sent between two or more members of the faculty (nodes of the graph), a connection between the sender and all receivers is established or weighted (edges of the graph). As timestamps showing the time of transmission of each email are available, there is an ordering amongst all edges. The email graph can be understood to illustrate recent interaction between email accounts and hence the information induced by edges can be given a lifetime in order to reflect true dynamics. Edges can be lowered in weight or deleted again from

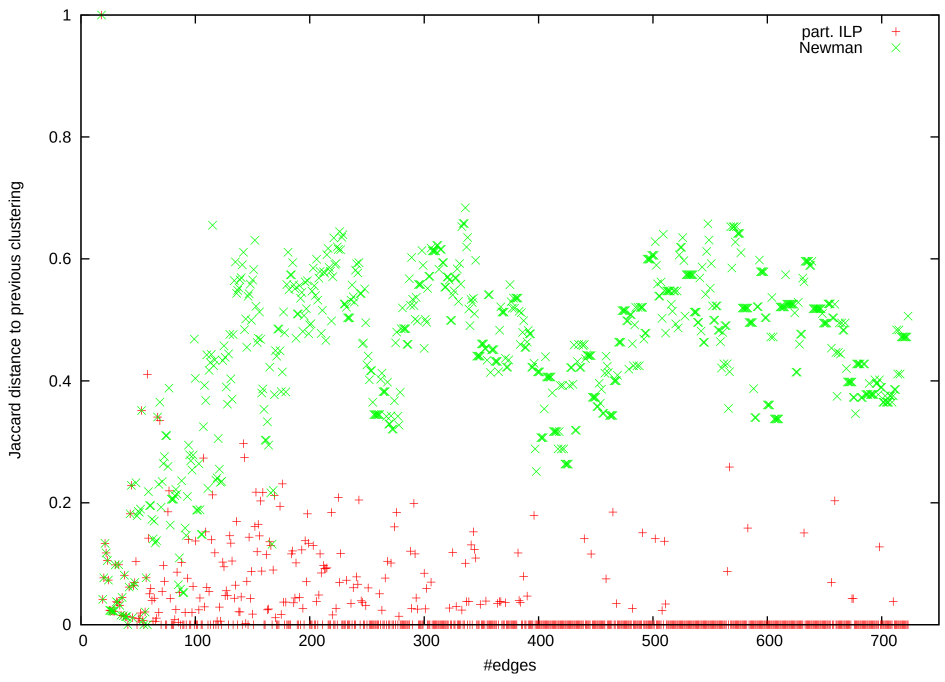

Figure 6.10: The graph shows the Jaccard distance between the current and the previous partial ILP's clustering and the current and previous Newman clustering.

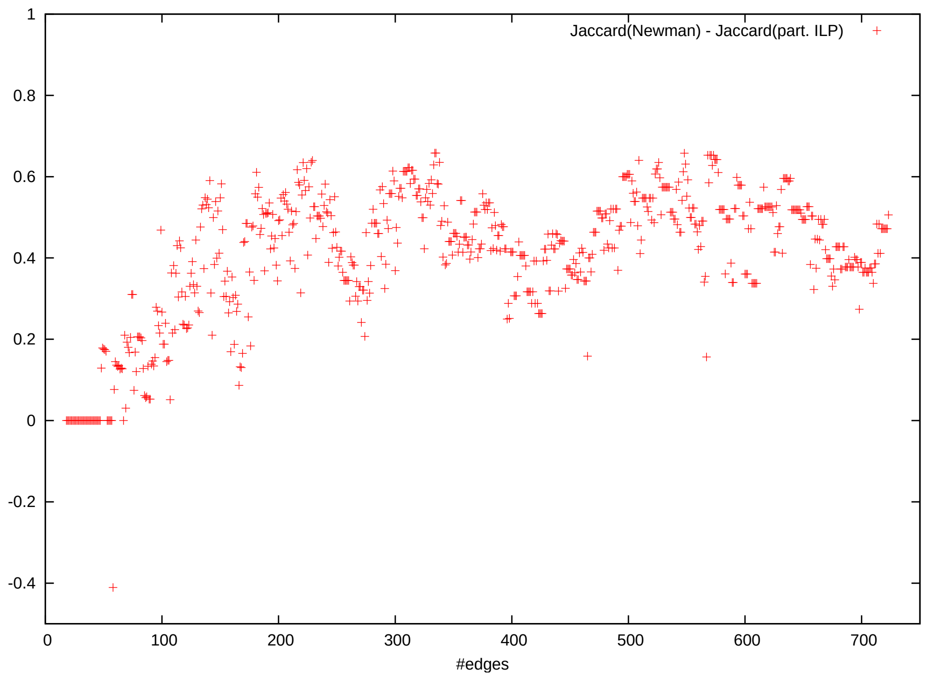

Figure 6.11: The graph shows the delta in the Jaccard distance between the current and the previous partial ILP's clustering and the current and previous Newman clustering.

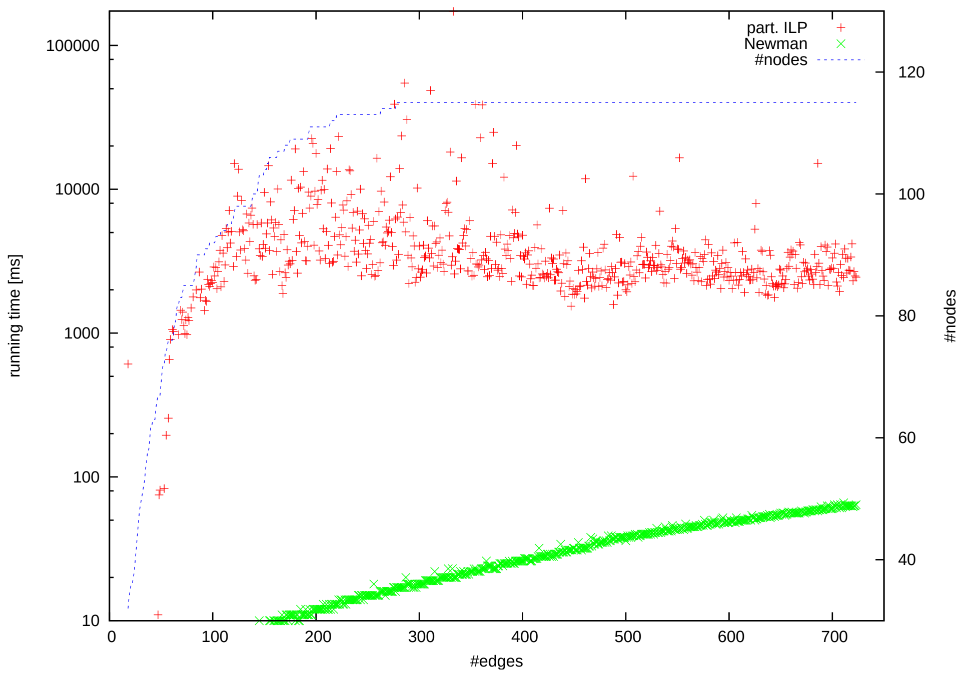

Figure 6.12: The graph shows the running time of each clustering step during the incremental clustering of the football network.

the graph after a defined timeout. The faculty is divided into several chairs, creating a natural partitioning of the graph legitimated by more intra-chair communication than emails between members of different professor's chairs. All personal information has been obfuscated.

To test the capabilities of the partial ILP algorithm to adapt to changing graph properties, the email graph has been constructed with different sizes (i.e. included chairs), for different dates and with different timeouts applied to connections. As the overall time performance of the partial ILP algorithm allows to process the complete graph, the main experiments have been conducted by starting the graph construction at the beginning of a month and maintaining a dynamic clustering showing its evolution over the following weeks. The timeouts for the graph's connections have been set to 12, 24, and 72 hours, in order to show the different changes within the graphs' (and thus the clusterings') structure. The following section will analyze the graph for a timeout of 24 hours, as it does not dissolve overnight though its number of nodes and edges varies significantly over time. For the sake of completeness, additional data considering connection lifetimes set to 12 hours and 72 hours is shown in Appendix D.1.

6.8.2 Analyzing the Email Graph

In this section, the results of the dynamic clustering of an instance of the email graph are presented in order to shed light on how the partial ILP's dynamic clustering behaves when working on real data. The example shows the communication within September 2007 with a connection lifetime set to 24 hours. The partial ILP is constructed for a neighborhood of 30 nodes around a modified edge, nodes with the lowest hop-distance and the lowest degree are extracted from their clusterings first. The weighted version of the objective function for the linear program is applied.

Figure 6.13 shows the modularity of the graph over the first three weeks of September. In general, the index achieves high values, most of the time exceeding 0.8. There are two noticeable collapses within the curve that correspond to the second and third weekend within the month. September 1st was a Saturday, thus the first weekend cannot be spotted as well. The graph does not reveal any major differences between the index values of the partial ILP clusterings and the Newman clusterings. Figure 6.14 shows the absolute difference in modularity of each timestep between both clusterings. It can easily be seen, that the biggest differences in modularity occur during the decomposition or new construction of the graph around the weekends. However, for the majority of the graphs the partial ILP's clustering is rated better or there is no difference in quality at all. The largest delta of 0.0149 is found at timestep 8882, this equals less than 2 percent. The similarity between the quality of the clusterings found is linked to the high overall modularity values, that point out a generally significant structure of the partitionings.

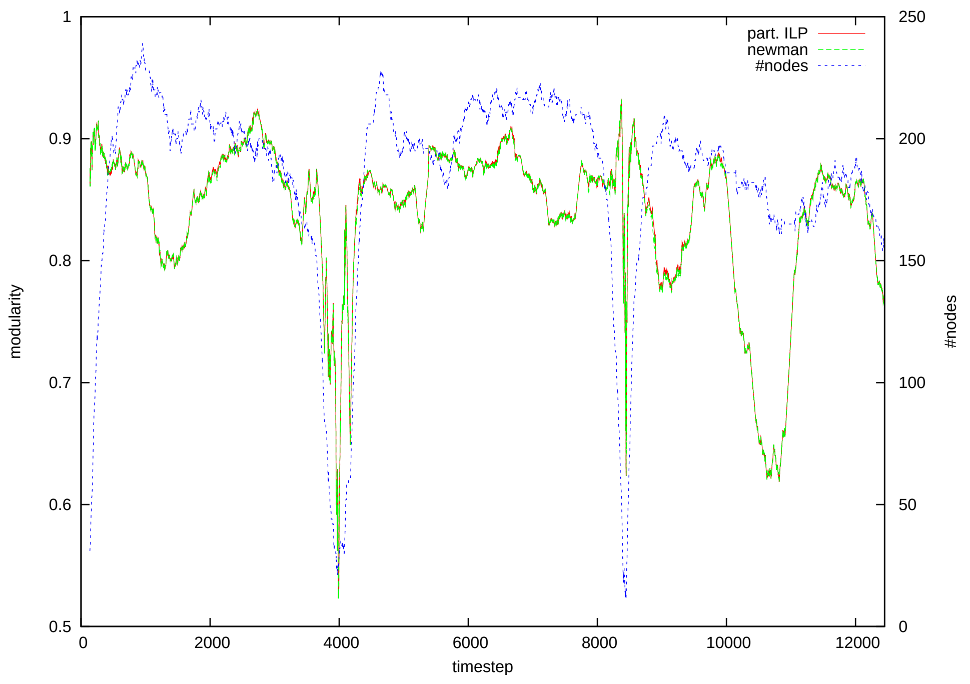

Figure 6.13: Modularity of the email graph for September 2007. The lifetime of connections is set to 24h. The collapses of the index value correspond to the decomposition of the graph due to connections timing out over the weekends.

Other fluctuations in modularity during the dynamic graph's lifetime have to be explained by different phenomena. As an example, note the minor collapse in the modularity graph during the third week (i.e. around timestep 10700). For some time the node count of the underlying graph drops, but not as extremely as over the weekends. The network depicted in Figure 6.15 shows the email graph at timestep 10820, where it has its minimum value besides the weekends. The snapshot holds 161 nodes connected by 173 edges. It is decomposed into two large components with 56 nodes (on the right side) and 34 nodes (on the left side) as well as 22 small components

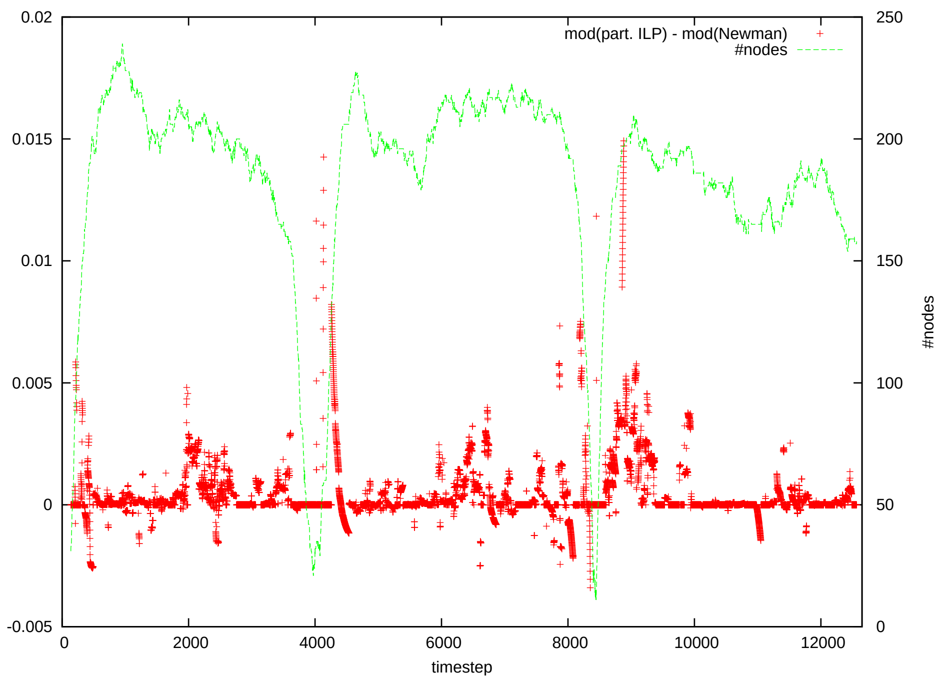

Figure 6.14: The graph shows the delta in modularity between the partial ILP and Newman's greedy algorithm for the email graph for September 2007. The lifetime of connections is set to 24 hours. The maximal difference is less than 0.015.

mostly holding 2 or 3 nodes. The identical clusterings found by the partial ILP and the Newman algorithm result in a modularity value of 0.6197. This is an optimal clustering for that particular instance and is also calculated by the ILP given in Chapter 4. The reason for the decline in email communication remains unknown, but might be due to the conference season or retreats.

Again, the individual clusterings' qualities achieved by both algorithms seem to be nearly identical, which points to their general significance and the fact, that the partial ILP algorithm performs at least as good as Newman's algorithm with respect to modularity. The stability of the partitionings has to be considered next. A combination of both gives insight into the dynamic clustering's overall quality with respect to the Dynamic Graph Clustering Problem. Figure 6.16 shows the delta in the Jaccard distance of both algorithms' clusterings, measured between each timestep's clustering and its predecessor. In addition, the modularity value of the partial ILP clustering is assigned to the right hand y axis. It can easily be seen, that the partial ILP approach outperforms the Newman approach throughout the lifetime of the graph in both, the number of timesteps with a better distance value and maximum delta. In comparison to the greedy algorithm the change within the partitioning of the graph is reduced to a minimum. Note that some distance is due to node creation and deletion and the capabilities of the distance measures to work with partial clusterings (see Section 2.4.3). There is no significant correlation between the modularity value of the clusterings found by the partial ILP algorithm and the delta in the Jaccard distance. Therefore, no general assumptions about when the approach performs best can be made at this point.

The time needed to update the dynamic clustering according to the modifications within the

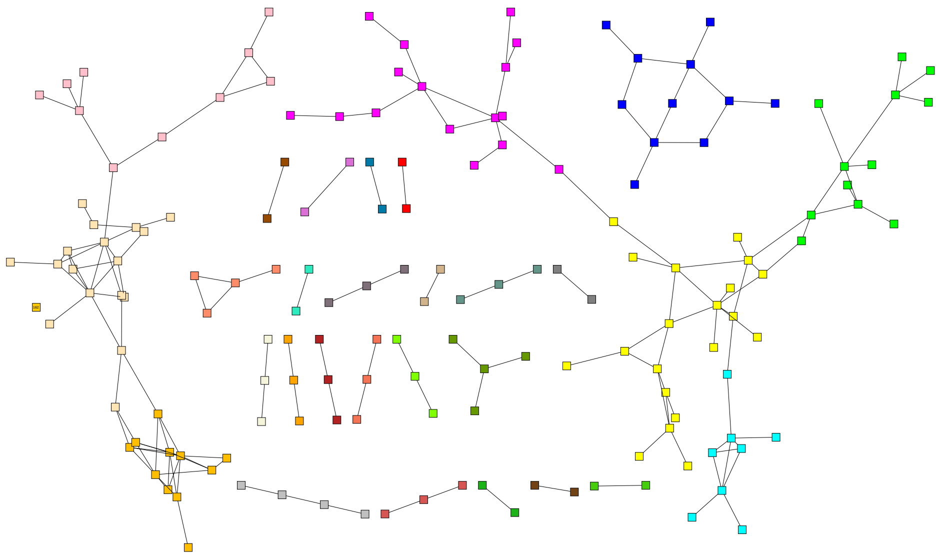

Figure 6.15: The email graph for September 2007 at timestep 10820. It consists of 161 nodes and 173 edges and is decomposed into two large (38 and 56 nodes) and many small components, due to sparse communication amongst existing nodes. The coloring of the nodes depicts an optimal clustering with modularity 0.6197. It is found by both, the partial ILP and Newman's greedy algorithm. The actual edge weights are not shown.

dynamic graph is shown in Figure 6.17. As expected, the calculation of the ILP needs around a 1000 times longer than the execution of the greedy algorithm. Still, on average the partial ILP is solved in around 3 seconds, while single (complicated) instances need around 7 to 10 seconds and many small instances (i.e. for small connected components) are processed even faster than by Newman's algorithm. The running time of the greedy algorithm is linked to the nodes, their number is not shown in the graph. Though there is no direct connection between edges and the construction of the linear program, the running time of the partial ILP is clearly linked to the number of edges within the email graph, also depicted in Figure 6.17. This can be explained by the fact that more edges lead to bigger components and hence to more clusters within one component. As long as the component size does not reach the maximum number of 30 nodes being extracted from clusters, the complexity of the partial ILP is exponential in the number of freed nodes. After that, each non-dissolvable cluster connected to those extracted nodes modified during one timestep, will lead to a constant increase in decision variables within the ILP (see Section 6.2) and therefore increased complexity. The large differences in average running time during the different phases of the graph (i.e. growing size, stable size, shrinking size) underline the assumption that the component size changes around the neighborhood size threshold, thus the expected constancy in running time cannot be seen. An analysis of the email graph with a timeout set to 72 hours and thus more nodes and edges in the graph is presented in

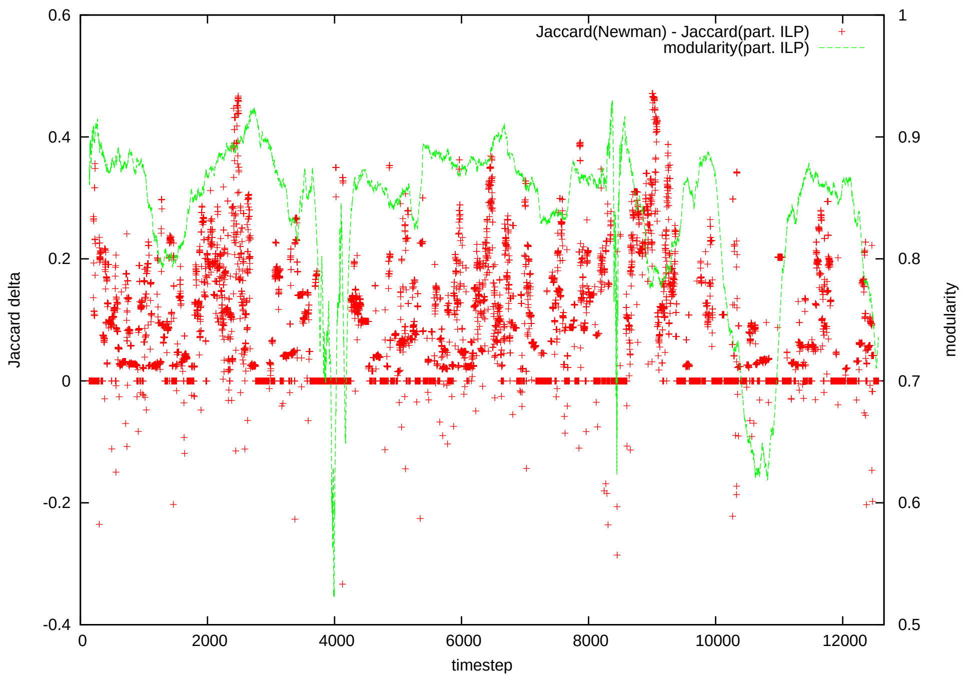

Figure 6.16: The graph shows the delta in the Jaccard distance between the current and the previous partial ILP's clusterings and the current and previous Newman clusterings.

Appendix D.1.2.

Considering the email graph, the partial ILP algorithm demonstrates its capabilities by achieving the same or better index values than Newman's greedy algorithm, while preserving the general structure of the dynamic clustering over the course of time. Though its running time is significantly higher, it is still applicable to practical problems as it allows the adaptation to the underlying email network within periods of seconds.

6.9 Applicability

The partial ILP approach allows to scale the number of decision variables independently from the size of the underlying graph and hence enables the processing of arbitrarily large graphs (with a bounded number of clusters each single cluster connects to). In practice the algorithm is capable of maintaining clusterings for the single timesteps' graphs of a dynamic graph that are at least as good as those found by Newman's greedy algorithm with respect to modularity. This result has been confirmed by incrementally clustering generated cluster graphs as well as known graphs. For the known graphs, a given optimal modularity value was nearly reached in most cases. The partial ILP algorithm can thus be applied as a trade off between trying to solve an optimal modularity ILP and finding a greedy solution fast.

The distance to previous clusterings has been evaluated by applying several types of well known distance measures. It has been shown that the partial ILP algorithm significantly outperforms the results achieved by solely optimizing the quality as done by the greedy approach. The

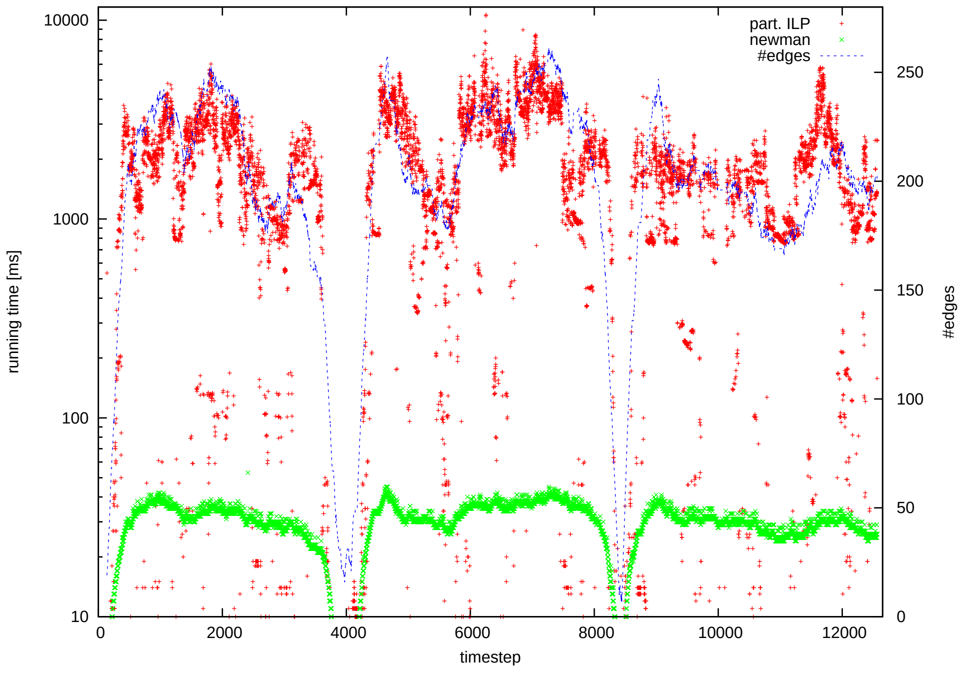

Figure 6.17: The graph shows the calculation time of each clustering step during the dynamic clustering of the football network.

case study described in Section 6.8 has underlined the applicability in case of true dynamics, i.e. the insertion and the deletion of nodes and edges as well as increasing or decreasing the weight of existing edges.

7 Conclusion

7.1 Summary of Results

It has been the goal of this thesis to explore and develop applicable techniques for solving the Dynamic Graph Clustering Problem defined in Chapter 3. In brief, it is dealing with the task to find optimal clusterings for all timesteps of a dynamic graph (i.e. a sequence of ordinary graphs) while minimizing the change in the clusterings' structure over time. It has been explained, that the main challenge is the combination of maximizing the quality of a clustering and minimizing the difference to a previous one, as both criteria can be conflicting. The developed algorithms are based on integer linear programs, that allow the exact definition of properties of their results.

First, in Chapter 4 it has been shown, how an ILP for index maximization can be optimized, in order to allow for the faster processing of larger graphs. The measures described do not reduce the theoretical complexity of the problem, but the size of graphs that can be handled has been increased significantly, as the original number of allowed nodes is quite small. Therefore, the applicability of the ILP to find optimal clusterings or as part of another algorithm has been extended substantially. Furthermore, the detailed analysis of the integer linear programs for the two indices modularity and significance performance has lead to the conclusion, that they rank clusterings of a graph identically. Though their absolute index values differ, it is sufficient to optimize either one of them.

In Chapter 5, a bi-criterial ILP explicitly optimizing a user-defined combination of a clustering quality index and distance measure (with regard to a previous clustering) has been constructed. The drawback of the algorithm is its dependence on finding solutions for an ILP whose number of decision variables (and thus its complexity) is linked to the number of nodes in the graph. At the moment, the approach is only applicable to graphs that do not have components with a size of more than 50 to 80 nodes (if standard hardware is used). A proof of concept employing the modularity index for quality and the Rand measure for distance to previous clusterings shows promising results, as it enables the emphasis of either one of the criteria. The anticipated balance between the index quality and the stability of a dynamic clustering has been shown, hence it allows the user to control what index quality at the cost of what fluctuation in the clustering structure shall be targeted.

A complementary ILP-based algorithm solving the Dynamic Graph Clustering Problem has been presented in Chapter 6. A partial ILP has been constructed, that locally re-optimizes the clustering with respect to a given index, after an atomary modification in the underlying graph has been made. It is based on the assumption that the employed modularity index has locality properties. While the clustering quality is optimized explicitly, minimizing the distance to previous clusterings is achieved implicitly by leaving the non-local parts of the partitioning unchanged. With respect to the index and the fixed parts of the clustering, the achieved result is optimal. The approach allows to scale the number of decision variables independently from the size of the underlying graph and hence enables the processing of arbitrarily large graphs (with a bounded number of clusters each single cluster connects to). In practice the algorithm is capable of maintaining clusterings for the single timesteps' graphs of a dynamic graph that are at least as good as those found by Newman's greedy algorithm. This result has been confirmed by incrementally clustering known graphs with a given optimal modularity value. The distance to previous clusterings has been evaluated by applying several types of well known distance measures. It has been shown that the partial ILP algorithm significantly outperforms the results achieved by solely optimizing the quality as done by the greedy approach. A case study conducted with a real-world example of a dynamic graph has underlined the applicability in case of true dynamics, i.e. the insertion and the deletion of nodes and edges as well as increasing or decreasing the weight of existing edges.

In total, several valuable approaches to solve the Dynamic Graph Clustering Problem have been developed.

7.2 Outlook

The methods presented in this thesis are only a first step towards finding optimal solutions to the Dynamic Graph Clustering Problem. There are remaining open questions considering their performance and further ideas and demands for future research arise.

The bi-criterial approach for solving the problem achieves excellent results but suffers from the demand to find solutions to a complex ILP. In order to be able to apply the concept to graphs with large sizes, it is desireable to find techniques to speed up this process. Rounding the modularity ILP has potential, but the question whether available techniques are able to sufficiently guarantee certain quality aspects is not answered yet. Additionally, the formulation of distance measures other than the pair counting variants might result in the need to revise the constraints model used to guarantee correct clusterings, as it does not allow for a straight forward implementation of those so far.

The potential of the partial ILP algorithm has not yet been fully analyzed. Further experimental evaluation is currently carried out and will include the clustering of a dynamic graph spanning more than a decade, being represented by more than 1 Million single timesteps. Possible further applications of the partial ILP include the combination with other information provided by a user or achieved by different mathematical criteria (e.g. centrality measures) and thus allow the extension to another type of hierarchical clusterings and fractional clusterings of very large datasets.

The biggest open question with regard to the partial ILP algorithm is the provability of locality properties of the used modularity index. The existence of those is the underlying assumption of the approach, and locally optimizing a clustering after a modification is promising in practice. Unfortunately, the theoretical analysis has been made difficult by the lack of practicable solutions to find optimal modularity clusterings. Thus it remains one of the biggest challenges to develop applicable algorithms to find clusterings with guaranteed quality with regard to modularity. During the course of the experiments, several available heuristics have been applied and it has become clear that they all suffer from obvious disadvantages. However, a fast algorithm to compute such clusterings would not only advance the processing of static graphs. It would also enable further studies of the locality of modularity and therefore possibly even the development of practical heuristics to update clusterings after modifications in the graph.

A Entropy Measures and Dynamic Clusterings

The behavior of entropy measures for distance between clusterings has not yet been studied with respect to dynamic clusterings. In Section 2.1.2 it has been defined, that two clusterings $\mathcal{C}(G), \mathcal{C}'(G')$ shall be compared based on a union $V_u = V \cup V'$ of their respective sets of nodes. Nodes within $V_u \backslash V'$ are considered to be unclustered within $\mathcal{C}$ and nodes within $V_u \backslash V$ are considered to be unclustered within $\mathcal{C}'$. A clustering that does not fully partition V_u will be called a *partial clustering*. The influence of a growing number of unclustered nodes within a clustering on entropy distance measures will be studied in this chapter.

A.1 Clustering Entropy for Partial Clusterings

The entropy of a clustering is defined as follows:

$$\mathcal{H}(\mathcal{C}) = -\sum_{C \in \mathcal{C}} \frac{|C|}{n} \log_2\left(\frac{|C|}{n}\right)$$

It is the sum over all clusters within the clustering. As each cluster C can be of different size $|C|$ which can be assumed to be of particular influence to the sum and relation to the total number of nodes n, the behavior of a single clusters' fraction $\mathcal{H}(\mathcal{C}, C)$ within $\mathcal{H}(\mathcal{C})$ will be analyzed in this section.

$$\begin{aligned}\mathcal{H}(\mathcal{C}, C) &= -\frac{|C|}{n} \log_2\left(\frac{|C|}{n}\right) \\ &= -\frac{|C|}{n} \log_2(|C|) + \frac{|C|}{n} \log_2(n)\end{aligned}$$

Its derivative with respect to n is supposed to give insight to its behavior in case n grows due to the introduction of unclustered nodes.

$$\begin{aligned}
\mathcal{H}(\mathcal{C}, C)' &= -\frac{|C|}{n}\log_2\left(\frac{|C|}{n}\right)\frac{\partial}{\partial n} \\
&= \frac{|C|}{n^2}\cdot\log_2(|C|) - \frac{|C|}{n^2}\log_2(n) + \frac{|C|}{n}\cdot\frac{1}{n\cdot\log_{10}(2)} \\
&= \frac{|C|}{n^2}\cdot\left(\log_2(|C|) - \log_2(n) + \frac{1}{\log_{10}(2)}\right)
\end{aligned}$$

It becomes clear that the function will have its only root within the possible range (which must be a maximum), if the second factor becomes 0.

$$\begin{aligned}
& & 0 &= \log_2(|C|) - \log_2(n) + \frac{1}{\log_{10}(2)} \\
&\Leftrightarrow & 0 &= \frac{1}{\log_{10}(2)}\cdot\left(\log_{10}(|C|) - \log_{10}(n) + 1\right) \\
&\Leftrightarrow & \log_{10}(n) &= \log_{10}(|C|) + 1 \\
&\Leftrightarrow & n &= e^{\log_{10}(|C|)+1} \\
&\Leftrightarrow & n &= |C|\cdot e
\end{aligned}$$

Any $\mathcal{H}(\mathcal{C}, C)$ for a cluster C with a size less than a fraction of $1/e$ of the total node count n shrinks if n grows further. As the number of clusters stays the same this means that at least for a clustering $\mathcal{C}$ that only consists of clusters of a smaller size, the total entropy will shrink as well. The interpretation that the information contained within the clustering with regard to the larger set of nodes is less than before seems unintuitive and is a consequence of the fact that the information what nodes are unclustered is not considered so far.

Two possible strategies for the integration of the unclustered nodes within the calculation of the clustering's entropy are counting them as if they were in one cluster or as if they were singleton clusters within $\mathcal{C}$, respectively. The first option would lead to a decrease of the clusterings entropy (i.e. $\mathcal{H}(\mathcal{C}) \overset{n\to\infty}{\longrightarrow} 0$) but would also imply a relation amongst unclustered nodes. The latter option on the other hand would increase the clusterings entropy towards its maximum value of $log_2(n)$ (i.e. $\mathcal{H}(\mathcal{C}) \overset{n\to\infty}{\longrightarrow} \log_2(n)$). This makes sense as the information to actually have a relation within a cluster is more uncertain for every single node.

A.2 Correlation Information for Partial Clusterings

The correlation information between two clusterings $\mathcal{C}(G), \mathcal{C}'(G')$ with $V = V'$ as defined in [23] is given as follows.

$$\mathcal{I}(\mathcal{C},\mathcal{C}') = \sum_{C\in\mathcal{C}}\sum_{C'\in\mathcal{C}'} \frac{|C\cap C'|}{n}\cdot\log_2\left(\frac{|C\cap C'|\cdot n}{|C|\cdot|C'|}\right)$$

$|C\cap C'|/n$ is the probability of a random node to be within the intersection of the two clusters C, C'. As for single clustering entropy it has to be decided how correlation information for two clusterings with different underlying node sets $V \neq V'$ can be calculated with regard to its meaning. Again, the behavior of a single summand for the two clusters $C \in \mathcal{C}, C' \in \mathcal{C}'$ and for a growing number of nodes $|V_u| = |V \cup V'|$ will be analyzed, it is given as follows.

$$\begin{aligned}\mathcal{I}(C,C') &= \frac{|C\cap C'|}{n}\cdot\log_2\left(\frac{|C\cap C'|\cdot n}{|C|\cdot|C'|}\right)\\ &= \frac{|C\cap C'|}{n}\cdot\log_2\left(\frac{|C\cap C'|}{|C|\cdot|C'|}\right) + \frac{|C\cap C'|}{n}\cdot\log_2(n)\end{aligned}$$

Its derivative is given as follows.

$$\begin{aligned}\mathcal{I}(C,C')' &= \frac{|C\cap C'|}{n}\cdot\log_2\left(\frac{|C\cap C'|}{|C|\cdot|C'|}\right) + \frac{|C\cap C'|}{n}\cdot\log_2(n)\frac{\partial}{\partial n}\\ &= -\frac{|C\cap C'|}{n^2}\cdot\log_2\left(\frac{|C\cap C'|}{|C|\cdot|C'|}\right) - \frac{|C\cap C'|}{n^2}\cdot\log_2(n) + \frac{|C\cap C'|}{n}\cdot\frac{1}{n\cdot log(2)}\\ &= \frac{|C\cap C'|}{n^2\cdot log(2)}\cdot\left(1-\log\left(\frac{|C\cap C'|}{|C|\cdot|C'|}\right) - \log_{10}(n)\right)\end{aligned}$$

The function will have its maximum, if the second factor is 0. It becomes clear, that for a fixed number of shared nodes this maximum will be reached for lower n by small clusters than by large clusters. This is due to the fact that $\log((|C\cap C'|)/(|C|\cdot|C'|)$ is always negative and smaller for large clusters.

$$\begin{aligned} && 0 =&\ 1-\log\left(\frac{|C\cap C'|}{|C|\cdot|C'|}\right) - \log_{10}(n)\\ \Leftrightarrow && \log_{10}(n) =&\ 1-\log\left(\frac{|C\cap C'|}{|C|\cdot|C'|}\right)\\ \Leftrightarrow && n =&\ e\cdot\left(e^{\log_{10}(|C\cap C'|/(|C|\cdot|C'|)}\right)^{-1}\\ \Leftrightarrow && \frac{n}{e} =&\ \frac{|C|\cdot|C'|}{|C\cap C'|}\end{aligned}$$

Small clusters do not only have the higher fractional values, they also surmount larger clusters in possible number. When comparing clusterings with themselves, it can be seen that the correlation information value of two clusterings with many clusters excels the value for two clusterings (of the same graph) with few clusters.

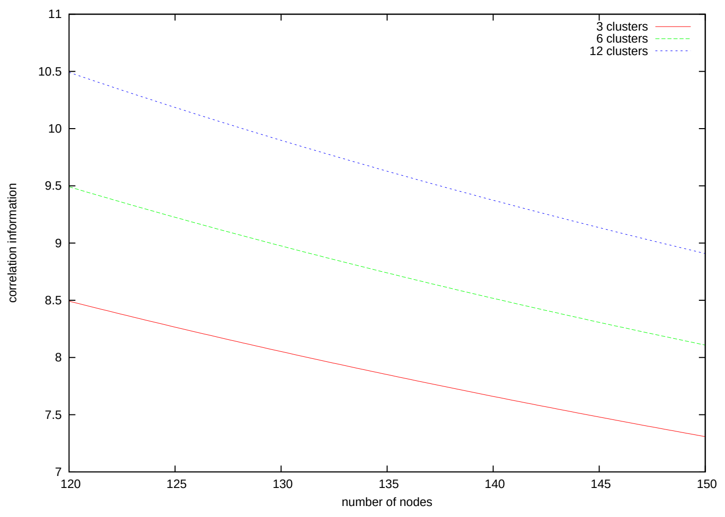

Figure A.1: Correlation information for identical clusterings with respect to different numbers of regular clusters and a growing number of unclustered nodes.

In Section A.1 it was reasoned that unclustered nodes within a clustering should be treated as singleton clusters as this conserves the semantics of the clustering entropy. This also holds for the correlation information. If two clusterings do not partition exactly the same node sets, all nodes that are part of one clustering will only be unclustered within the other. Treating them as singleton clusters slows down the decrease of the correlation information value while expressing the fact that the clusterings are less correlated. Also, applying the same manipulations in order to calculate clustering entropy and correlation information maintains their relation. Especially the fact that the value of the correlation information is bounded by the minimum of the two involved clusterings' entropy values is conserved.

B Implementation Details

B.1 Software

Most software components used and developed for this thesis are written in the Java programming language (version 1.5). The implementation of the presented algorithms makes heavy use of existing software, partly commercial, developed at the Institute for Theoretical Computer Science (Institut für Theoretische Informatik) of the Universität Karlsruhe, and open source. As a basis for all operations on graphs, the yFiles for Java by yWorks ([33]) have been used. They are extended by the clustering framework developed within the research group, offering a vast set of operations on clusterings as well as clustering visualization, index calculation and distance measures for clusterings.

This framework has been extended by dynamic graphs and incremental graphs as well as several interfaces to mysql databases, holding the graph material for the experiments. Dynamic graphs have been used for the email graph and the patent application network (that is work in progress). Incremental graphs have been used to cluster generated graphs and known graphs.

The linear program solver used to calculate the ILPs constructed for the bi-criterial approach described in Chapter 5 and the partial ILP approach described in Chapter 6 is taken from the open source project lp_solve ([28]), providing an implementation of the revised simplex method and the branch-and-bound method for integers. In order to conduct the experiments of this thesis, a variety of linear program generators as well as different objective functions have been implemented using the Java API of lp_solve. For performance reasons, all ILPs have been ported to minimization problems as well, as lp_solve seems to find optimal integer solutions faster for those. This includes the ILPs for optimal modularity (Section 4.1.4), optimal significance performance (Section 4.2.3), bi-criterial optimization of modularity and the Rand distance measure (Section 5.2.1), the partial ILP (Section 6.2) as well as the LP generators for complete graphs and connected components.

The results have been analyzed using the yWorks graph editor Yed in combination with custom plug-in modules for visualization. Besides, the results have been plotted with gnuplot (http://www.gnuplot.info/, version 4.2) and movies, showing the continuous evolution of dynamic networks, have been animated with custom Java software and encoded with the open source movie encoder MEncoder by The Mplayer Team (http://www.mplayerhq.hu/).

In total, more than 8500 lines of code (mostly Java) have been written.

B.2 Hardware

The clustering experiments as well as all tests of the developed algorithms have been computed on standard hardware only. If not mentioned otherwise, a PC equipped with an AMD Opteron processor running at 2.6 GHz has been utilized. The operating system of that machine has been running the 64 bit version of the linux kernel version 2.6.16.13-4-smp.

C Distance Measures for the Football Network

In Section 6.7.2, the football network also used by Girvan and Newman in [14] has been chosen as an example to demonstrate the capabilities of the partial ILP algorithm in finding high quality clusterings. In addition to the given descriptions, further information about the incremental clustering is available in the form of the evaluation of all clustering distance measures that are given in Section 2.4.2.

Figures C.1–C.4 show the relative delta in clustering distances measured between the current and the previous partial ILP clustering and the current and previous Newman clustering. After stretching the graphs for the Rand, Fowlkes-Mallows, and Maximum-Match distance measures shown in Figures C.1–C.3 as well as the Jaccard measure depicted in Figure 6.11, general characteristics of the curves as well as several distinct maxima are highly similar amongst them. This observation, that has been made throughout all experiments, leads to the conclusion that it is enough to present the results for one distance measure while all of them are tested. The Jaccard measure has been chosen, as it seemed to be the most sensitive.

The graph for the Fred and Jain entropy measure in Figure C.4 does not compare to the others. It clearly does underline the supremacy of the partial ILP's dynamic clustering over Newman's version in terms of clustering stability, though for many timesteps there is no difference within the impact of graph modifications on the clusterings' structures. As this could be due to no changes or equally sized changes in both, it is worthwile evaluating the absolute distance results. The respective graph is shown in Figure C.5, only values greater zero have been plotted. The partial ILP clusterings do not contain much new information with respect to their predecessors and hence the measure denotes no large distance values. The Newman clustering on the other hand is subject to constant change, though its magnitude is classified differently than by the other distance measures. Those timesteps reaching maximal values for the Fred and Jain distance measure yield high values within the other measures' graphs as well, but not the other way round. Therefore a combination of the different techniques might lead to further insights. Figure C.6 shows a simple implementation of such a combination. The two distance measures Fowlkes-Mallows and Fred and Jain are shown within one graph by adding their respective values for each timestep. They were chosen, because they seem to scale similarly. Again, the same characteristics of the curve are coming through, though being dampened by the seemingly more fuzzy Fred and Jain values.

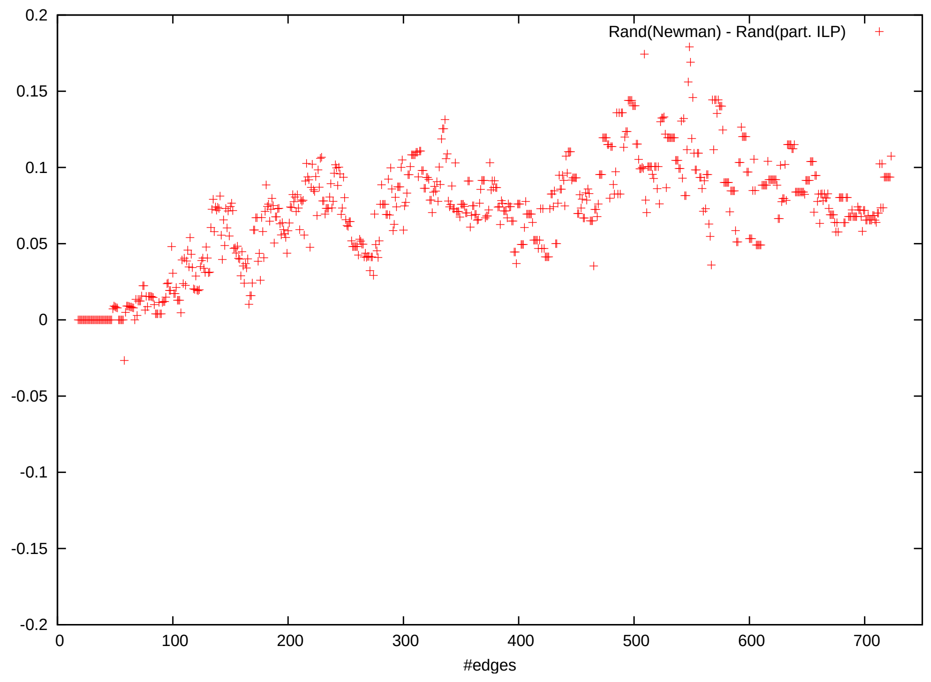

Figure C.1: The graph shows the delta in the Rand distance between the current and the previous partial ILP's clustering and the current and previous Newman clustering.

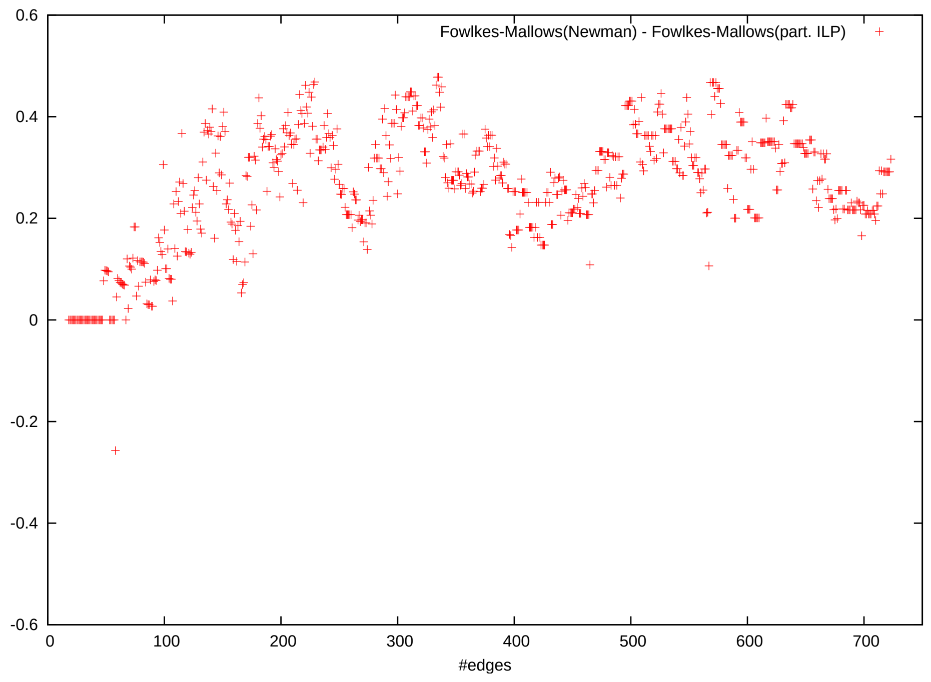

Figure C.2: The graph shows the delta in the Fowlkes-Mallows distance between the current and the previous partial ILP's clustering and the current and previous Newman clustering.

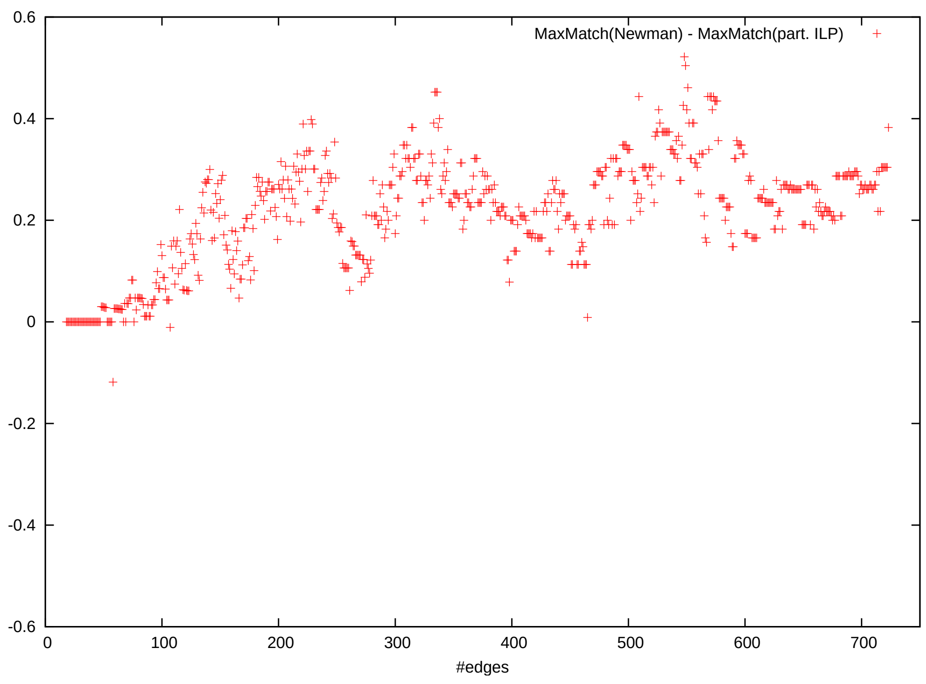

Figure C.3: The graph shows the delta in the Maximum-Match distance between the current and the previous partial ILP's clustering and the current and previous Newman clustering.

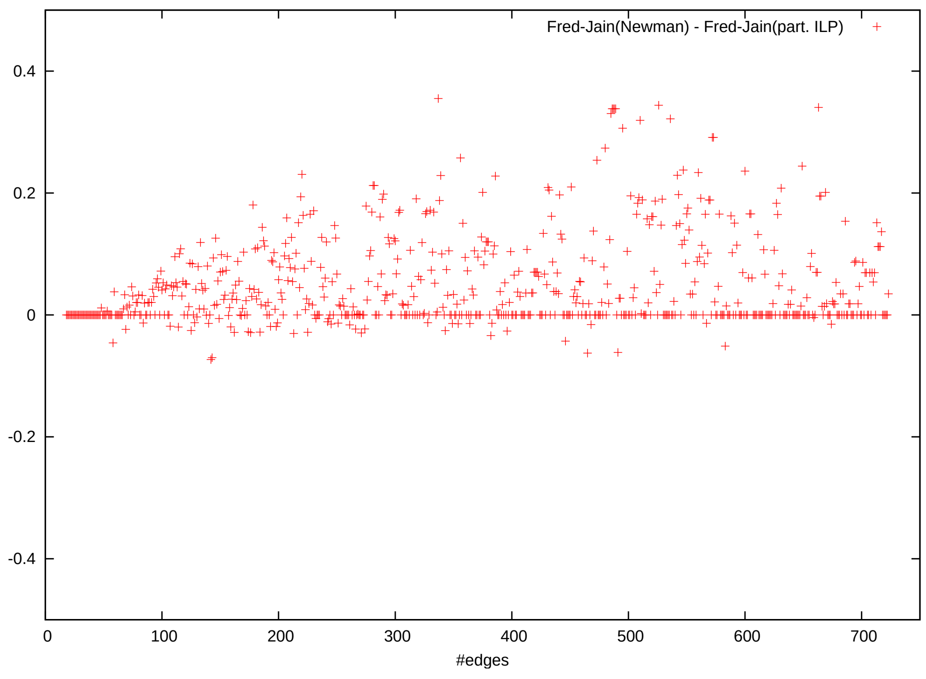

Figure C.4: The graph shows the delta in the Fred-Jain distance between the current and the previous partial ILP's clustering and the current and previous Newman clustering.

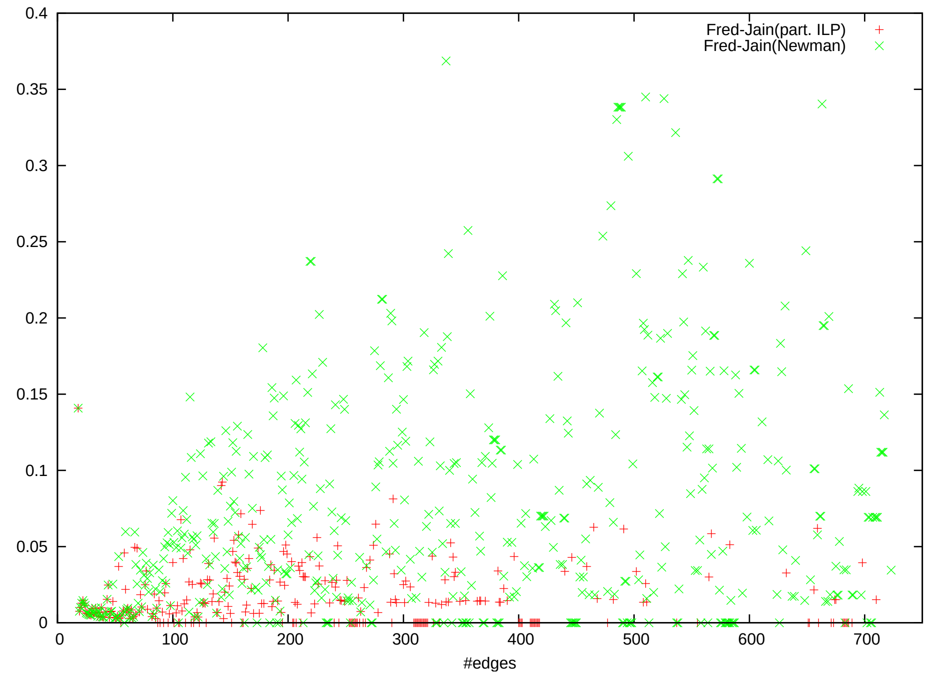

Figure C.5: The graph shows the Fred-Jain distance between the current and the previous partial ILP's clustering and the current and previous Newman clustering. Only values greater zero are shown.

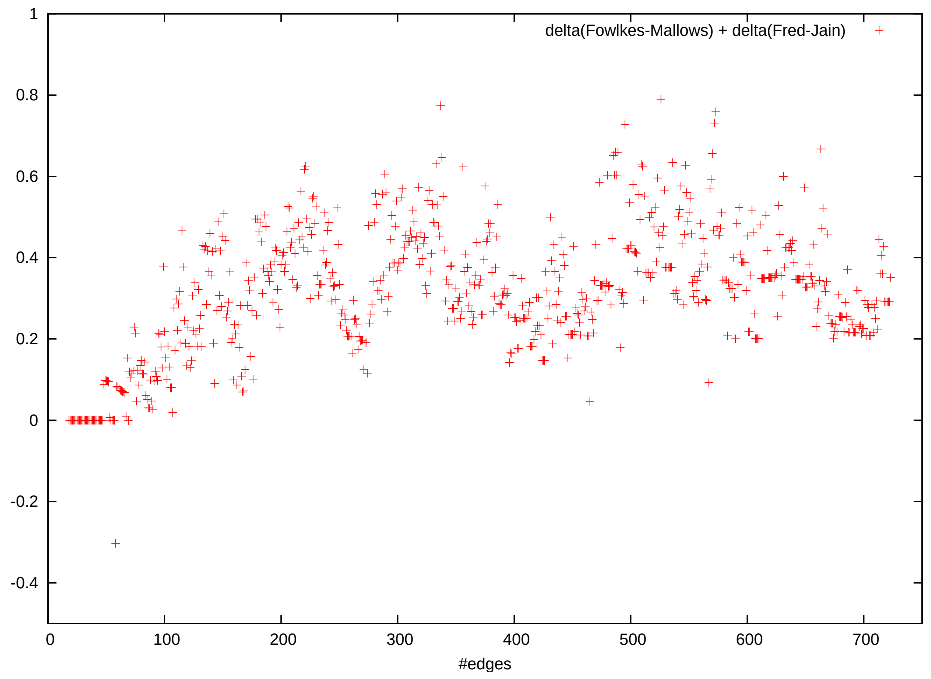

Figure C.6: The graph shows the delta in the Fowlkes-Mallows distance plus the delta in the Fred-Jain distance between the current and the previous partial ILP's clustering and the current and previous Newman clustering.

D The Email Graph

D.1 Further Results from Experiments with the Partial ILP

D.1.1 Influence of Connection Lifetime on the Email Graph

In Section 6.8.2 an instance of the email graph for September 2007 has been presented. The connections within the graph show the anonymized email communication between individuals at the Faculty of Informatics (Fakultät für Informatik) of the Universität Karlsruhe. Sending an email can be interpreted as establishing or intensifying a relationship between two individuals. The underlying assumption is, that such relationships are subject to a certain lifetime, that has been set to 24 hours for the given example. In this section the same instance is clustered with a shorter connection lifetime of 12 hours and a longer lifetime of 72 hours. This has the effect that the graph gets sparse in case of little communication quite fast or does not decompose during times with less communication (e.g. a weekend or holiday) at all, respectively.

Figure D.1 and Figure D.2 show the modularity for both instances (12 hours lifetime and 72 hours lifetime) as done in Figure 6.13 for the same graph and 24 hours connection lifetime. Within Figure D.1 the degeneration of the email graph is clearly visible within the node count as well as the index value, that drops during the night due to the decomposition into small sized components that last. Still, the second and the third weekend can easily be spotted around the timesteps 4000 and 8400. Note that the graph has around 1000 timesteps more, directly corresponding to the same number of connections timing out and being lowered in weight or deleted.

Due to a sparse link structure during times of decomposition resulting in relatively small components, the average modularity of the dynamic clustering will generally be lower, if the lifetime is set to a shorter period. This is underlined by the constantly high index values seen in Figure D.2 for a 72-hour timeout length. Though a decrease in the node count points out the weekends, the 72 hour timeout is just long enough to have most connections outlast the 2.5 days. In contrast to both graphs with the shorter timeouts, a slight increase in modularity occurs during this period. It is explained by the fact that now small components disappear and the inter-cluster connections are pruned while especially larger and heavier clusters still exist. The same collapse of the modularity function within the third week as seen for the 24-hour instance can be spotted for both new graphs as well. Figure D.3 reveals that the phenomenon happens despite a growing number of nodes, leading to the conclusion that the number of components is rising again.

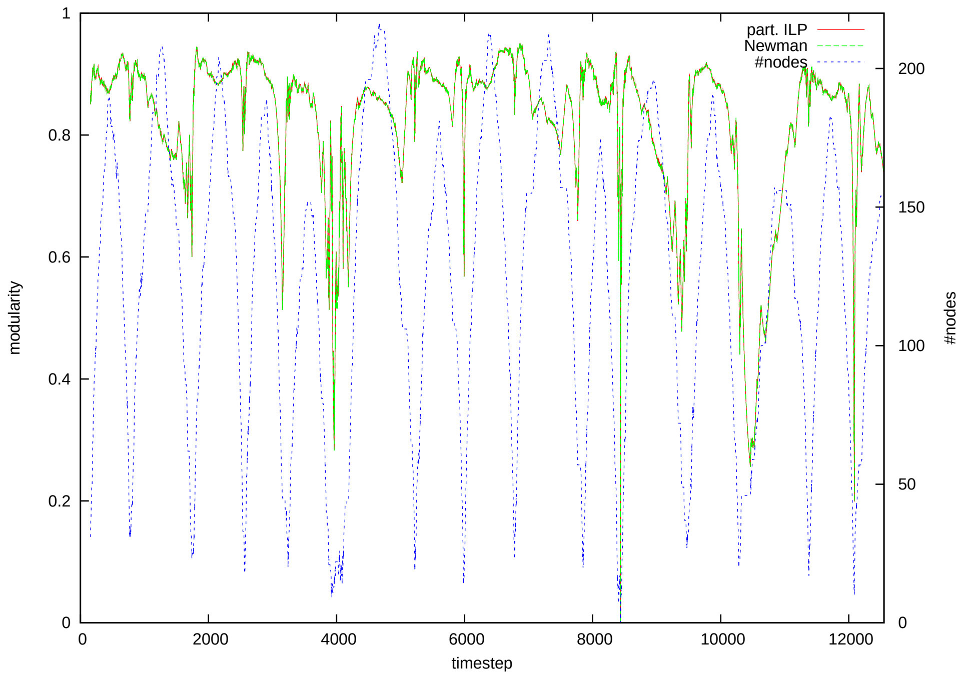

Figure D.1: The graph shows the modularity values of the partial ILP algorithm and the Newman algorithm for all timesteps during the evolution of the email graph for September 07 with a connection lifetime of 12 hours.

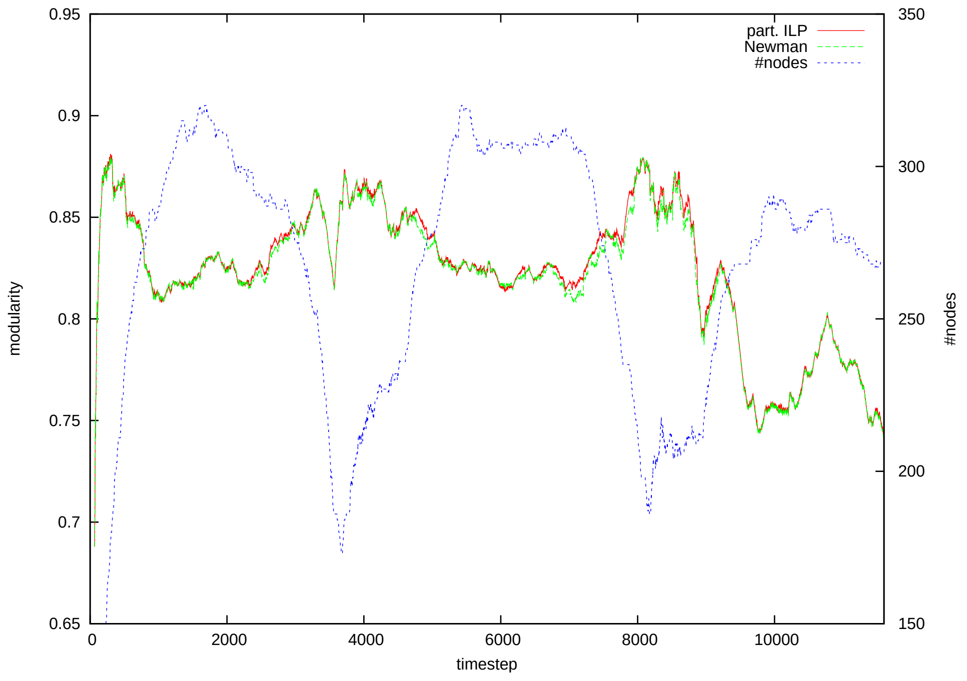

Figure D.2: The graph shows the modularity values of the partial ILP algorithm and the Newman algorithm for all timesteps during the evolution of the email graph for September 07 with a connection lifetime of 72 hours.

D.1.2 Calculation Time of the Dynamic Clustering of a Large Instance of the Email Graph

The analysis of the time needed to cluster single timesteps of the instance of the email graph presented in Section 6.8.2 with the partial ILP approach does not show the constancy predicted in Section 6.3. As the graph's components do not reach sizes significantly bigger than the maximum number of nodes being extracted from clusters during the construction of the partial linear program, the complexity of the ILP is exponential in the component size and thus the expected upper bound for the calculation time cannot be seen. Figure D.3 shows the running time for the same month but with a connection lifetime set to 72 hours. This leads to a significant increase in node count (180 to 320 nodes instead of 20 to 220 nodes) and edge count (up to 545 instead of 260 edges). Here it can be seen that the growth of the calculation time needed declines dramatically, once around 350 edges are inserted. It is explained by the assumption that now the components do reach sufficient size, and thus fill the complete pool of 30 possibly unclustered nodes. If any (pruned) clusters are obtained within the component, they will generate another 30 decision variables each. The complexity of the linear program then mostly depends on the graph's structure influencing the significance of a locally optimal solution.

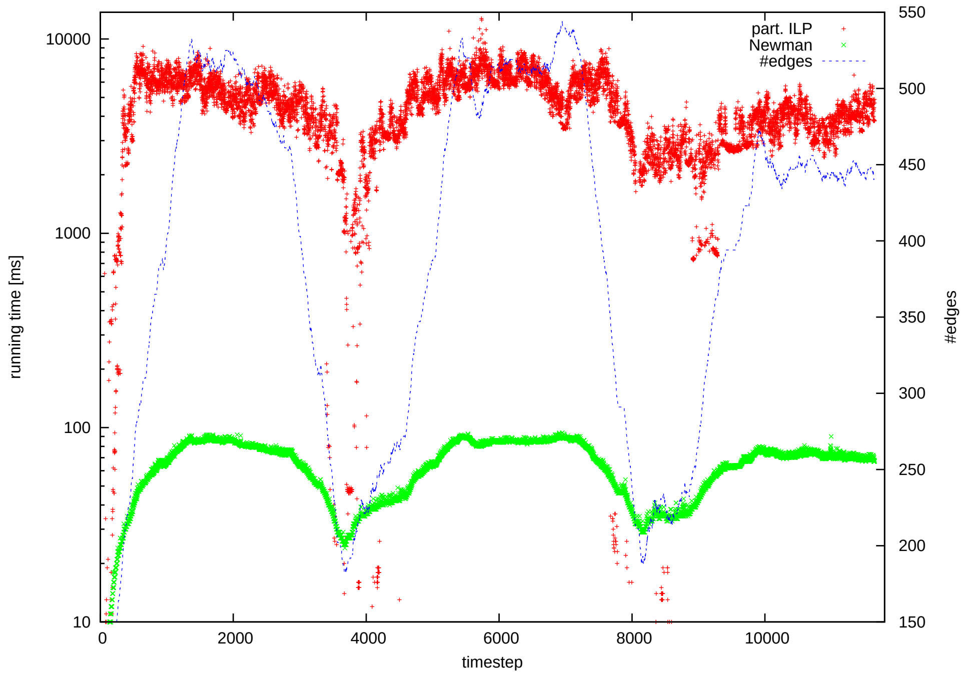

Figure D.3:

D.2 Further Results from Experiments with the Bi-Criterial ILP

In Section 5.4 a snapshot of the email graph spanning three chairs of the faculty is clustered using a bi-criterial ILP, in order to explicitly optimize the quality of the clusterings for each timestep as well as the similarity of the clustering structure between consecutive timesteps. The objective function employed for the graph G_t of each timestep t with respect to a given clustering $\mathcal{C}_{t-1}$ for the previous timestep is given as follows.

$$\begin{aligned}
&bicrit(G_t, \mathcal{C}_{t-1}) \\
&= mod_{ILP}(G_t) - b \cdot rand_{ILP}(\mathcal{C}_{t-1}) \\
&= \sum_{\{u,v\} \in V^2, u \neq v} \left(w(u,v) - \frac{w(u) \cdot w(v)}{2 \cdot w(E)} + 2 \cdot b \cdot Y_{uv} - b \right) \cdot X_{uv}
\end{aligned}$$

Figure D.4 shows the results of the bi-criterial clustering for three institutes and over the course of the first three weeks of September 2007. The observations made in Section 5.4 are confirmed by the results for the longer period. It can easily be seen, that the clustering calculated with the weighting factor $b = 0.0$ has the highest (even optimal) index values but the highest fluctuation in clustering structure as well. For larger values for b, the quality decreases and stability within the partitioning is established, as the optimization of the Rand distance measure is emphasized.

Note that the curves have no strict order. For example, it can be observed that the Rand distance curve for a weighting factor of $b = 5.0$ sometimes yields a higher value than the curve for $b = 2.5$. This can be explained by the particular structure of the previous timestep's clusterings that do not have to be alike. Therefore the impact of the modifications necessary in order to achieve the maximal combination of modularity and Rand distance with respect to the different weighting factors b is not comparable. For $b = 5.0$ the previous timestep's clustering might result in such a bad modularity value for the current timestep's graph, that a large number of modifications is necessary in order to inhibit a large decrease in the index value (and therefore the combination of both criteria). Thus a higher Rand distance is tolerated. The previous clustering for the same timestep and $b = 2.5$ might be close or at least closer to an optimal modularity clustering and hence the number of modifications needed to optimize the combination is small.

Additionally, the graph reveals its decomposition due to most connections timing out over the two weekends around the timesteps 600 and 1150 (as September 2007 started with a Saturday, the first weekend cannot be spotted). For those snapshots of the graph, the clustering becomes trivial as only 2 nodes and 1 edge are conserved over the first visible weekend, and 3 nodes and 3 edges over the second one. Thus, modularity shrinks to 0 and the distance to the previous clustering is 0 as well, as it contains the same groups. As the graph regrows after the weekends, the values normalize. The algorithm is able to adapt to the situation.

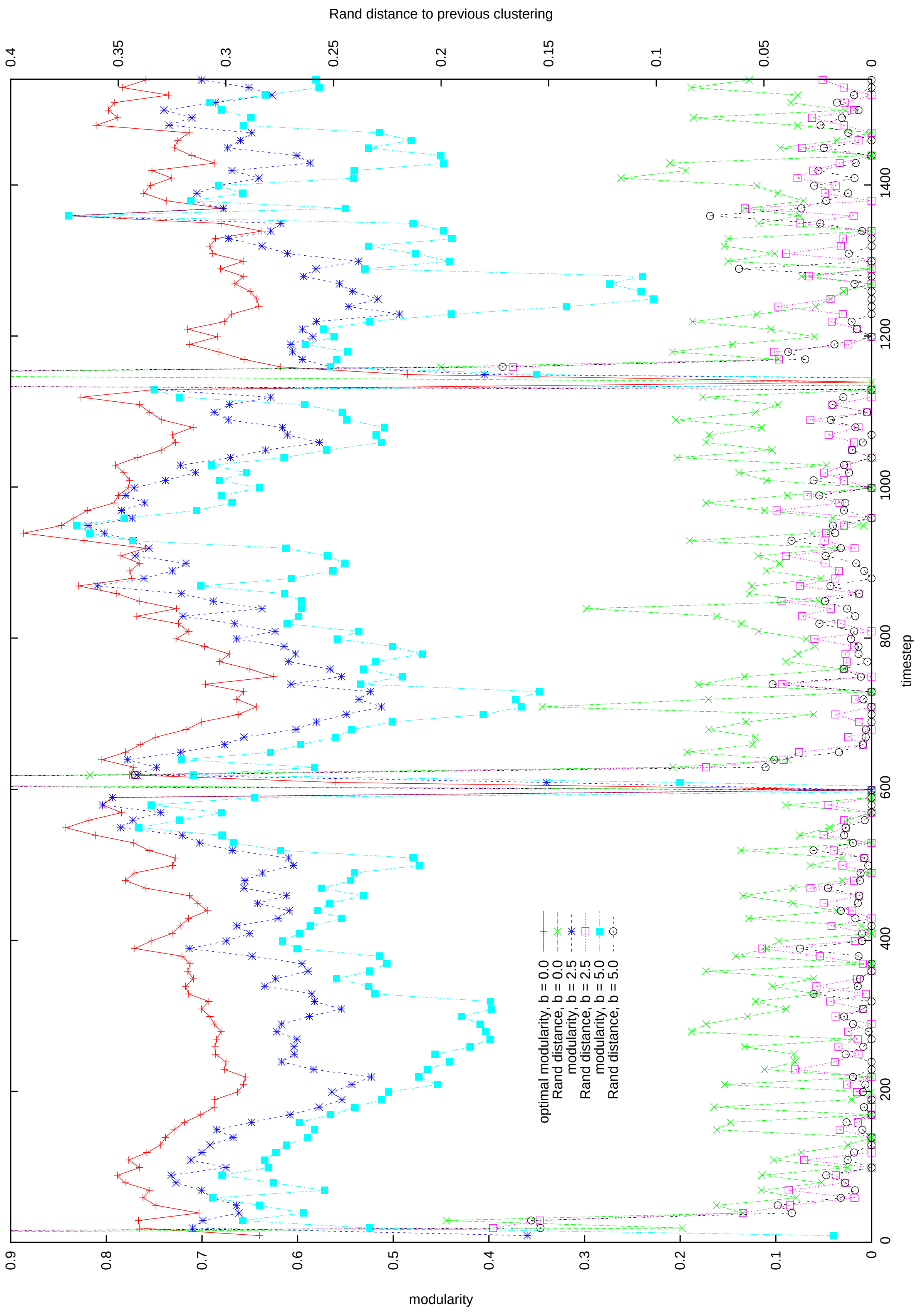

Figure D.4: The graphs show the modularity and Rand distance measure for the dynamic clustering of a fraction of the email graph (first three weeks of September 2007) with a connection lifetime of 24 hours. Different weighting factors have been applied, the step size is 10.

List of Figures

Bibliography

[1] BOLTEN, E., SCHLIEP, A., SCHNECKENER, S., SCHOMBURG, D., AND SCHRADER, R. Clustering protein sequences—structure prediction by transitive homology. *Bioinformatics 17*, 10 (2001), 935–941.

[2] BRANDES, U., DELLING, D., GAERTLER, M., GÖRKE, R., HÖFER, M., NIKOLOSKI, Z., AND WAGNER, D. On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering 20*, 2 (Feb. 2008), 172–188.

[3] BRANDES, U., GAERTLER, M., AND WAGNER, D. Experiments on Graph Clustering Algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)* (2003), vol. 2832 of *Lecture Notes in Computer Science*, Springer, pp. 568–579.

[4] CHAKRABARTI, D., KUMAR, R., AND TOMKINS, A. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), pp. 554 – 560.

[5] CHARIKAR, M., GURUSWAMI, V., AND WIRTH, A. Clustering with qualitative information. *Journal of Computer and System Sciences* (2005), 360–383.

[6] DELLING, D. Analyse und Evaluierung von Vergleichsmaßen für Graphclusterungen. Diplomarbeit, Universität Karlsruhe (TH), Fakultät für Informatik, Feb. 2006.

[7] EADES, P., LAI, W., MISUE, K., AND SUGIYAMA, K. Preserving the mental map of a diagram. In *Compugraphics* (1991), pp. 34–43.

[8] FACEBOOK INC. Facebook. http://www.facebook.com/.

[9] FORTUNATO, S., AND BARTHÉLEMY, M. Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the USA 104*, 1 (2007), 36–41.

[10] FOWLKES, E. B., AND MALLOWS, C. L. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association 78*, 383 (September 1983), 553–569.

[11] FRED, A. L., AND JAIN, A. K. Robust data clustering. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2003), pp. 128–136.

[12] GAERTLER, M., GÖRKE, R., AND WAGNER, D. Significance-Driven Graph Clustering. In *Proceedings of The Third International Conference on Algorithmic Aspects in Information and Management (AAIM'07)* (2007), M.-Y. Kao and X.-Y. Li, Eds., vol. 4508 of *Lecture Notes in Computer Science*, Springer, pp. 11–26.

[13] GAERTLER, M., GÖRKE, R., WAGNER, D., AND WAGNER, S. How to Cluster Evolving Graphs. In *Proceedings of the European Conference of Complex Systems (ECCS'06)* (Sept. 2006).

[14] GIRVAN, M., AND NEWMAN, M. E. J. Community structure in social and biological networks. *PNAS Applied Mathematics 99*, 12 (June 2002), 7821–7826.

[15] GOOGLE INC. Orkut. http://www.orkut.com/.

[16] GÄRTLER, M. *Algorithmic Aspects of Clustering*. PhD thesis, Institut für Theoretische Informatik, Universität Karlsruhe, 2007.

[17] IOACHIM, I., DESROSIERS, J., DUMAS, Y., SOLOMON, M., AND VILLENEUVE, D. A request clustering algorithm for door-to-door handicapped transportation. *Transportation science 29* (1995), 63–78.

[18] KEMPE, D., AND AGARWAL, G. Modularity-maximizing graph communities using mathematical programming.

[19] KRINGS, M. LP-basierte Heuristiken für Graphenclusterung mit Modularity als Qualitätsmaß. Studythesis, Institut für Theoretische Informatik, Universität Karlsruhe, February 2008.

[20] KUHN, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2* (1955), 83–87.

[21] LENSTRA, H. W. Integer Programming with a Fixed Number of Variables. *Mathematics of Operations Research 8*, 4 (November 1983), 538–548.

[22] LIN, C., AND GERLA, M. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications 15*, 7 (1997), 1265–1275.

[23] MEILĂ, M. Comparing clusterings by the variation of information. *Learning Theory and Kernel Machines 2777* (2003), 1173–187.

[24] MEILĂ, M., AND HECKERMAN, D. An experimental comparison of model-based clustering methods. *Machine Learning 42* (2001), 9–29.

[25] MISUE, K., EADES, P., LAI, W., AND SUGIYAMA, K. Layout adjustment and the mental map. *Journal of Visual Languages and Computation 6* (1995), 183–210.

[26] NEWMAN, M. E. J. Fast algorithm for detecting community structure in networks. Tech. rep., Department of Physics and Center for the Study of Complex Systems, University of Michigan, Sep 2003.

[27] NEWMAN, M. E. J., AND GIRVAN, M. Finding and evaluating community structure in networks. *Phys. Rev. E 69*, 2 (Feb 2004), 026113.

[28] OPEN SOURCE. lp_solve. http://sourceforge.net/projects/lpsolve, 2007.

[29] RAND, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66* (December 1971), 846–850.

[30] SHANNON, C. E. A mathematical theory of communication. *Bell System Technical Journal 27* (July, October 1948), 379–423,623–656.

[31] STUDIVZ LTD. studiVZ. http://www.studivz.net/.

[32] SØRENSEN, T. J. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biologiske Skrifter / Kongelige Danske Videnskabernes Selskab 5* (1948), 1–34.

[33] YWORKS GMBH. yFiles for Java. http://www.yworks.com/en/products_yfiles_about.html. 2007.

[34] ZACHARY, W. W. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research 33* (1977), 452–473.