Systems Research Institute, Polish Academy of Sciences

Łukasz Bolikowski
ICM, University of Warsaw

Methods of Semantic Drift Reduction in Large Similarity Networks

Ph.D. Thesis

Supervisor: Prof. Marek Niezgódka
ICM, University of Warsaw

Warsaw, December 2009

Abstract

We have investigated the problem of clustering documents according to their semantics, given incomplete and incoherent hints reflecting the documents' affinities. The problem has been rigorously defined using graph theory in set-theoretic notation. We have proved the problem to be NP-hard, and proposed five heuristic algorithms which deal with the problem using five quite different approaches: a greedy algorithm, an iterated finding of maximum cliques, energy minimization inspired by molecular mechanics, a genetic algorithm, and an adaptation of the Girvan-Newman algorithm. As a side effect of the fourth heuristic, an efficient and aesthetically appealing method of visualization of the large graphs in question has been developed.

The approaches have been tested empirically on the network of links between articles from over 250 language editions of Wikipedia. A thorough analysis of the network has been performed, showing surprisingly large semantic drift patterns and an uncommon topology: a scale-free skeleton linking tight clusters. It has been demonstrated that, using a blend of the proposed approaches, it is possible to automatically detect, and to a large extent eliminate, the semantic drift in the network of links between the language editions of Wikipedia. Last but not least, an open-source implementation of the proposed algorithms has been documented.

To my wife Duygu and my son Leon

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In the opening chapter, we shall present the framework of our research. First, we shall briefly sketch the background of the research. Next, we shall introduce the essential notation and formulate the research problem. Finally, we shall state the key contributions, and outline the thesis structure.

1.1 Background

Let us take a look at a couple of topics that are relevant to our research problem. We wish to present the research problem in the wider context of the global knowledge infrastructure.

This section is structured as follows. We start with a short summary of a potential as well as challenges of the information revolution in general and the Semantic Web in particular. Next, we identify a class of semantic networks that exhibit a common problem, namely, the problem of semantic drift. Having done that, we describe one notable example of the networks in question: the so-called interlanguage links in Wikipedia.

1.1.1 Information Age

Approximately 1.5 billion people[1] enjoy instant access to approximately 240 million sites[2] on the Internet. Access to information is easier than it was ever before, and the present times have been dubbed the Information Age.

New content is either transferred from physical media, such as books, journals, paintings, audio recordings, etc. in the process known as digitization, or it is born-digital. There are countless ongoing digitization projects, including: Google Books

[1]Source: Internet World Stats, `http://www.internetworldstats.com/`

[2]Source: Netcraft, `http://news.netcraft.com/archives/2009/06/17/june_2009_web_server_survey.html`

Library, JSTOR, Project Gutenberg, Times Online, and World Digital Library. To illustrate the scale of the efforts and their significance, let us mention that: Times Online provides on-line access to all the issues from 1785 to 1985[3], JSTOR contains over 32 million pages of documents from over four centuries[4], while World Digital Library "makes available on the Internet, free of charge and in multilingual format, significant primary materials from cultures around the world"[5]. The latter is not an exception: digitized content is often available for free, often under liberal copyright licenses. For example, Deutsche Fotothek has released over 250 thousand photos under the Creative Commons BY-SA 3.0 license.

Articles in most, if not all, of the contemporary peer-reviewed journals are born-digital, meaning they are created as digital documents in the first place, skipping the process of digitization. Some of the journals, such as First Monday, don't even have a printed version. As in the case of digitized works, many of the journals are so called "Open Access journals", i.e., are available online free of charge. However, the Web does not simply mimic the forms of information exchange known from the pre-digital times. Wikipedia, "the free encyclopedia that anyone can edit", with over 10 million articles in over 250 languages in an excellent example of a novel form of knowledge communication.

1.1.2 Semantic Web

The unprecedented supply of information creates unique opportunities, but also brings new challenges. The first problem that a novice web user faces is: how to find *valuable* information in the ocean of documents, articles, blog notes, and other web pages. Google and other search engines implement heuristics (such as Google's PageRank) which do a remarkably good job finding the pages relevant to the user's query. One of the next steps is to index *knowledge* instead of *data*: harvest and infer semantic relationships between entities, allowing users to formulate more precise queries [9]. The Semantic Web, in short, is an attempt to encapsulate the available knowledge in the form of subject-predicate-object triples and process the knowledge to users' benefit. The triples represent relations between entities represented by well-defined resource identifiers (URI). This data model is known as Resource Description Framework (RDF).

There are several established vocabularies, called ontologies, which define relationships within a given domain. For example Dublin Core allows to express properties of documents, such as: "X is the author of Y", "X is the title of Y", or "X was referenced by Y". Friend of a Friend, or FOAF, expresses relations between people and people's properties, such as: "X's e-mail address is Y", "X knows Y", or "X is interested in Y".

[3]Source: Times Online, `http://archive.timesonline.co.uk/`

[4]Source: JSTOR, `http://www.jstor.org/page/info/about/archives/facts.jsp`

[5]Source: World Digital Library, `http://project.wdl.org/`

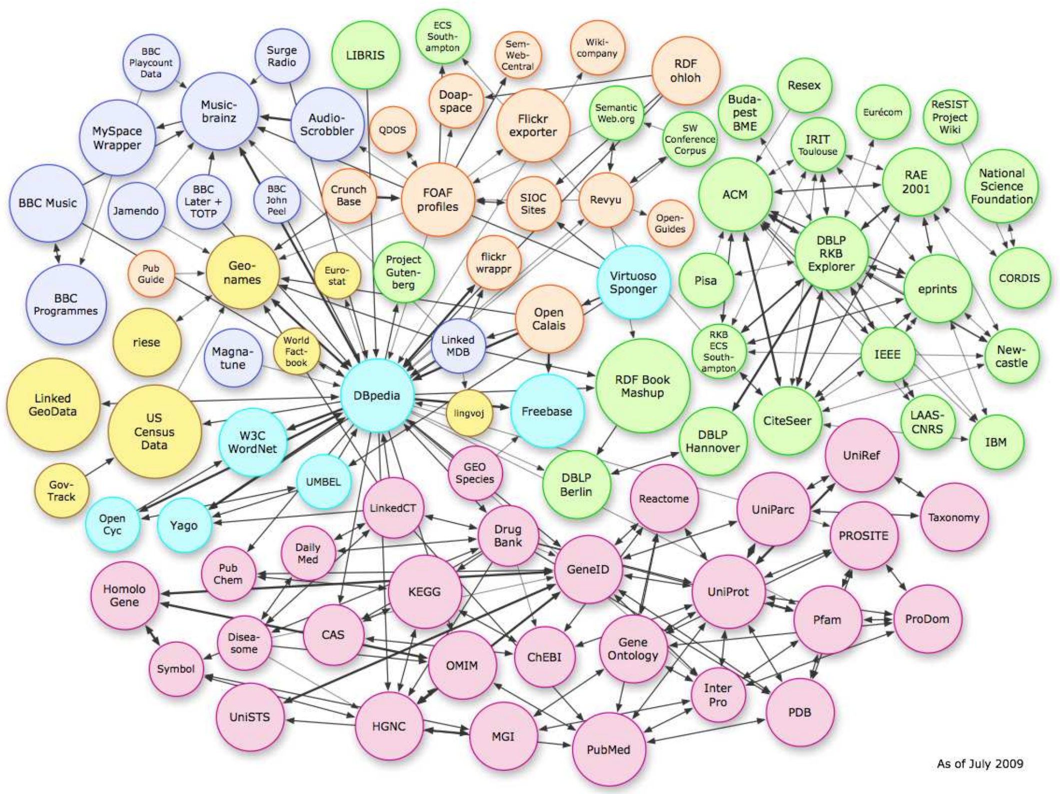


Figure 1.1: RDF data sets that have been published and interlinked by the Linking Open Data project as of July 2009. Each bubble represents an RDF data set, each connection represents semantic links between the sets. Colors represent topics. (author: Anja Jentzsch, license: CC-BY-SA 3.0)

Concurrently, there are efforts to extract knowledge (structured information) from the existing data sources. One such project, DBpedia [5] "harvested" Wikipedia and extracted 274 million relations between 2.6 million entities corresponding to the Wikipedia articles.

Last but not least, there are query languages, such as SPARQL, which provide access to the inferred knowledge. Sir Tim Berners-Lee, the father of both the World Wide Web and the Semantic Web, argues that the possibility of combining knowledge (structured information) obtained from several sources will further revolutionize the way we access information on the Web [10]. Continuing the previous example: a user can ask, in a single query, for e-mail addresses of authors of documents that referenced a given document. Such query would include verbs defined in both Dublin Core and Friend of a Friend ontologies. Another example: a single SPARQL query in DBpedia returns a list of German musicians born in Berlin before 1900. Let us note that none of the above queries are feasible using the "traditional" search engines.

1.1.3 Same knowledge, many locations

In a typical setting, the same knowledge is stored in a number of independent repositories. For example, a scholarly article's metadata is stored in a number of virtual libraries, a word is defined in a number of dictionaries, a movie's details are presented in a number of movie databases, a big city, a notable event, or a famous person is described by a number of encyclopedias.

A repository may provide links to other repositories covering the same domain. For example: in DrugBank, a drug database, each entry contains references to corresponding entries in other databases from the domain, such as PharmGKB or RxList. In Wikipedia, each article in a given language edition features a list of corresponding articles in the other language editions.

The correspondence relations mentioned above may be generalized to a semantic relation: "X corresponds to Y", or in other words: "X is equivalent to Y". This relation satisfies the properties of the set-theoretical equivalence relation. Let us recall that the set-theoretic equivalence relation $\equiv$ is a relation that satisfies the following three properties:

1. reflexivity: $\forall_X X \equiv X$
2. symmetry: $\forall_{X,Y} (X \equiv Y) \Rightarrow (Y \equiv X)$
3. transitivity: $\forall_{X,Y,Z} (X \equiv Y) \land (Y \equiv Z) \Rightarrow (X \equiv Z)$.

Examples include relations such as: "X is a homologue of Y" (among biological sequences) or "X is a synonym of Y" (among words).

However, in the case of independent knowledge databases, the equivalence relation "X corresponds to Y" is necessarily stored without coherence constraints, i.e., neither symmetry nor transitivity can be guaranteed. Lack of these constraints may lead, among other things, to a semantic drift.

1.1.4 Semantic drift

As it was noted above, an actual representation of an equivalence relation in a distributed knowledge database may substantially differ from its idealized model. Generally speaking, two classes of problems are prone to emerge (cf. Figure 1.2): missing links and excess links. The first class contains triples that are not present, although the relation between the corresponding entities holds while the other class contains triples that are present, although the relation between the corresponding entities does not hold.

The are numerous reasons why the actual triples do not correctly reflect the intended relation. It may be due to: concurrent edits by editors whose interpretations of

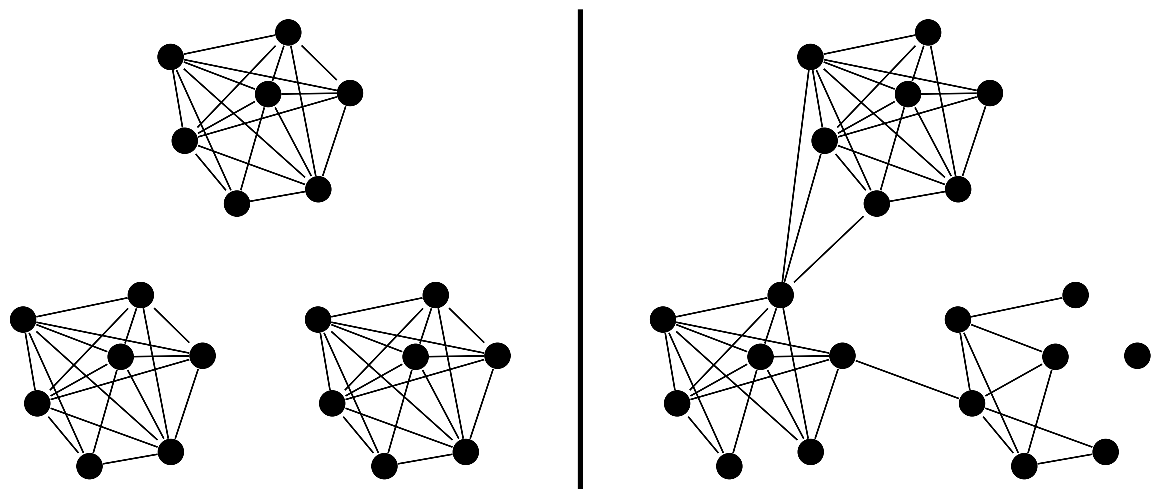


Figure 1.2: Illustration of semantic drift in a semantic relation, showing potential discrepancy between an "ideal" relation and its actual representation. The bullets represent entities, and segments represent relations between them. Sample equivalence relationship between entities is presented in the left panel. When the relationship is stored without enforcing data coherence (due to distributed storage, lack of co-ordination between the editors, etc.), then the stored relation may look as in the right panel. Note that non-equivalent entities are connected by a series of links which are believed to represent an equivalence relation.

the relation are incompatible, a mistake during data entry, relocation of an entry which used to be a target of a reference, etc.

What are the consequences of the abovementioned problems? A missing link causes that the equivalence between two entities is not reflected in the system and thus not visible to a potential user. An excess link causes that non-equivalent entities are treated in the system as if they were equivalent. A series of excess links escalates the problem, creating a semantic drift. The effect may be compared to the children's game of "Broken telephone": a series of inexact relays of a message may cause a great discrepancy between the message sent by the first child and the message received by the last.

There are two simple solutions, which may, to a certain extent and under certain conditions, alleviate the problems in question.

The first solution is to switch to another relation, which binds an entity to the equivalence class of the original relation. A contrived example: instead of a relation "X lives in the same city as Y", one may introduce a relation "X lives in Y" and infer that "X lives in the same city as Y" holds whenever there exists Z such that X lives in Z and Y lives in Z. Cities, which were the equivalence classes of the original relation, are now referenced directly. This trick does not work, however, when the equivalence class in not a tangible entity. For example: word meanings, which are the equivalence classes of the "X is a synonym of Y" relation, are not necessarily easily represented by an entity. Moreover, it is often not a viable option in the case of distributed storage of the triples in question, since it would require a certain level of centralization: a master repository, or at least a centralized identification of the represented concepts would have to be agreed upon by the participating partners. Digital Object Identifier (DOI), a *de facto* standard of identifying contemporary scholarly articles, is an example of a successful adoption of such a scheme.

Another solution, which may work in certain cases, is to perform symmetric and transitive closure of the triples at hand. It is likely to reduce the problem of the missing links, but on the other hand, it propagates false information conveyed by excess links.

None of the above simple solutions, however, is satisfactory in general, and a more robust approach is called for.

1.1.5 Wikipedia

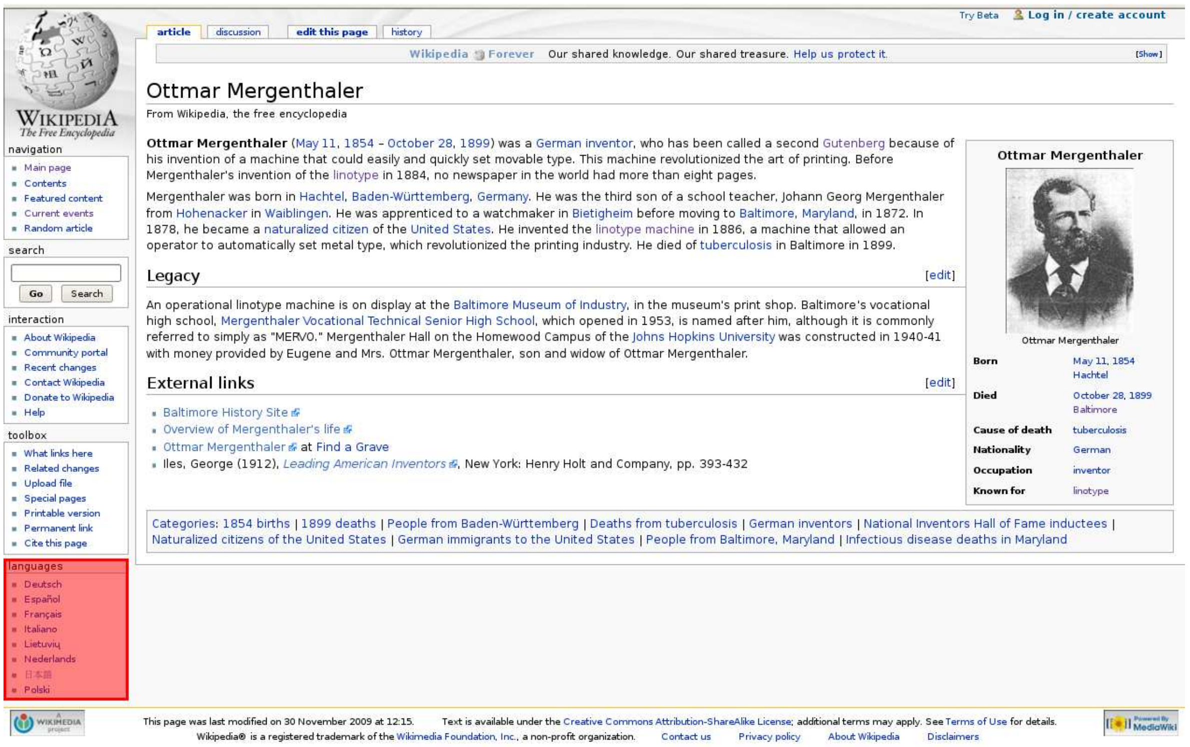
Throughout this thesis, Wikipedia will serve as the testing ground of the approaches developed. Therefore, let us present a brief overview of the technical aspects of Wikipedia that are relevant to our research.

Wikipedia is a free online encyclopedia that anyone can edit. The name is a portmanteau of the words "wiki" (a kind of collaborative online edition) and "encyclopedia". Wikipedia textual content is licensed under GNU Free Documentation License[6]. Wikimedia Foundation regularly publishes database dumps of all its projects, thus providing an excellent research material for various large network analyses. Our experimental results are based on two sets of database dumps: the semantic drift analysis in Chapter 3 is based on all the 262 language editions of Wikipedia as of August 27. 2008, while the experimental results in Chapter 5 are based on all the 265 language editions as of October 12, 2009.

In this short overview, we would like to focus on *articles*, *categories*, *page redirects* and *interlanguage links*, which are the building blocks of the researched network. It should be stressed that this is in no way a comprehensive description of the entire Wikipedia project or the software engine running it, as we deliberately omit most of their prominent features.

Each language edition of Wikipedia is powered by MediaWiki, a wiki engine. Content managed by MediaWiki is organized into *pages*, which are (within a language edition) uniquely identified by their titles. Each page resides in exactly one *namespace*, which may be thought of as its type. The main namespace, also referred to as article namespace, contains most of the textual content of a language edition. Approx. 84.8% of all the pages in all the language editions of Wikipedia are located there, and referred to simply as *articles*. The second most popular namespace (approx. 13.8% of all the pages) is the category namespace. A *category* is a sort of tag, and each article may belong to zero, one or more categories. To give an example from the English edition: the "London" article is, among others, in the categories "Capitals in Europe" and "Host cities of the Summer Olympic Games". From now on, we focus our attention on the

[6]Full text of the license: `http://www.gnu.org/copyleft/fdl.html`



Wikipedia Forever Our shared knowledge. Our shared treasure. Help us protect it.

Ottmar Mergenthaler

From Wikipedia, the free encyclopedia

Ottmar Mergenthaler (May 11, 1854 – October 28, 1899) was a German inventor, who has been called a second Gutenberg because of his invention of a machine that could easily and quickly set movable type. This machine revolutionized the art of printing. Before Mergenthaler's invention of the linotype in 1884, no newspaper in the world had more than eight pages.

Mergenthaler was born in Hachtel, Baden-Württemberg, Germany. He was the third son of a school teacher, Johann Georg Mergenthaler from Hohenacker in Waiblingen. He was apprenticed to a watchmaker in Bietigheim before moving to Baltimore, Maryland, in 1872. In 1878, he became a naturalized citizen of the United States. He invented the linotype machine in 1886, a machine that allowed an operator to automatically set metal type, which revolutionized the printing industry. He died of tuberculosis in Baltimore in 1899.

Legacy

An operational linotype machine is on display at the Baltimore Museum of Industry, in the museum's print shop. Baltimore's vocational high school, Mergenthaler Vocational Technical Senior High School, which opened in 1953, is named after him, although it is commonly referred to simply as "MERVO." Mergenthaler Hall on the Homewood Campus of the Johns Hopkins University was constructed in 1940-41 with money provided by Eugene and Mrs. Ottmar Mergenthaler, son and widow of Ottmar Mergenthaler.

External links

- Baltimore History Site
- Overview of Mergenthaler's life
- Ottmar Mergenthaler at Find a Grave
- Iles, George (1912), *Leading American Inventors*, New York: Henry Holt and Company, pp. 393-432

Ottmar Mergenthaler

Ottmar Mergenthaler

Born	May 11, 1854 Hachtel
Died	October 28, 1899 Baltimore
Cause of death	tuberculosis
Nationality	German
Occupation	inventor
Known for	linotype

Categories: 1854 births | 1899 deaths | People from Baden-Württemberg | Deaths from tuberculosis | German inventors | National Inventors Hall of Fame inductees | Naturalized citizens of the United States | German immigrants to the United States | People from Baltimore, Maryland | Infectious disease deaths in Maryland

This page was last modified on 30 November 2009 at 12:15. Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. See Terms of Use for details. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization. Contact us Privacy policy About Wikipedia Disclaimers

Figure 1.3: Screenshot of a Wikipedia page with interlanguage links. In the language editions using left-to-right scripts (such as the English edition using Latin alphabet) the interlanguage links are located on the left side of a page (highlighted in red).

pages in the two most popular namespaces mentioned above.

MediaWiki has a page aliasing mechanism through which any reference to page A is interpreted as a reference to another page B. This way, it is possible to refer to a page by using any of the titles associated with it. Policies of the English edition list a number of purposes for page redirects[7], which can be roughly divided into two groups. The first one includes cases of alternative spellings, common misspellings, or alternative terms for the page's subject – shortly, the cases when the two titles are semantically equivalent. For example, the pages "JFK" and "35th President of the United States" in the English edition are both *redirects* to the "John F. Kennedy" article in the same edition. The other group contains redirects from A to B, where B covers A, but has a broader meaning. In such cases it is advised that the redirect leads to the relevant section of B (should a relevant section exist). Example (English edition): "One pair", "Three of a kind" and "Straight flush" are all redirects to corresponding sections of the "List of poker hands" article. However, redirects to page sections are relatively rare and account for approx. 1.77% of all the redirects in all the language editions.

MediaWiki engine provides a means of indicating that a given page has a corre-

[7]See: `http://en.wikipedia.org/wiki/Wikipedia:Redirect`

	Articles		Categories	
Language	Pages	IL Links	Pages	IL Links
English	2 502 189	4 763 819	389 987	608 787
German	791 848	3 331 290	55 323	417 800
French	687 290	3 398 456	96 073	482 098
Polish	552 267	2 834 358	37 814	284 455
Dutch	513 295	2 927 619	37 202	238 934
Japanese	512 872	2 170 917	51 284	407 294
Italian	506 223	2 986 566	51 948	195 548
Portuguese	469 382	2 697 324	48 389	256 680
Spanish	389 929	2 542 999	61 377	376 110
...	...	...	...	...
Total	11 510 142	89 339 694	1 724 088	13 902 852

Table 1.1: Number of articles, categories, and corresponding outgoing interlanguage links for 10 largest language editions of Wikipedia (ordered by the number of articles). Based on the database snapshots available on August 27, 2008.

sponding one in another wiki (possibly, but not necessarily, powered by MediaWiki). In Wikipedia, this functionality is primarily used to present corresponding articles or categories in other language editions (see Figure 1.3). Such relations are formally called *interlanguage links*, but often referred to as *interwiki links* [8].

The intended meaning of an interlanguage link (ILL), placed on page A and pointing to page B in a different language edition, is that A and B are on the same subject, i.e., semantically equivalent. Example: English article entitled “London” contains, among others, an interlanguage link to the French article entitled “Londres” and to the Finnish article entitled “Lontoo”. From now on, for the sake of brevity, we shall prefix all the titles with language codes, e.g.: “en:London”, “fr:Londres”, “fi:Lontoo”. Let us point out a couple of important consequences of placing interlanguage links on pages. Firstly, the links stored that way are necessarily directed: “en:London” → “fr:Londres” and “fr:Londres” → “en:London” are two separate links, and technically, one does not imply the other. Secondly, since the links are stored across many autonomous databases, no coherence checks can be performed. Thus, neither *symmetry* nor *transitivity* of the links is guaranteed. Moreover, queries for incoming interlanguage links are not feasible.

Therefore, the interlanguage links in Wikipedia serve as an excellent example of an equivalence relation that is stored without coherence constraints. Huge amount of available data and free, easy access to it were the key reasons of picking Wikipedia's ILLs as the experimental data for this thesis.

[8]Technically, “interwiki link” is a broader term, including links to wikis other than just the other language editions of Wikipedia, but practically it is synonymous with “interlanguage link”

1.2 Notation and Definitions

Let us first introduce a set-theoretic notation for graphs, which will be used throughout this work. Readers unaccustomed with the graph theory may want to consult the classic introductory book by Bondy and Murty [13].

We focus our attention on undirected, vertex-colored, edge-weighted graphs. Such graph are defined as a quadruple $G = \langle V, E, \kappa, \mu \rangle$, where:

- V is a set of vertices;
- $E \subseteq \left\{ e \in 2^V \middle| |e| = 2 \right\}$ is a set of undirected edges;
- $\kappa : V \to K$ is a function representing colors of vertices;
- $\mu : E \to \mathbb{R}_+$ is a function representing weights of edges.

Occasionally, unweighted graphs are investigated as well. In these cases, it is assumed that $\mu \equiv 1$, and such a graph is denoted by a triple $G = \langle V, E, \kappa \rangle$.

We say that a graph G is **κ-partite** iff no two vertices of the same color are connected by an edge:

$$\forall_{v,w \in V}\ \kappa(v) = \kappa(w) \Rightarrow \{v, w\} \notin E \tag{1.1}$$

At times we reference algorithms that only accept κ-partite graphs on input. In those cases, a trivial transformation that removes all the edges connecting vertices of the same color (thus yielding a κ-partite graph) becomes handy. To put it in set-theoretic terms, the transformation creates a graph:

$$G' = \langle V, E', \kappa, \mu|_{E'} \rangle \tag{1.2}$$

where:

$$E' = \left\{ \{v, w\} \in E \middle| \kappa(v) \neq \kappa(w) \right\} \tag{1.3}$$

Let **neighborhood** of a vertex $v \in V$ in graph G be defined as:

$$\epsilon(v) := \left\{ w \in V \middle| \{v, w\} \in E \right\} \tag{1.4}$$

Let a **partition** (or **clustering**) π of a graph G be defined as any function $\pi : V \to 2^V$ satisfying the following properties:

$$\forall_{v \in V}\ v \in \pi(v) \tag{1.5}$$

$$\forall_{v,w \in V}\ \pi(v) = \pi(w) \vee \pi(v) \cap \pi(w) = \emptyset \tag{1.6}$$

$$\forall_{v \neq w \in V}\ \pi(v) = \pi(w) \Rightarrow \kappa(v) \neq \kappa(w) \tag{1.7}$$

The first two conditions simply state that π splits V into disjoint subsets that sum up to V. The third condition states that any subset must contain at most one vertex of each color. The set of all the possible partitions of a graph G will be denoted by $P(G)$.

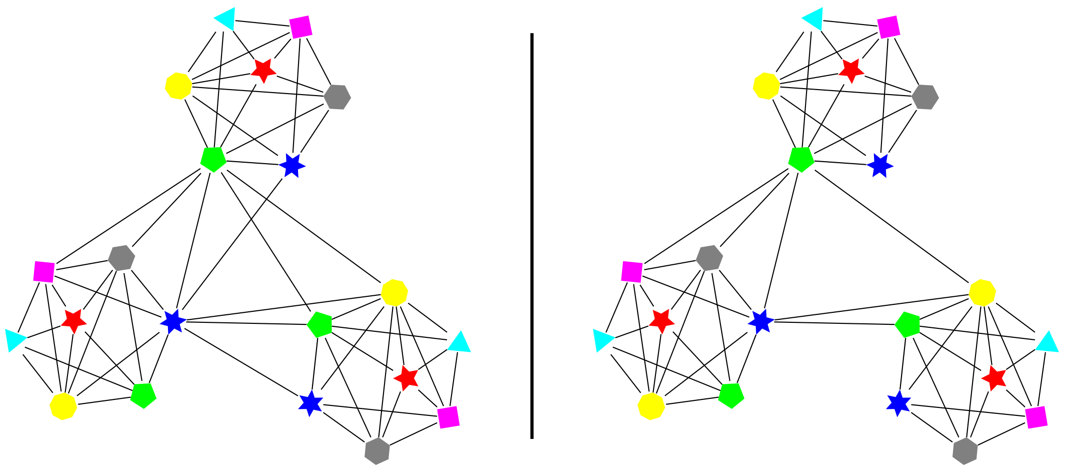


Figure 1.4: Example of a κ-partite graph. The one on the right side is a κ-partite graph, while the one on the left side is not; neither is coherent. The graph on the right was obtained from the graph on the left through the transformation described in the text. Note: in order to improve legibility of b/w printouts, in this figure, as well as in all the following ones, node colors are paired with their shapes, and thus: triangle = cyan, square = magenta, pentagon = green, hexagon = gray, octagon = yellow, pentagram = red, hexagram = blue.

Note that a partition π is unambiguously represented by its range:

$$\Pi = \bigcup_{v \in V} \pi(v) \tag{1.8}$$

The two forms (π and Π), being equivalent, are used interchangeably. This should not lead to confusion, since one of the forms is a function, while the other is a set.

We say that an edge $e = \{v, w\} \in E$ is **incoherent** with a partition $\pi \in P(G)$ iff $\pi(v) \neq \pi(w)$. In other words, an edge is incoherent with a partition iff it connects two different subsets of the partition. Let $I_G(\pi)$ denote the set of all the edges of G incoherent with π:

$$I_G(\pi) := \left\{\{v, w\} \in E \middle| \pi(v) \neq \pi(w)\right\} \tag{1.9}$$

Next, we say that a partition $\pi \in P(G)$ is **incoherent** iff $I_G(\pi) \neq \emptyset$. In other words, a partition of a graph is incoherent if there is at least one edge in the graph incoherent with the partition. Let us define a function $\|\cdot\|_G$ on π quantifying its incoherence with G as the sum of weights of all the edges incoherent with π:

$$\|\pi\|_G := \sum_{e \in I_G(\pi)} \mu(e) \tag{1.10}$$

Finally, we say that a graph G is **coherent** iff there is at least one coherent partition of the graph. There is an alternative characterization of a coherent graph, one that doesn't use the concept of coherent partitions: a coherent graph is such that each of its connected components contains at most one vertex of any given color.

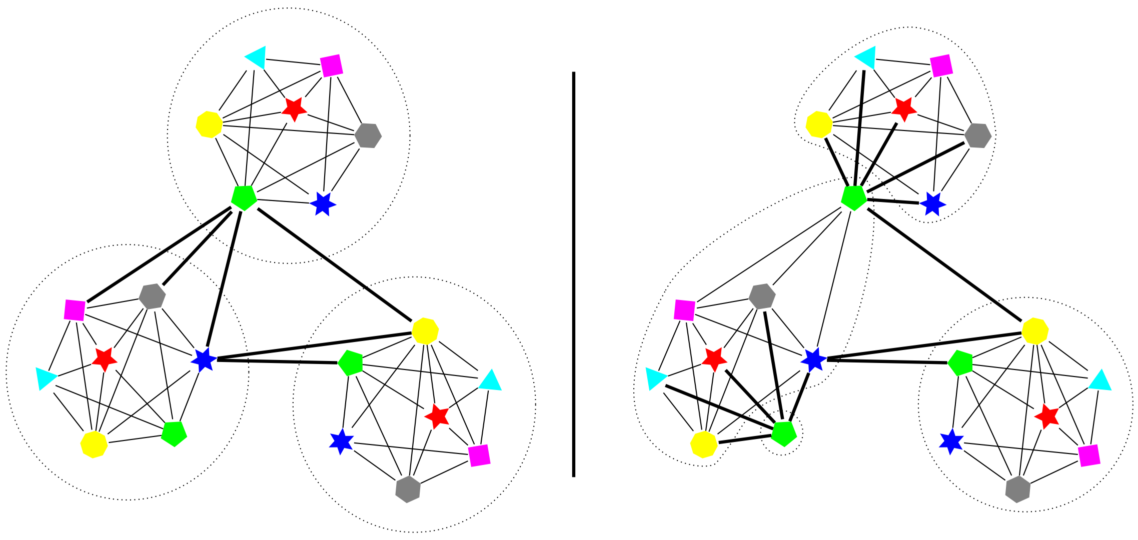


Figure 1.5: Two examples of graph partitions. The partitions are marked by dotted lines. Bold edges represent cuts associated with the partitions. In each of the graphs, the sum of weights of the bold edges is the measure (cost) of the partition, and thus, by definition, the measure of the cut.

A graph G is said to be **complete** iff it is coherent and addition of any new edge renders it incoherent. Alternatively: in a complete graph each connected component is a clique and contains at most one vertex of any given color.

A **natural partition** π_G of a coherent graph G is a partition that assigns to each vertex v the set of vertices of the connected component of G containing v. Thus, a natural partition is always coherent.

For any partition π of graph $G = \langle V, E, \kappa, \mu \rangle$, a **truncated graph** is defined as:

$$G|_\pi := \langle V, E', \kappa, \mu|_{E'} \rangle \tag{1.11}$$

where:

$$E' := \big\{ \{v, w\} \in E \big| \pi(v) = \pi(w) \big\} \tag{1.12}$$

Therefore, for any partition π the truncated graph $G|_\pi$ is coherent. In other words, a truncated graph $G|_\pi$ is created by removing all the edges incoherent with π.

For a given graph G and a set of edges $F \subseteq E$, if $\langle V, E \setminus F, \kappa, \mu|_{E \setminus F} \rangle$ is coherent, then F is called a **cut** with weight $\|F\|_G := \sum_F \mu(F)$. It is worth to notice that for a given graph G, each partition π has a corresponding cut $I_G(\pi)$ and $\|\pi\|_G = \|I_G(\pi)\|_G$. We say that a cut F is **locally-minimal** iff no proper subset of F is a cut.

1.3 Problem Statement

Putting the formal notation aside for a moment, we want to find a coherent graph that is the "closest" to the graph at hand. A coherent graph is, by definition, one in which

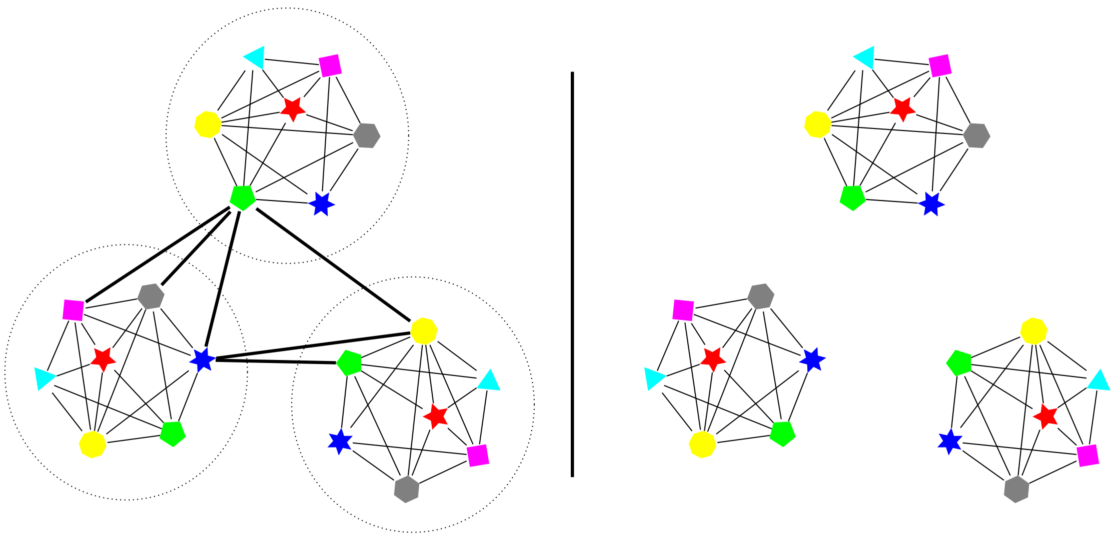


Figure 1.6: Example of a truncated graph. The graph G' on the right side is a truncated graph, obtained from the graph G and the partition π presented on the left side. All the edges incoherent with π (bold edges on the left side) were removed, and the resulting graph (on the right) is coherent, while the original graph (on the left) is not.

no two vertices in the same connected component have the same color. "Closeness" is measured by the sum of weights of edges that have to be removed in order to transform one graph to another.

Let us express the above using the nomenclature introduced in Section 1.2. The research problem is:

Problem 1.1 (Minimum Partition) *Given a graph $G = \langle V, E, \kappa, \mu \rangle$, find a partition π such that $\|\pi\|_G$ is minimal.*

An alternative, equivalent formulation of the problem asks for a cut, rather than a partition:

Problem 1.2 (Minimum Cut) *Given a graph $G = \langle V, E, \kappa, \mu \rangle$, find a cut F such that $\|F\|_G$ is minimal.*

Note that transitive closure of a coherent graph models an equivalence relation. Thus, we are effectively looking for a way to transform a relation to an equivalence relation in two steps: edge removal (which is difficult) and edge addition (which is trivial).

1.4 Key Contributions

The thesis presents a number of novel results:

1. We give a comprehensive description of the topology of so-called interlanguage links in Wikipedia. The network possesses an interesting and uncommon topology: its connected components are scale-free skeletons of tight clusters. Sizes of the connected components in the network have power-law distribution, and the largest connected component is roughly three orders of magnitude larger than expected.

2. We establish the computational complexity of the research problem. We show that the problem is NP-complete by constructing a polynomial-time reduction from a known NP-complete problem.

3. We propose five algorithms that return approximate solutions to the research problem. The algorithms use five quite different techniques:
   - a greedy algorithm;
   - an iterated finding of the maximum clique;
   - a numerical minimization of a multi-dimensional potential;
   - a genetic algorithm;
   - an adaptation of the Girvan-Newman algorithm for finding community structure.

4. As a side effect of one of the algorithms, we propose a method of visualizing the networks in question.

5. We carry out a quantitative evaluation of the proposed algorithms, with respect to both the quality of results and the running time. Our experimental data is the network of interlanguage links in Wikipedia.

6. We document the architecture of an open-source software package containing a data import module, implementations of all the proposed algorithms, a visualization module and a result export module. We have used the software to evaluate the proposed algorithms, and to generate edit recommendations that we have subsequently presented to the Wikipedia community.

1.5 Thesis Structure

The thesis is organized as follows:

Chapter 1 introduces the research problem together with its background, presents a notation used throughout the work, states the key contributions and outlines the structure. In the beginning, we show the wider context of the problem, the paradigm shift in knowledge exchange and the semantic networks. Moving on, we focus on one particular challenge faced by a class of semantic networks, namely, semantic drift. We show that the network of so-called interlanguage links in Wikipedia is a good illustration of the issue. Next, we decide on the vocabulary and symbols that are used in the thesis. Having done that, we formally state the computational problem. Finally, we state the key contributions and outline the structure of the thesis.

Chapter 2 summarizes the state of the art in the areas related to the research problem. In particular, we give a survey of similar computational problems, present advances in the understanding of topology and dynamics of social networks, and report state of the art in the techniques on which the algorithms presented in this thesis are based. Next, we give an account of various academic studies focusing on Wikipedia. Finally, we briefly delineate the limits of a translation process from the linguistic point of view.

Chapter 3 gives a thorough description of the interlanguage links in Wikipedia – a notable network to which the research problem applies. We present the topology, quantify the scale of semantic drift, and calculate several characteristics of the network. We successfully explain certain anomalies in the aggregated indicators, and present an algorithm that extracts a meaningful top-level structure of the network.

Chapter 4 presents the theoretical results: the computational problem is shown to be NP-complete, and five algorithms yielding approximate solutions are presented. The algorithms attack the computational problem from several sides, employing: a greedy strategy, a heuristic for the MAXIMUM CLIQUE problem, a numerical minimization of a potential, a genetic algorithm framework, and an algorithm reconstructing community structure.

Chapter 5 presents the experimental results: the proposed algorithms are evaluated on the network of interlanguage links in Wikipedia. We compare both the results and the running times of the proposed algorithms on over 86 thousand problem instances divided into three categories according to their size.

Chapter 6 gives a summary of the thesis, draws conclusions, and presents outlook. We recapitulate the main results and contributions of the thesis. Next, we draw conclusions from the results obtained. Finally, we list possible extensions of this thesis and directions of further research.

Appendix A documents technical details of processing the experimental data. We describe the the design of MediaWiki engine that powers all the language editions of Wikipedia. Next, we document the data import process and outline the design of a software package implementing the approaches proposed in this thesis.

Appendix B lists symbols and terms occurring in this thesis.

Chapter 2

Literature Review and State of the Art

2.1 Computational Problem

To our best knowledge, the main computational problem introduced in Chapter 1 has not yet been a subject of any academic study. However, a number of similar problems in surprisingly diverse contexts were studied.

Dahlhaus et al. [25] have established the computational complexity of a similar problem, called Multiterminal Cut Problem, which in turn generalizes the classic problem of finding the maximum flow and the minimum cut in a network [41]. The Multiterminal Cut problem asks for the cheapest way of separating a given set of nodes (terminals) in a network from one another. The authors show that the problem is NP-hard.

Problems similar to the one researched in this thesis also surface in the context of automatic optimization of work scheduling in distributed computing [95], also known as the Index Domain Alignment problem [63, 14], and in the context of decomposing an electrical network into groups of components [64]. He et al. have shown [53] NP-hardness of a problem called Minimum Orthogonal Partition, which differs from our research problem in one point only: it imposes a certain restriction on the cardinality of partitions. The restriction is natural in the context of distributed computing, as it expresses the fact that there is always a limited number of processors available.

2.2 Models of Network Dynamics

Several models of network dynamics exist in the literature. Let us recall three such models, arguably the most important ones, each named after its authors: Erdős-Rényi, Barabási-Albert, and Watts-Strogatz. In each case, the model is a procedure that creates an unweighted, undirected graph by random addition of edges or links, and statistical properties of the resulting graphs are investigated.

Erdős and Rényi [36, 37] examined random graphs constructed in the following way: first, n vertices are created, and then m edges out of $\frac{n(n-1)}{2}$ are chosen at random with equal probabilities. They found that for a number of fundamental structural properties there is a so-called "threshold function" $A(n)$ such that, if $\lim_{m,n\to+\infty} \frac{m}{A(n)} = 0$ then the the property occurs with probability 0, and if $\lim_{m,n\to+\infty} \frac{m}{A(n)} = +\infty$ then the the property occurs with probability 1. In particular, a giant component will almost certainly appear in a graph iff $\lim_{m,n\to+\infty} \frac{m}{n} > 1$. Vertex degrees in the Edrős-Rényi model follow the Poisson distribution.

Watts and Strogatz [104] proposed a different model. This time, n nodes and $m \gg n \log n$ edges initially formed a regular ring lattice. Next, each edge was rewired with probability p. Statistical properties of characteristic path length and clustering coefficient were studied as functions of p. It was observed that while $p = 0$ gives a perfectly ordered network and $p = 1$ a perfectly random one, the intermediate states yield so-called "small-world" networks, echoing the famous "small-world" experiment conducted by Milgram[1] [98].

Barabási and Albert [6, 2] take a different approach, based on the "rich-get-richer" observation that in the real world situations a vertex with higher number of incident edges is more likely to attract new edges. In their model, the constructed graph initially consists of m_0 vertices and no edges. In each step, a new vertex is added and connected to $m < m_0$ vertices already present. The probability of connecting to a given vertex is proportional to the vertex' degree. Barabási and Albert show that this model yields a so-called "scale-free" network in which the vertex degrees follow power-law distribution.

The network of interlanguage links in Wikipedia studied in this thesis does not fit any of the above three models. However, as it is shown in Chapter 3, skeletons of individual connected components are indeed scale-free.

The proposed models attracted a lot of attention and their properties were further studied. Dorogovtsev and Mendes [29] offer a continuous approach to the evolution of scale-free networks, Barabási et al. [7], Dorogovtsev et al. [27], as well as Ravasz et al. [88] show that certain deterministic fractal-like structures exhibit properties similar to those of random scale-free networks. Robustness and error tolerance of the networks was studied by Albert et al. [86], Callaway et al. [19], Dorogovtsev et al. [28], as well as Duch and Arenas [31]. Finally, Bollobás [11] gives a thorough account of the results in the theory of random graphs.

[1]Stanley Milgram is famous for two experiments: the "small-world" experiment and a study of obedience [68]. The latter is arguably even more popular, and usually referred to simply as "the Milgram experiment". The results of the former, on the other hand, were immortalized by a Broadway play "Six Degrees of Separation" by John Guare, which was in turn adapted for a film starring Stockard Channing, Will Smith, Donald Sutherland and Sir Ian McKellen.

2.3 Power-Law Distributions

A power-law distribution is a probability distribution such that $P(x) \propto x^{-\gamma}$. We have already referenced this relation (also known as "Zipf's law" or "Pareto distribution") while describing the degree distributions in scale-free networks (cf. Section 2.2). Power-law distributions are encountered[2] in extremely diverse settings, for example: the distribution of bank assets [85], frequency of Japanese family names [70], number of firm bankruptcies in a given day [56], weather persistence [18], city sizes [67], occurrences of DNA base pair sequences [66], number of sent e-mails [34], and occurrences of numbers in the Web [30] all follow the power law. For an excellent description of the distribution and numerous examples of its occurrences, see Newman [75] and Clauset et al. [23].

In Chapter 3 we discover another occurrence of a power-law distribution.

2.4 Social Networks

Another important class of problems deals with so-called community detection in social networks. For an excellent introduction to the field of social network analysis, see Wasserman and Faust [103].

Although the graphs researched in the community detection problems are not vertex-colored, which is a crucial property in our research, it is nevertheless fitting to become familiarized with the results in this field, and certain results can and will be reused in our research.

We are particularly interested in a community detection algorithm by Girvan and Newman [48]. The algorithm utilizes a concept of *edge betweenness* that quantifies the importance of each edge in the graph. The concept is derived from a similar vertex measure known as *betweenness centrality*, proposed independently by Freeman [43] and Anthonisse [4]. Brandes developed an elegant algorithm for fast calculation of betweenness centrality [15]. Newman and Girvan [77] show that two betweenness measures: one motivated by resistor networks, the other by random walks, are equivalent and, in the same paper, introduce the important notion of modularity.

The list of articles researching methods of identifying communities in social networks is much longer. M. E. J. Newman, a very prolific researcher in the field of social network analysis, investigates scientific collaboration network [71, 72], offers certain improvement over the original Girvan-Newman algorithm [74], reviews existing community detection algorithms [73], rewrites in terms of eigenvalues and eigenvectors the problem of finding the maximal modularity and comes up with a fast and accurate algorithm [76].

[2]To be more precise: data from diverse settings fit well to power-law distributions. In a typical case, we cannot be certain that a given set of values is drawn from a power-law distribution.

Newman and Park [78] show properties of social networks, such as assortative mixing, that are absent in other types of networks. Clauset et al. [22] show yet another algorithm and evaluate it on the network of purchases on Amazon, the algorithm is further discussed by Gulbahce and Lehmann [50]. Radicchi et al. [87], Flake et al. [40], Capocci et al. [21] show other community identification methods. Finally, Danon et al. [26] compare the performance of community structure identification algorithms .

Note: various centrality measures are related to the robustness and error tolerance analyses referenced in Section 2.2.

2.5 Cliques and Cores

Seidman [93] proposed a notion of k-core, that is, a group of nodes such that each of them is connected to at least k other. Since then, a number of techniques improving the original idea appeared, such as k-cliques [81] or k-dense method [90].

The Maximum Clique problem for a weighted graph searches for a clique S such that its sum of weights is maximal. An excellent review of the problem, its formulations, complexity, bounds, exact solutions, heuristics, and applications is carried out in Ref. [12]. The problem itself is one of the classic problems in computer science. It was among the 21 problems proved to be NP-complete by Karp in his famous paper [58].

A heuristic returning approximate solutions of the Maximum Clique problem is the crucial part of an algorithm proposed in Section 4.3.

2.6 Genetic Algorithms

Genetic Algorithm (GA) is an optimization technique inspired by the natural selection process [55, 49]. The approach has been successfully used in various settings, including: phylogeny [62, 54], drug design [108], and protein folding [100].

The goal of GA is to find a global minimum (or a sufficiently deep local minimum) of a function $f : X \to \mathbb{R}_+$. There are numerous flavors of GA, and hence it is more suitable to call it a "technique" or "framework", rather than an algorithm *per se*. For the sake of brevity, we shall present a single algorithm to illustrate the idea behind GA, which is very similar to the "Classical Simple Genetic Algorithm" as described by Vose [101]. The reader should bear in mind that virtually any element of the described algorithm may be replaced by a more complex counterpart. For an enhanced treatment of GA, see Refs. [49, 69].

In its very simple form, a genetic algorithm requires two problem-specific inputs: a bijective encoding $\epsilon : X \to \{0,1\}^M$ and a fitness function $\phi : \{0,1\}^M \to \mathbb{R}_+$, satisfying $\phi(\epsilon(\cdot)) \equiv f(\cdot)$. The algorithm may be summarized as follows:

1. Draw N codes $e \in \{0,1\}^M$ uniformly at random, and store them in G^1. This is the first generation of candidates.

2. Given a generation G^i, randomly choose N pairs of candidates (a candidate may be chosen more than once) in such a way that during each random draw, the j-th candidate (G^i_j) is chosen with probability:

$$\frac{\phi(G^i_j)}{\sum_{k=1..N} \phi(G^i_k)}$$

   This formula gives preference to the candidates that yield higher values of ϕ. Such a selection process is called the *roulette-wheel selection*. The chosen pairs of candidates will be the parents of the next-generation candidates.

3. For each chosen candidate, for each bit in its code, flip the bit of with (a very low) probability p. This step is called *mutation*.

4. For each pair of parent candidates (g', g''), draw a number $k \in [1, M-1]$ uniformly at random. Produce a child by so-called *crossover* operation in the following way: concatenate the first k bits of the first parent with the $M-k$ last bits of the second parent. This step creates N new candidates which constitute the next generation (G^{i+1}).

5. Find the candidate in G^{i+1} with the best fitness. If the value of the best fitness has not changed in S iterations: stop. Otherwise, go back to step 2.

The algorithm proposed in Section 4.5 is a variation of GA.

2.7 Molecular Dynamics and Visualization

The problem of aesthetic drawing of large graphs has been thoroughly studied since at least 1980s. One of the first feasible solutions was proposed by Eades [32], and refined by Fruchterman and Reingold [45] and Kamada and Kawai [57].

A large portion of currently employed methods, as well as the classic ones cited above, share a common idea: define a potential function assessing the quality of vertex arrangements, and find a sufficiently good local minimum of the potential. The potential is a balance of harmonic and repulsive terms.

The algorithm for node placement presented in Section 4.4 also belongs to this category.

2.8 Wikipedia in Academic Studies

In the recent years, Wikipedia has been increasingly a subject of scientific study, both qualitative and quantitative. On the qualitative side, its collaborative nature is of interest to sociologists [17, 51, 92], the way it revolutionizes access to knowledge attracts epistemologists [97, 109, 91, 65], the contributors' motivations have been studied by psychologists [8, 80]. Qualitatively, its content serves as an excellent example of a large, complex network [111]. The exponential growth of the number of contributors and text content [3] has been described in terms of the preferential attachment mechanism [20], the dynamics of user contributions (social interactions, conflict patterns) have been thoroughly measured and visualized [96, 16, 79], several linguistic corpora, taxonomies and ontologies have been extracted based on the contents of individual Wikipedia language editions[3], a number of bilingual dictionaries have been built based on the interlanguage links [35, 99], and methods of content enrichment have been proposed [1]. The above examples are by no means an exhaustive list. Wikipedia maintains a handy list of scholarly articles, theses, and books in which Wikipedia is studied[4].

In this thesis, we investigate yet another interesting aspect of the network in question: the topology of so-called interlanguage links. As far as we know, there were no earlier studies of this structure, quite possibly due to a false premise that its topology is trivial.

2.9 Translation as an Equivalence Relation

The question whether the approaches presented in this thesis may be applied to multilingual dictionary-like resources (such as the interlanguage links in Wikipedia) appears to be a facet of a profound problem in linguistics. The limits of a translation process, and properties of a hypothetical "largest common denominator" of all the natural languages have been long studied by the linguists.

Let us start with a couple of examples of linguistic findings that outline the limits of a translation process and at the same time demonstrate the extraordinary diversity of spoken languages. The only colors that exist in the vocabulary of every language are black, white and (with certain disclaimers) red [106]. In the Iquito language, spatial relationships are expressed relative to the river ("upriver", "downriver", "away from the river"), rather than relative to the position of the sun ("east" or "south") [33]. The Piranã language has no numerals, there are only expressions "small quantity" and "large quantity" [39, 42].

[3]See the list of papers accepted to the Language Resources and Evaluation Conference 2008, available here: `http://www.lrec-conf.org/lrec2008/List-of-accepted-papers.html`

[4]See: `http://en.wikipedia.org/wiki/WP:ACST`

On the bright side, Wierzbicka identifies the basic concepts common to all languages and defines the more sophisticated ideas using the basic vocabulary [106, 105, 107].

Moving on to the realm of multilingual online dictionaries and wikis, a project called OmegaWiki[5] is an attempt to create a multilingual dictionary. Much care is given to precision: OmegaWiki distinguishes between a language-neutral meaning ("defined meaning") and its manifestation in particular languages ("expression"). Much care is given to minimizing semantic drift, and there are a number of relations between defined meanings, such as: "narrower term" or "broader term".

In contrast to OmegaWiki, the interlanguage links in Wikipedia are clearly a secondary functionality. There are, however, a number of bots constantly checking the interlanguage links, in particular maintaining symmetry and transitivity of the links. There are also analysis tools providing users and bots with useful hints regarding the interlanguage links. For example, Interwiki Link Checker[6] compiles a list of pairs of articles in different languages with the same titles — potential candidates for establishing an interlanguage link. Interwiki Conflict Resolver[7] compiles a list of articles accessible from a given initial one, and presents it to user, sorted by languages. User can group the articles into subjects by assigning group numbers to articles. Let us note that there is an implicit assumption in all the tools: one that the interlanguage links should form an equivalence relation.

[5]See: `http://www.omegawiki.org/`

[6]See: `http://de.wikipedia.org/wiki/Benutzer:Flacus/Wikipedia_Interwiki-Link-Checker/en`

[7]See: `http://en.wikipedia.org/wiki/User:Yurik/Interwiki_Conflict_Resolver`

Chapter 3

Case Study: Semantic Drift in Wikipedia

In this chapter we shall describe the topology of so-called interlanguage links in Wikipedia. This network is as an excellent illustration of the class of networks researched in this thesis. We shall begin by defining the researched networks, and then we shall describe various properties of the networks: starting with the basic ones, and moving on to distributions of component sizes, node degrees and clustering coefficients. Next, we shall present a method of extracting the "skeleton" of the researched networks and analyze its properties. Finally, we shall summarize our findings.

3.1 Research Material

Let us begin by defining the research material, that is, the networks of interlanguage links in Wikipedia that are be investigated in this chapter.

3.1.1 Two Networks

It is convenient to describe the properties of the network of interlanguage links in graph-theoretic terms. For that reason, let us introduce two undirected graphs, $\mathcal{A}$ and $\mathcal{C}$, that will be analyzed throughout this chapter[1]. Vertices of one of the graphs represent articles, while vertices of the other represent categories. In each case, a vertex also represents all the redirects to the represented page. Two vertices v_1, v_2 are connected by an edge iff there exists an interlanguage link $v_1 \to v_2$ or $v_2 \to v_1$ (possibly leading to a relevant redirect). Each vertex has a color assigned to it: it is the language

[1]We consider the following pairs of terms to be synonyms: "graph"–"network", "vertex"–"node", "edge"–"link". Also, "node" will be used interchangeably with either "article" or "category" (depending on the context).

of the article or category representing it. Due to the nature of the interlanguage links, it is guaranteed that no two vertices having the same language assigned are connected by an edge. Throughout the rest of the chapter, $\mathcal{A}$ will denote the article network, and $\mathcal{C}$ – the category network.

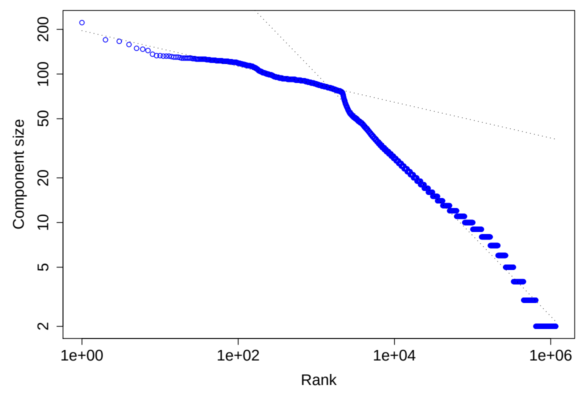
While building the two graphs, we have processed a total of 11 510 142 articles, 8 057 367 article redirects, and 89 339 694 article interlanguage links, as well as 1 724 088 categories, 5 630 category redirects and 13 902 852 category interlanguage links from all the 262 language editions of Wikipedia (cf. Table 1.1). This initial phase required approx. 10 GB of RAM. Next, all the connected components of the two graphs have been identified and each component has been stored separately. This way, all the subsequent calculations could be done sequentially, component-by-component, with a relatively small memory footprint.

3.1.2 Limitations

It should be kept in mind that a number of technical limitations might have influenced the results presented further in this chapter. Firstly, redirects to article sections are treated as redirects to the articles themselves. Also, in the case of redirects to pages representing a broader meaning, the relevant sections often do not exist, and the redirects point to the whole page instead. Thus, a redirect usually gives no hint whether the relation between the connected meanings is "equivalence" or "generalization". In consequence, interlanguage links leading to "generalizing" redirects represent "generalization" instead of intended "equivalence". For example, the combination: "en:Mother-in-law" $\xrightarrow{I}$ "ru:Tëshcha" $\xrightarrow{R}$ "ru:Rodstvo" $\xrightarrow{I}$ "en:Kinship" is reflected in the constructed graph as: "en:Mother-in-law" – "ru:Rodstvo" – "en:Kinship" ($\xrightarrow{R}$ denotes a redirect and $\xrightarrow{I}$ an interlanguage link; actual Russian titles were transliterated according to the BGN standard).

Secondly, the database snapshots provided by MediaWiki are not taken at the same instant: each language edition is dumped individually at a different time. There is over a month of difference between the timestamps of the earliest and the latest dump. That way, some coherent modifications spanning multiple language editions may seem incoherent when analyzing the dumps together.

Finally, some extremely rare cases – technically possible, but discouraged by the Wikipedia policies – are simply ignored. These include: interlanguage links to non-existent pages, more than one interlanguage link from a given page to a given language edition, circular redirects, interlanguage links from a given page to another page in the same language edition.



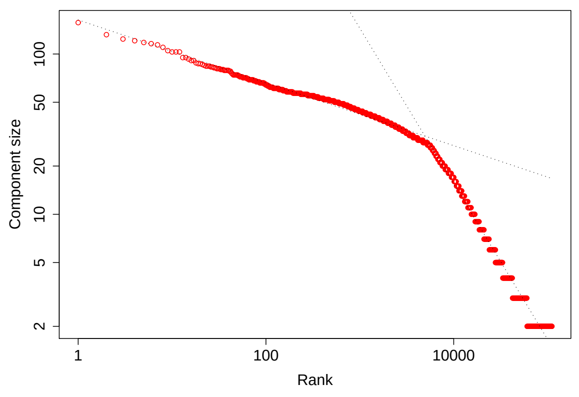


Figure 3.1: Component sizes of coherent components in the article network $\mathcal{A}$ (left) and the category network $\mathcal{C}$ (right), plotted against their ranks. Both plots have log-log scales. The influence of mass-produced date-related topics on the shapes of the distributions is discussed in the text.

3.2 Basic Properties

Let us proceed to study the properties of two networks of interlanguage links: one connecting the articles ($\mathcal{A}$), and the other connecting the categories ($\mathcal{C}$). In both cases we will treat the networks as undirected graphs, assuming a link $a - b$ iff there is an interlanguage link $a \longrightarrow b$ or $b \longleftarrow a$.

Network $\mathcal{A}$ consists of 11 510 142 nodes and 89 339 694 links. Approx. 42% of the nodes are isolated, and the remaining nodes are grouped into 1 223 183 connected components. Network $\mathcal{C}$ consists of 1 724 088 nodes and 13 902 852 links, approx. 51.5% of the nodes are isolated, and the rest are grouped into 118 039 connected components.

We will say that a connected component is *coherent* when no two pages are in the same language[2], and that it is *complete* when it contains all the possible links (i.e., is a clique). There are 59 323 incoherent components in $\mathcal{A}$ and 6 152 incoherent components in $\mathcal{C}$. In both cases it is approx. 5% of all the non-singleton connected components. Completeness is correlated with coherence: for example in the case of $\mathcal{A}$, 63% of the coherent components are complete, and 99% contain at least half of all the possible links. On the other hand, none of the incoherent components are complete, and only about 61% contain at least half of all the possible links.

[2]We use the terms "node" and "page" interchangeably. Both are equivalent to "article" in the context of network $\mathcal{A}$, and to "category" in the context of network $\mathcal{C}$.

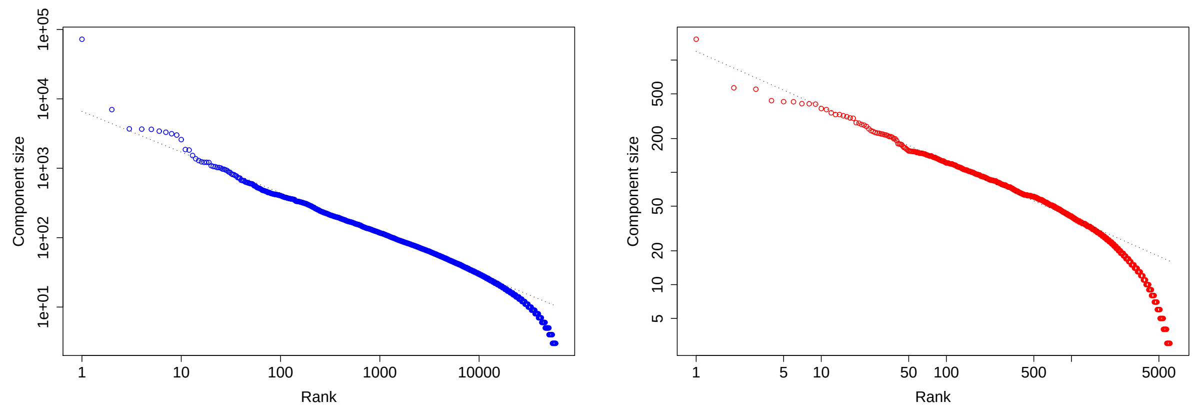


Figure 3.2: Component sizes of the incoherent components in the article network $\mathcal{A}$ (left) and the category network $\mathcal{C}$ (right), plotted against their ranks. Both plots have log-log scales. The so-called "king effect" [61] is discussed in the text.

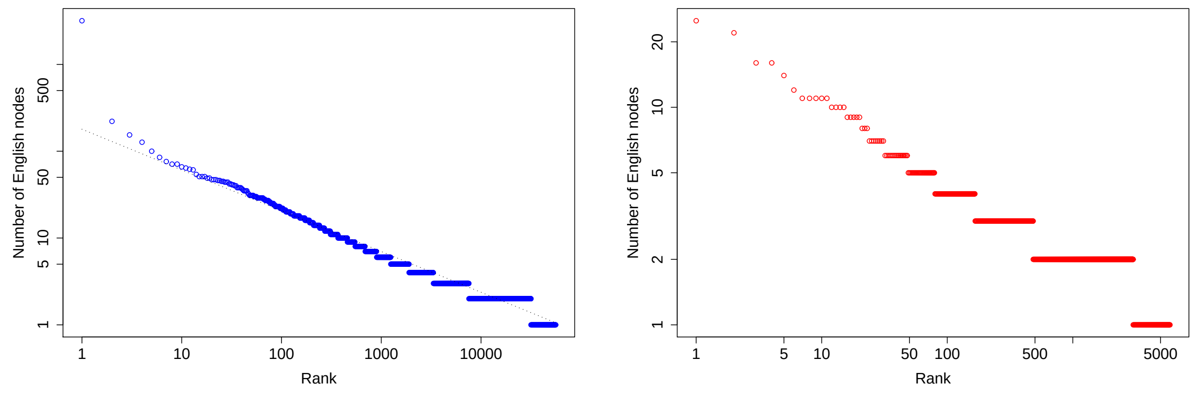


Figure 3.3: Number of English pages in the incoherent components in the article network $\mathcal{A}$ (left) and the category network $\mathcal{C}$ (right), plotted against their ranks. Both plots have log-log scales.

3.3 Component Sizes

Three component size vs. rank plots for $\mathcal{A}$ and $\mathcal{C}$ are presented. Sizes of the coherent components are plotted in Figure 3.1 and sizes of the incoherent components are plotted in Figure 3.2. For the incoherent components of each network, the number of English pages in the case of incoherent components (a coarse measure of the number of topics mixed in a component) is plotted in Figure 3.3. In each of the plots, both scales are logarithmic.

The plotted points are (piecewise) well approximated by straight lines, which indicates that a component's size s is a power-law function of its rank r, namely: $s \sim r^{-\gamma}$.

Let us take a closer look at the obtained distributions. In the case of the coherent components of $\mathcal{A}$ (left panel of Figure 3.1) there are two clear regimes. Most of the top 2 200 or so components (the first regime), each covered by at least 75 language editions, contain articles on the years of the current era and centuries. Such articles are easy to create in an automated way, and it is easy to maintain the interlanguage links to corresponding articles in the other language editions (easy maintenance explains why the components are coherent). An informal competition among the language editions for the largest number of articles might be an additional motivation for the mass-creation of the date-related pages.

Similarly, two regimes in the component size distribution of $\mathcal{C}$ (right panel of Figure 3.1) can also be observed, although the transition between them is smoother than in the previous case. Date-related topics account for about 75% of the top 5 000-6 000 components (each containing at least 26-28 nodes). Among these are categories for years, decades, centuries, births and deaths in a given year, and (for the recent times) films and video games in a given year. Other prominent categories are: countries (including "History of ..." and "Geography of ..." as separate categories), and users speaking a given language on a given level.

Moving on to the sizes of incoherent components, we note that the largest in $\mathcal{A}$ (left panel in Figure 3.2) is well above the best-fit line ($\gamma \approx 0.587$). This anomaly is an example of the so called "king effect", discovered by Laherrère [61] while analyzing the sizes of the world's oilfields: the largest element is much larger than a log-log regression would predict. The same component is a clear outlier in the distribution of the number of English articles in components (left panel in Figure 3.3)[3].

Figure 3.2 and Table 3.1 show that semantic drift is a serious problem both in $\mathcal{A}$ and $\mathcal{C}$. To give a perspective: the largest component consists of 72 284 articles, including 3 184 articles in English, while in theory each component should contain at most one article in English, and its size should be bounded by the number of language editions, i.e., about 250. It contains articles on such a diverse subjects as: "Abelian group",

[3]It is not clear whether the largest component is about to become the so-called "giant component" or its larger-than-expected size is a mere fluctuation. The answer depends on the choice of a model governing the network's growth, which is outside the scope of this analysis.

	Range		Fit results	
	size	ranks	γ	R^2
Articles (C)	$[75, \infty)$	$(-\infty, 2\,173]$	0.120	0.9852
Articles (C)	$(-\infty, 75)$	$(2\,173, \infty)$	0.544	0.9664
Articles (I)	$[15, \infty)$	$(-\infty, 23\,463]$	0.587	0.9889
Articles (E)	$(-\infty, \infty)$	$(-\infty, \infty)$	0.469	0.9690
Categories (C)	$(-\infty, 27]$	$[5\,638, \infty)$	0.196	0.9811
Categories (C)	$(27, \infty)$	$(-\infty, 5\,638)$	0.970	0.9855
Categories (I)	$(-\infty, 20]$	$[2\,472, \infty)$	0.493	0.9897

Table 3.1: The power law applied to the component sizes of the article and category networks. "C", "I" and "E" refer to Figures 3.1, 3.2, and 3.3 (respectively). It is tested whether the relation between a component size s and its rank r is indeed $s \sim r^{-\gamma}$. The last column denotes adjusted R^2 – a measure of correlation.

"Beekeeping", "Chinese poetry", and "Districts of Luxembourg". The second largest component in terms of the total number of articles contains 7 004 nodes, while the second largest in terms of the number of English articles contains 221 nodes.

Table 3.1 presents the parameters of the best fits corresponding to the lines in Figures 3.1, 3.2, and 3.3.

3.4 Node Degrees and Clustering Coefficients

The left panel of Figure 3.4 presents the node degree distribution in both researched networks (note that both axes are logarithmic). The median node degree is 6 in $\mathcal{A}$ and 12 in $\mathcal{C}$. There are two anomalies in the distribution in the case of the article network: a plateau spanning degrees 74-93, and a peak at degrees 117-119. Both phenomena have plausible explanations. The plateau is a result of articles where the subject is on years of the current era. Some editions contain articles on all the 2000+ years, others only on the more recent years. For example, there are approx. 88 language editions covering year 1709, 82 covering year 1209 and 74 covering year 509. The articles on a given year are usually forming a clique, thus each article has degree equal to one less than the size of the clique. As a consequence, we observe that an increased number of nodes with degrees 74-93 relates to a high number of cliques of sizes 75-94. On the other hand, the peak at degrees 117-119, relates to articles on days of the year. There are approximately 120 language editions where such articles are present, and these editions usually contain articles on all the 366 days. Most of the groups of articles are connected in cliques, hence a peak in the degree distribution. The article with the highest degree (337) is "ca:Llista de personatges de la Mitologia Egípcia" which contains short descriptions of various gods of the Egyptian mythology. A number of pages on articles in other languages, including 42 from the English edition, contain

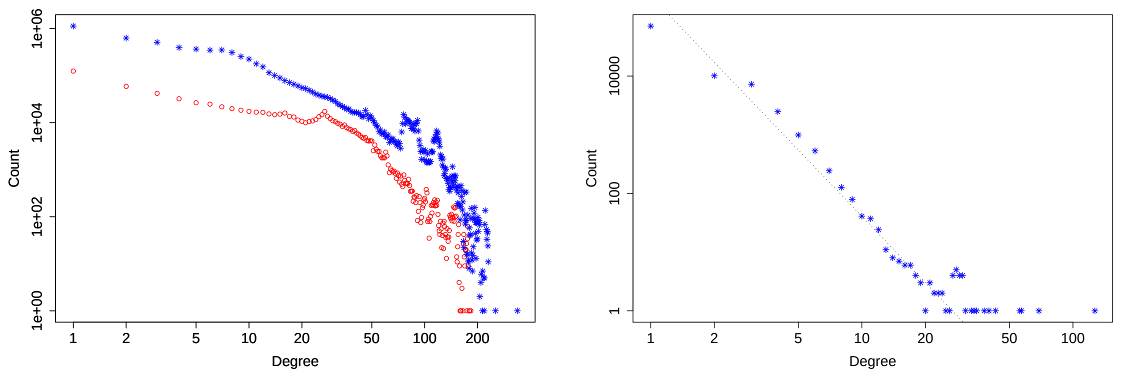


Figure 3.4: (Left) Degree distribution for the article network (blue stars) and category network (red circles). Log-log scale, degree 0 omitted. The plateau at 74-93 and the peak at 117-119 in the case of articles are commented on in the text. (Right) Degree distribution for the skeleton of the article network (log-log scale). The skeleton extraction procedure is described in the text. Only the incoherent components are accounted for, degree 0 is omitted. The distribution fits the power law with $\gamma \approx 3.75$ (adjusted $R^2 \approx 0.9770$). The peak at 27-30 is commented on in the text.

interlanguage links to (redirects to) this page.

Watts and Strogatz [104] have demonstrated the usefulness of an indicator named *clustering coefficient* in describing network topologies. The clustering coefficient of a perfectly coherent and complete network of interlanguage links would be 1. In reality, for both networks the value is quite high (approx. 0.97), with over 0.98 for the coherent components, and approx. 0.91 for the incoherent ones. Figure 3.5 presents the distribution of the values of the clustering coefficient for nodes having at least 10 neighbors[4]. As expected, a low clustering coefficient is fairly uncommon. More interesting is the conditional probability that a node with degree > 9 and clustering coefficient < 0.80 will be part of an incoherent component: it is approximately equal to 0.9958 in the case of $\mathcal{A}$ and "only" 0.7748 in the case of $\mathcal{C}$.

3.5 Skeleton Extraction

Let us summarize the results so far: having analyzed the distributions of component sizes, degrees, and clustering coefficients, we have found that the network of interlanguage links mainly consists of tight clusters. There are rare connections between the clusters, which are usually symptoms of incoherence. Our next question is: what is the topology of these rare connections?

[4]If the lower degrees were included, peaks at $\frac{1}{2}, \frac{1}{3}, \frac{2}{3}, \ldots$ would be visible, since the low-degree nodes have only a few possible values of the clustering coefficient. We have decided to "subtract" the expected peaks and show the less-obvious pattern.

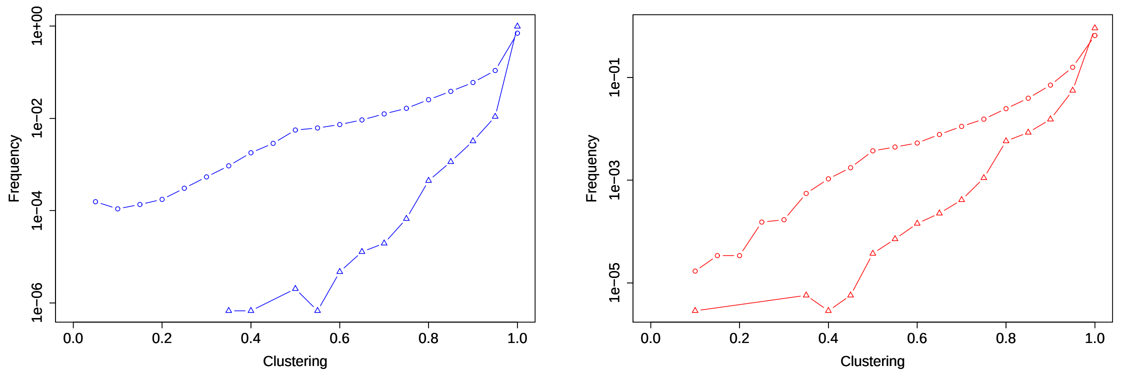


Figure 3.5: Distribution of the clustering coefficient values of nodes. Only the nodes with degree $>$ 9 have been accounted for. The left diagram presents distributions for the article network, the right diagram presents distributions for the category network. Circles denote values for the incoherent components, triangles – for the coherent ones. Note that the y-axis is logarithmic, so the vast majority of nodes have clustering coefficient close to one.

We would like to extract the "skeleton" of an incoherent component, a network in which each set of nodes representing a given topic (usually a tight cluster) is shrunk to a single point, thus revealing the connections between separate topics. Of course, partitioning a network into topics requires expert knowledge, which we cannot provide. Instead, we propose a method of extracting an approximate structure of the skeleton network, presented in Algorithm 3.1.

The algorithm takes a graph $G = \langle V, E, \kappa \rangle$, where V represents vertices (i.e., pages), E represents edges (i.e., interlanguage links), and function $\kappa(\cdot)$ represents nodes' languages. The algorithm consists of a couple of phases, each of which has a clear objective:

1. First, any of the most frequently occurring languages is chosen as the reference language. The nodes in this language will be the reference nodes.

2. For each node v, the closest reference node(s) $z(v)$ is determined. Let us bear in mind that the number of the closest reference nodes of a node v, denoted by $|z(v)|$, may be larger than 1.

3. In the loop starting in line 7, nodes with only one closest reference nodes are "collapsed" to the reference node. All the links of a collapsed node are "inherited" by the respective reference node.

4. Next, we want to collapse the remaining non-reference nodes. In the loop starting in line 15, as long as there is any pair of non-reference nodes connected by a link, we merge the pair into a single (non-reference) node.

Algorithm 3.1 FindSkeleton(G) – extraction of the skeleton of a graph

```
1:  $l_{max} \leftarrow \arg\max_x \left|\{v \in V | \kappa(v) = x\}\right|$    ▷ choose the most frequent language
2:  $L \leftarrow \{v \in V | \kappa(v) = l_{max}\}$    ▷ nodes in the most frequent language
3:  for $v \in V$ do
4:      $d_{min}(v) \leftarrow \min_{w \in L} d(v, w)$    ▷ distance to the closest node in L
5:      $z(v) \leftarrow \{w \in L | d(v, w) = d_{min}(v)\}$    ▷ the closest nodes in L
6:  end for
7:  for $\overline{v} \in L$ do
8:      for $v \in V$ such that $z(v) = \{\overline{v}\}$ do
9:          for $w \in \epsilon(v)$ do
10:             $E \leftarrow E \cup \{\{\overline{v}, w\}\} \setminus \{\{v, w\}\}$
11:         end for
12:         $V \leftarrow V \setminus \{v\}$    ▷ v removed, its links "acquired" by $\overline{v}$
13:     end for
14: end for
15: for $\{v', v''\} \in E$ such that $|z(v')| > 1 \wedge |z(v'')| > 1$ do
16:     $V \leftarrow V \cup \{w\}$    ▷ merge $v'$ and $v''$ into a new node $w$
17:     $\kappa(w) \leftarrow \emptyset$
18:     $z(w) \leftarrow z(v') \cup z(v'')$
19:     for $v''' \in \epsilon(v') \cup \epsilon(v'') \setminus \{v', v''\}$ do
20:         $E \leftarrow E \cup \{\{w, v'''\}\}$
21:     end for
22:     $V \leftarrow V \setminus \{v', v''\}$
23:     for $v''' \in \epsilon(v') \cup \epsilon(v'')$ do
24:         $E \leftarrow E \setminus \{\{v', v'''\}, \{v'', v'''\}\}$
25:     end for
26: end for
27: for $v \in V$ such that $|z(v)| = 2 \wedge |\epsilon(v)| = 2$ do
28:     $E \leftarrow E \cup \{\epsilon(v)\}$
29:     $V \leftarrow V \setminus \{v\}$
30: end for
31: return $\langle V, E, \kappa \rangle$
```

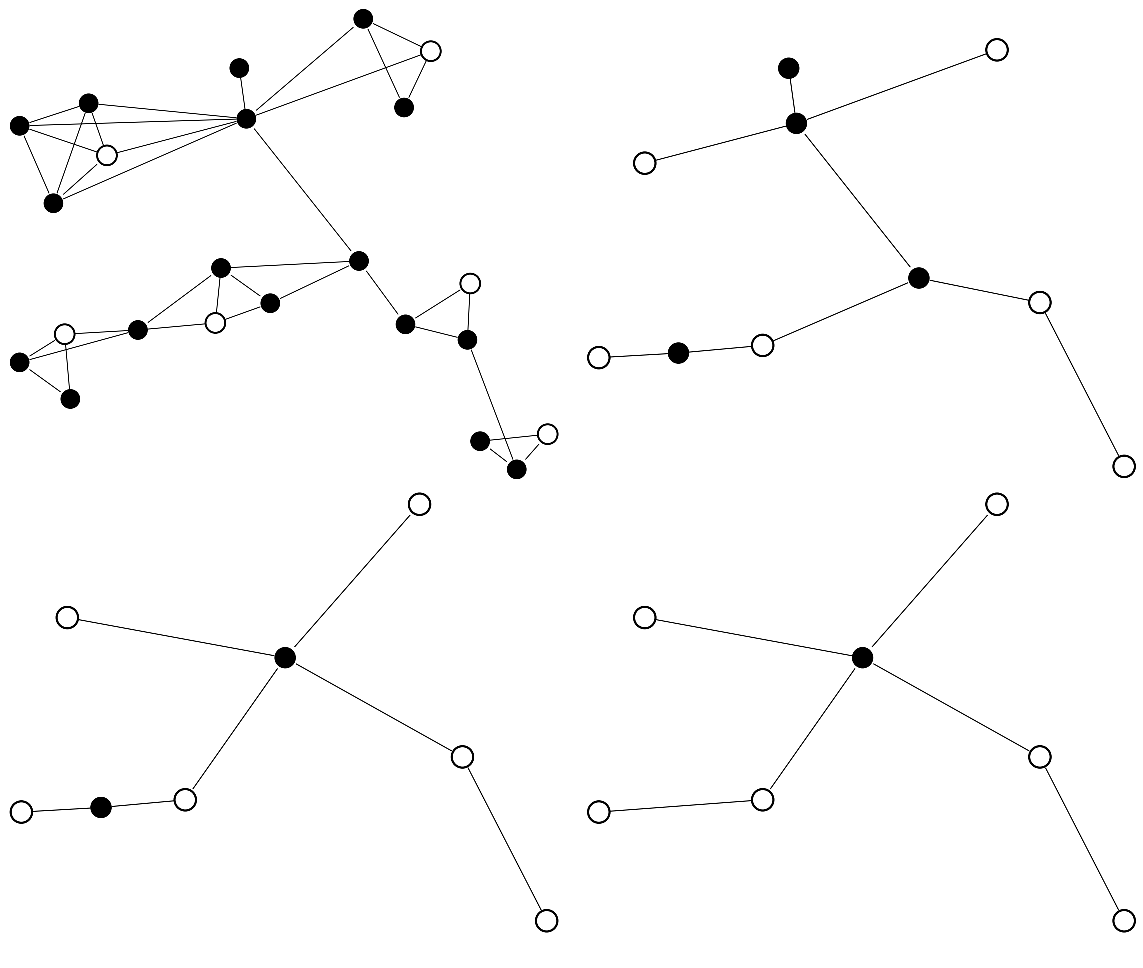


Figure 3.6: Illustration of the skeleton extraction procedure. Top-left: the original network, reference nodes are hollow. Top-right: the nodes with exactly one closest reference node are merged (loop in line 7 in Algorithm 3.1). Bottom-left: the nodes with more than one closest reference node are merged (loop in line 15 in Algorithm 3.1). Bottom-right: the non-reference nodes with exactly two neighbors are removed (loop in line 27 in Algorithm 3.1).

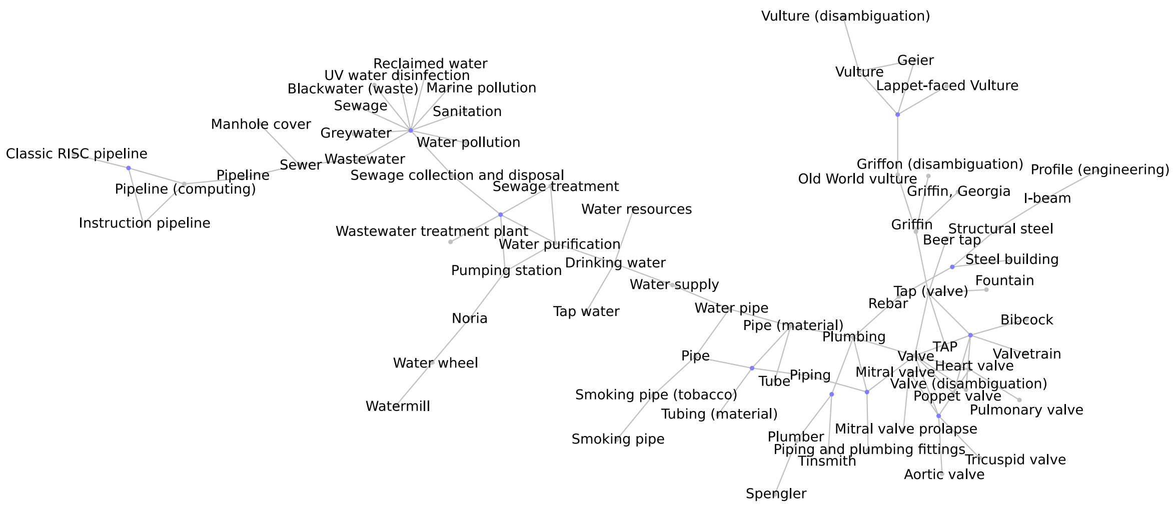


Figure 3.7: Skeleton of a medium-sized incoherent component (812 articles, including 47 in English). The skeleton extraction procedure is described in the text.

5. At this point any non-reference node left may only have the reference nodes as its neighbors. We remove all the non-reference nodes that have exactly two neighbors, and in each case we directly connect the pair of reference nodes (loop starting in line 27).

In other words, the algorithm first collapses all the nodes that are in the sphere of influence of only one reference node ($|z(v)| = 1$). After this operation the network contains only the reference nodes and some "intermediate" nodes. We wish to minimize the number of the "intermediate" nodes. Therefore, the algorithm collapses all the connected "intermediate" nodes, and ultimately removes nodes the "intermediate" nodes that connect only two reference nodes. Figure 3.6 presents an example of the skeleton extraction procedure.

Figure 3.7 presents a result of the skeleton extraction procedure applied to a middle-sized component. After extracting the skeleton of the entire $\mathcal{A}$ network, the average degree of a skeleton node is approx. 1.17, and the clustering coefficient is approx. 0.37. The distribution of node degrees is shown in the right panel of Figure 3.4. The distribution is power-law (with $\gamma \approx 3.75$), indicating a scale-free network. The peak at degrees 27-30 is yet another result of mass-edition, this time related to articles on the days of the year (such as: *en:December 1*, *en:December 2*, etc.) In 10 out of 12 cases, at least one language edition contains a bizarre copy-and-paste error that connects all the days of a given month, for example all the 30 articles on the days of September from the Hindu edition contain an interlanguage link to the article in Kannada on September 11. Thus, in the skeleton network, the node representing September 11 has 29 neighbors. Note that in the ideal case (no incoherence) the skeleton network should consist solely of isolated nodes, i.e., should contain no links at all.

3.6 Summary

Summing up, we have presented the surprisingly complex topology of the interlanguage links in Wikipedia. Instead of a set of isolated cliques, the structure can be informally described as a scale-free network of loosely interconnected cores. From a user's point of view, lack of coherence results in semantic drift, e.g. *en:Pipeline* and *en:Vulture* are connected by a series of interlanguage links which are supposed to model equivalence (cf. Figure 3.7).

Chapter 4

Theoretical Results

This chapter presents the main theoretical results of author's original research. Its contents are laid out as follows: Section 1.2 introduces the notation and definitions used throughout the chapter, illustrating them with examples. Section 1.3 presents the algorithmic problem of drift detection in a vertex-colored graph, together with a brief motivation behind it. Then the problem's computational complexity is assessed in Section 4.1.

As the problem is proved to be computationally hard, three approximate solutions are presented, each using a distinct approach. In Section 4.2 we describe a meta-approach that is a common part of two other approaches: a greedy algorithm for finding the minimal cut given a sequence of edges ordered decreasingly by their "credibility". In Section 4.3, we iteratively solve the Maximum Clique graph-theoretic problem (using fast heuristics). In Section 4.4 we present an approach motivated by molecular mechanics, in which a carefully designed potential is minimized. In Section 4.5 a genetic algorithm yielding an approximate solution to the computational problem is presented. In Section 4.6, we approach the problem by applying the Girvan-Newman algorithm for finding communities in a network based on interactive calculation of edge betweenness.

4.1 Computational Complexity of the Researched Problem

Computational complexity theory is the part of computer science that studies the resource requirements in time and memory of various computational tasks [83]. Full presentation of the theory is outside the scope of this work, readers unfamiliar with the theory may consult Refs. [47, 102].

In the following section, we prove that the Minimum Partition optimization problem is NP-hard, i.e., for any problem in the **NP** complexity class, there exists a *polynomial-*

time reduction to our problem. This implies, in particular, that if there is a polynomial-time solution to our problem, then any problem in the **NP** class has a polynomial time solution.

In practical terms, the result has the following significance: since a polynomial-time solution to the MINIMUM PARTITION problem would answer the most important and arguably the most difficult open question in the field of computer science ($\mathbf{P} \overset{?}{=} \mathbf{NP}$), then finding such a solution is extremely unlikely. Therefore, we may focus our attention on looking for fast *heuristics*, returning approximate solutions to the problem.

We express our main result in the form of a theorem. The reminder of the section shall be devoted to the proof of the theorem.

Theorem 4.1 MINIMUM PARTITION *optimization problem, even with the restriction of edge weights to* $\mu \equiv 1$, *is NP-hard.*

Proof (4.1) In order to prove NP-hardness of the optimization problem, we show that the corresponding decision problem is NP-complete:

Instance:	A graph $G = \langle V, E, \kappa \rangle$, and a bound B.
Question:	Is there a partition Π such that $\|\Pi\|_G \leq B$?

Let us start the proof by noting that the decision problem is NP. Indeed, given an instance of Π, the calculation of $\|\Pi\|_G$ is polynomial. NP-completeness is shown by a polynomial transformation from the MULTITERMINAL CUT problem for arbitrary graphs, which was proved to be NP-hard in [25]:

Problem 4.2 (Multiterminal Cut, [25]) *Given a graph* $G = (V, E)$, *a set* $S = \{s_1, s_2, \ldots, s_k\}$ *of* k *specified vertices or terminals, and a positive weight* $w(e)$ *for each edge* $e \in E$, *find a minimum weight set of edges* $E' \subseteq E$ *such that the removal of* E' *from* E *disconnects each terminal from all the others.*

Theorem 4.3 (NP-completeness of 3-Terminal Cut, [25]) *If arbitrary*[1] *graphs are allowed,* MULTITERMINAL CUT *for* $k = 3$ *(i.e.,* 3-TERMINAL CUT*) is NP-complete even if all weights are equal to* 1.

A detailed proof of Theorem 4.3 can be found in [25]. The proof uses a polynomial transformation from the SIMPLE MAX CUT problem [46], [47].

Let us consider the decision problem corresponding to the 3-TERMINAL CUT problem with unit edges:

[1]In this context, arbitrary means: "not necessarily planar". This remark in the original formulation is due to the fact that in the preceding parts of [25] the tractability of some special cases of planar graphs is investigated.

Instance:	An unweighted graph $G = (V, E)$, a set of three terminal nodes S, and a bound B.
Question:	Is it possible to remove at most B edges in such a way that each terminal would be disconnected from all the others?

Having chosen a problem known to be NP-complete, we proceed by showing a polynomial transformation from the selected problem to the decision version of the Minimum Partition problem.

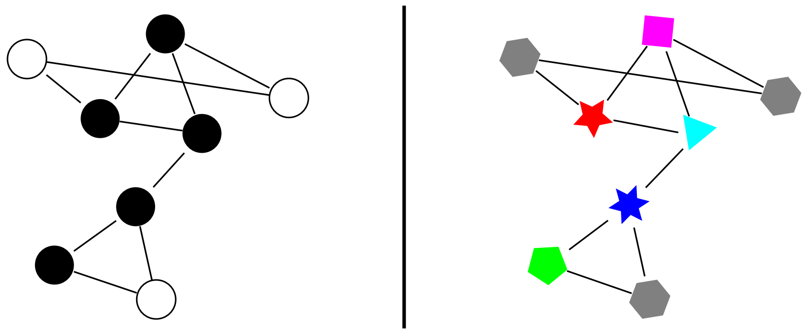


Figure 4.1: Sample transformation from 3-Terminal Cut *to* Minimum Partition. *Input for* 3-Terminal Cut*, shown on the left, is a graph G with three terminal nodes marked with hollow circles. The corresponding input for* Minimum Partition*, shown on the right, is a graph G' with vertices in six different colors. All the nodes in G' corresponding to the terminals in G are of the same color (gray hexagon) and each of the remaining nodes has a different, distinct color.*

Let us show how to construct a graph G' being the input for the Minimum Partition, based on an instance $G = (V, E)$ of the 3-Terminal Cut. The graph G' shall contain the same vertices V and edges E as graph G, and the vertex coloring shall be as follows:

$$\kappa(v) := \begin{cases} S & \text{for } v \in S \\ \{v\} & \text{otherwise} \end{cases} \tag{4.1}$$

An example of the transformation is shown in Figure 4.1. The reverse transformation, that is, mapping of the answer to the Minimum Partition problem to the setting of the 3-Terminal Cut problem is as follows: given Π, return $I_G(\Pi)$, which is a polynomial operation.

Lemma 4.4 *For $G = (V, E)$ and a set of terminals S, there exists a disconnecting subset E' such that $|E'| \leq B$* ***if and only if*** *for a corresponding G' there exists a domain partition Π such that $\|\Pi\|_{G'} \leq B$.*

Proof (4.4, ⇒) Assume that E' disconnects all terminals S in graph G from each other, and $|E'| \leq B$. Since S is the only non-trivial layer of G', and E' disconnects all elements in S from each other, then E' is a cut, and therefore for Π, the natural partition of $\langle V, E \setminus E', \kappa \rangle$, the inequality $\|\Pi\|_{G'} \leq B$ holds.

$\square$

Proof (4.4, $\Leftarrow$) Given a partition Π such that $\|\Pi\| \leq B$, from the choice of colors (equation 4.1), and from the definition of a partition (Eq. 1.7) it follows that:

$$\forall_{s_1 \in S} \forall_{s_2 \in S} (s_1 \neq s_2) \Rightarrow \Pi(s_1) \neq \Pi(s_2) \tag{4.2}$$

Therefore, removing edges $I_{G'}(\Pi)$ from graph G separates each terminal node in S from all the other ones. The total number of edges which need to be removed in order to achieve separation is thus at most: B.

$\square$

We have demonstrated that the 3-Terminal Cut decision problem can be reduced to Minimal Partition decision problem using transformations of polynomial complexity (in time and in memory). Therefore, the latter problem is at least as difficult as the former one. Since the former is NP-complete, and the latter is NP, then the Minimum Partition decision problem is NP-complete, and thus the corresponding optimization problem in NP-hard.

$\square$

4.2 "Greedy" Approach

Our first algorithm belongs to a class of so-called greedy algorithms. It is a well-known and simple problem solving strategy, which is based on an assumption that a series of locally optimal choices will yield a globally optimal solution. The set systems such that the corresponding optimization problem can be solved by a greedy algorithm are called greedoids [60]. Even in the case of optimization problems that cannot be solved by a greedy algorithm, such a strategy may yield quite useful approximate solutions.

Let us assume that we are given a list of edges of the graph ordered descending by their "trustworthiness" (denoted by $\vec{E}$). Next, starting with an empty set of accepted edges, we process each edge from the ordered list in the following way: if the edge can be added to the set of accepted edges without violating coherence, then we add it. The corresponding set of rejected edges is returned as an approximate solution to the problem. Algorithm 4.1 implements the above procedure.

Observe that for any input $\vec{E}$ the algorithm returns a locally-minimal cut. Also, there is always an input sequence of edges $\vec{E}$ such that the algorithm 4.1 returns the

Algorithm 4.1 GreedyMerge$(G, \vec{E})$ – greedy meta-approach

```
 1: $F \leftarrow \emptyset$        ▷ rejected edges, i.e., cut
 2: for $v \in V$ do
 3:     $A[v] \leftarrow \{\kappa(v)\}$
 4:     $B[v] \leftarrow \{v\}$
 5: end for
 6: for $\{v_1, v_2\} \in \vec{E}$ do
 7:     if $A[v_1] \cap A[v_2] = \emptyset$ then
 8:         $A[v_1] \leftarrow A[v_2] \leftarrow A[v_1] \cup A[v_2]$
 9:         $B[v_1] \leftarrow B[v_2] \leftarrow B[v_1] \cup B[v_2]$
10:     else if $B[v_1] \neq B[v_2]$ then
11:         $F \leftarrow F \cup \{\{v_1, v_2\}\}$
12:     end if
13: end for
14: return $F$
```

(globally) minimal cut. In particular, any sequence that ends with the edges of the minimal cut will yield the minimal cut.

As noted, this is in fact a meta-algorithm: the missing part is a method of ordering the edges. The easiest way to complete the algorithm is to provide a random sequence of edges on input, as presented in Algorithm 4.2. However, we will describe two different approaches that produce the required sequence in Sections 4.4 and 4.6. The greedy strategy will also be employed in Section 4.5.

During an evaluation of the approaches developed in this chapter, Algorithm 4.2 shall serve as a baseline. It is arguably the simplest and the most obvious heuristic for the main computational problem, and all the other approaches shall be evaluated relative to the results of this algorithm.

Algorithm 4.2 GreedyCut(G) – simple greedy algorithm, baseline for further approaches

```
1: $\vec{E} \leftarrow$Shuffle$(E)$
2: $F \leftarrow$GreedyMerge$(G, \vec{E})$
3: return $F$
```

4.2.1 Computational Complexity

Each vertex of the meta-algorithm holds a set of at most K values, where K is the number of colors in the graph. Therefore, the space cost of the algorithm is $\mathcal{O}(|V|K)$. The main loop has $\mathcal{O}(|E|)$ iterations, the time cost of each iteration is limited by the set intersection and sum operations, each of which may be performed within time cost $\mathcal{O}(K)$. Therefore, the time cost of the meta-algorithm is $\mathcal{O}(|E|K)$.

4.3 "Cliques" Approach

Our second approach features another classic algorithmic problem: finding the maximum clique in a graph. As we shall show, well-known clique-finding heuristics can be employed to solve the Minimum Partition problem.

4.3.1 Motivation

As it was mentioned in Section 1.3, completeness is the desired property of the researched graphs. Therefore, in an ideal layered graph, each connected component is a clique. Recall that clique is defined as a set of vertices S such that $\forall_{v,w \in S}\, \{v, w\} \in E$. An example of a complete graph, consisting of three cliques, is presented in the left panel of Figure 4.2.

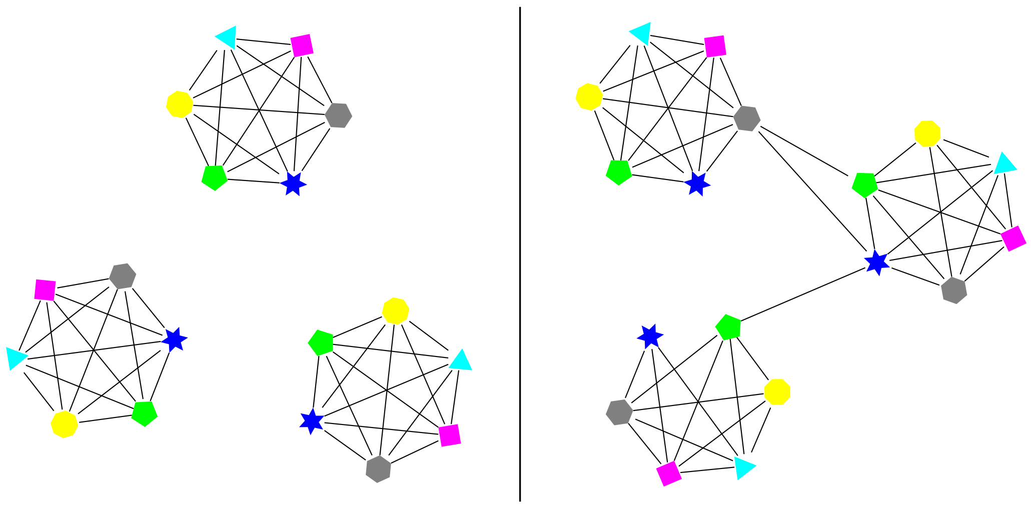


Figure 4.2: Example of a complete graph (left) and an incomplete graph (right). Adding an extra edge to the graph on the left would render the graph incoherent. Using only three edge removals, followed by three edge additions, the graph on the right may be transformed to the complete graph on the left.

The main idea of the approach is the following: given a problem instance which is "close" to the ideal case (measuring by the number of edges added or removed with respect to the complete graph), we may approach it by repetitively employing an algorithm for finding maximum clique in a weighted graph.

4.3.2 The Maximum Clique Heuristic

Algorithm 4.3 A simple heuristic for the MAXIMUM CLIQUE problem

```
1: $S \leftarrow \emptyset$
2: $P \leftarrow V$
3: $v \leftarrow \arg\max_{v' \in P} \sum_{v'' \in V} \mu(\{v', v''\})$
4: while $v \neq \emptyset$ do
5:     $S \leftarrow S \cup \{v\}$
6:     $P \leftarrow P \cap \{v' \in P \mid \{v, v'\} \in E\}$
7:     $v \leftarrow \arg\max_{v' \in P} \sum_{v'' \in V} \mu(\{v', v''\})$
8: end while
9: return $S$
```

We have selected an extremely simple heuristic for the MAXIMUM CLIQUE problem. Using the terminology introduced by Kopf and Ruhe [59], we employ a *best in* strategy for finding the best clique. Our algorithm starts from the empty set, and iteratively finds best vertices for addition, simultaneously updating the set of possible extensions. Algorithm 4.3 presents a pseudo-code of this approach.

4.3.3 The Algorithm

In short, our clique-based heuristic for MINIMUM PARTITION creates an initial partition, and then iteratively clusters the vertices using maximum clique heuristics[2]. The clustering is performed in two slightly differing passes. Algorithm 4.4 presents the general structure of the heuristic.

Algorithm 4.4 CLIQUESCUT(G) – general structure of the clique-based approach

```
1: $\Pi \leftarrow \bigcup_{v \in V} \{\{v\}\}$
2: $\Pi \leftarrow \textsc{FirstPass}(G, \Pi, \theta)$
3: $\Pi \leftarrow \textsc{SecondPass}(G, \Pi)$
4: return $I_G(\Pi)$
```

In the initial clustering, each vertex v is assigned to its own singleton cluster $\{v\}$. Written formally, the initial clustering is:

$$\Pi = \bigcup_{v \in V} \{\{v\}\} \tag{4.3}$$

Any two clusters $\{v_1\}$ and $\{v_2\}$ are linked iff the corresponding vertices v_1 and v_2 are linked too. We introduce a function $M : 2^V \times 2^V \rightarrow \mathbb{R}$ measuring the sum of

[2]One should keep in mind that iterative application of a maximum clique heuristic does not necessarily yield a clique. In other words, slightly informally: a clique of cliques of vertices is not necessarily a clique of vertices.

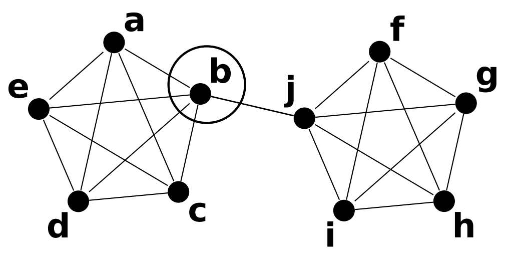


Figure 4.3: Example of an undesired start for finding maximum clique. Algorithm 4.3 will fail to find any of the cliques of size 5*, since it is going to choose either b or j as the initial vertex, expand by selecting the other one (j or b, respectively), and return* $\{b, j\}$ *as a clique.*

weights between any two given clusters:

$$M(C', C'') := \begin{cases} 0 & \text{when } \exists_{v' \in C'} \exists_{v'' \in C''} \, \kappa(v') = \kappa(v'') \\ \sum_{v' \in C'} \sum_{v'' \in C''} \mu(\{v', v''\}) & \text{otherwise} \end{cases} \tag{4.4}$$

Note that for the initial setting:

$$M(\{v_1\}, \{v_2\}) = \mu(v_1, v_2) \tag{4.5}$$

unless $\kappa(v_1) = \kappa(v_2)$. As the partition Π changes, so does the subdomain of M that we're interested in. In Subsection 4.3.4 we show that the function can be efficiently computed using extra $\mathcal{O}(|V| + |E|)$ space.

In the first pass, only cliques above a given threshold size θ are merged. The reason for introducing a threshold is illustrated in Figure 4.3. Assuming all edges in the presented graph have unit weights, a naïve heuristic, such as algorithm 4.3, is going to select either b or j as the initial vertex (line 3 in Algorithm 4.3), expand by selecting the other vertex (j or b, respectively), and return $\{b, j\}$ as a clique to merge.

One possible solution to the above problem would be application of a more robust (and slower) maximum clique heuristic. However, since we need to run the maximum clique search multiple times, we may as well use the original heuristic, and discard any unsatisfactory results.

Choosing to discard cliques of size less than θ, we need to keep track of the clusters which, once selected as the starting cluster, did not yield a satisfactory clique. Otherwise, the algorithm would enter an infinite loop: constantly choosing the same cluster as the starting one, and constantly discarding the resulting clique. To avoid this, a set of unsuccessful initial clusters is maintained (variable F in Algorithm 4.5).

Algorithm 4.5 presents the pseudo-code for the first pass of the "clique" approach.

When no more cliques of size θ or greater can be found, the first pass of the heuristic is concluded. What remains to be done is to merge all the remaining clusters, so that the resulting clustering Π does not contain any mergeable pair of clusters (note that a mergeable pair of clusters is a clique of size 2).

Algorithm 4.5 FirstPass(G, Π, θ) – first pass of the clique-based approach: clusterization of big cliques

```
1: $F \leftarrow \emptyset$        ▷ “failed” clusters
2: while $\exists_{c',c'' \in \Pi} M(c', c'') > 0$ do
3:   $S \leftarrow \emptyset$
4:   $P \leftarrow \Pi \setminus F$        ▷ possible extensions
5:   $s \leftarrow$ FindHeaviest$(\Pi, P)$
6:   $c \leftarrow s$
7:   while $c \neq \emptyset$ do
8:     $S \leftarrow S \cup \{c\}$
9:     $P \leftarrow P \cap \{c' \in P \mid M(c, c') > 0\}$
10:    $c \leftarrow$ FindHeaviest$(\Pi, P)$
11:  end while
12:  if $|S| \geq \theta$ then
13:    for $c' \in S$ do
14:      $\Pi \leftarrow \Pi \setminus \{c'\}$
15:    end for
16:    $\Pi \leftarrow \Pi \cup (\bigcup_{c' \in S} c')$        ▷ merge the clique into one cluster
17:  else
18:    $F \leftarrow F \cup \{s\}$
19:  end if
20: end while
21: return $\Pi$
```

Algorithm 4.6 FindHeaviest(Π, P) – “heaviest first” strategy of choosing the next node to add to a clique

```
1: $c \leftarrow \emptyset$
2: $max \leftarrow 0$
3: for $c' \in P$ do
4:   $sum \leftarrow \sum_{c'' \in \Pi} M(c', c'')$
5:   if $sum > max$ then
6:     $max \leftarrow sum$
7:     $c \leftarrow c'$
8:   end if
9: end for
10: return $c$
```

Algorithm 4.7 SecondPass(G, Π) – second pass of the clique-based approach: clusterization of the remaining clusters

```
while $\exists_{c',c'' \in \Pi} M(c', c'') > 0$ do
    $S \leftarrow \emptyset$
    $P \leftarrow \Pi$    ▷ possible extensions
    $c \leftarrow$ FindHeaviest$(\Pi, P)$
    while $c \neq \emptyset$ do
        $S \leftarrow S \cup \{c\}$
        $P \leftarrow P \cap \{c' \in P \mid M(c, c') > 0\}$
        $c \leftarrow$ FindClosest$(\Pi, S, P)$
    end while
    for $c' \in S$ do
        $\Pi \leftarrow \Pi \setminus \{c'\}$
    end for
    $\Pi \leftarrow \Pi \cup (\bigcup_{c' \in S} c')$    ▷ merge the clique into one cluster
end while
return $\Pi$
```

The second pass is very similar to the first one, there are however two important differences. Firstly – it was already mentioned above – cliques of any size are merged. Secondly, a different strategy of choosing the next cluster to merge is employed.

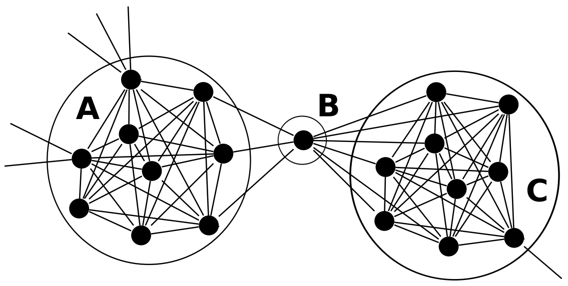


Figure 4.4: A typical setting in the second pass of the heuristic. Among the three clusters, $A - B$ and $B - C$ are still mergeable. If B is selected as the initial cluster, then assuming the "heaviest first" strategy of cluster expansion, $A - B$ will be merged. On the other hand, assuming the "closest first" strategy, $B - C$ will be merged.

The reason for such a change is illustrated in Figure 4.4. The figure represents a typical setting during the second phase of the execution of the algorithm. Assuming that cluster B was selected as the initial one (line 4 of Algorithm 4.7), the repeated application of the "heaviest first" strategy would select cluster A for expansion, while it is more relevant to choose cluster C. Therefore, the "closest first" strategy (cf. Algorithm 4.8) is used instead.

Algorithm 4.7 presents the pseudocode for the second pass of the "clique" approach, and Algorithm 4.8 presents the "closest first" strategy.

Algorithm 4.8 FINDCLOSEST(Π, S, P) – "closest first" strategy of choosing the next node to add to a clique

```
1: $c \leftarrow \emptyset$
2: $max \leftarrow 0$
3: **for** $c' \in P$ **do**
4:     $sum = \sum_{c'' \in S} M(c', c'')$
5:     **if** $sum > max$ **then**
6:         $max \leftarrow sum$
7:         $c \leftarrow c'$
8:     **end if**
9: **end for**
10: **return** $c$
```

4.3.4 Computational Complexity

Let us estimate the computational complexity of the proposed algorithm, that is, the time and space cost of its execution.

Let us start with assessing the cost of computing the omnipresent function $M(\cdot,\cdot)$. The space cost of storing the non-zero values of the function together with the clusters is limited by the number of clusters (at most $|V|$) and the number of edges $|E|$. Therefore, in order to store the non-zero values, one needs $\mathcal{O}(|V| + |E|)$ space. If the values stored at a given cluster are sorted, then one can access any value with time cost $\mathcal{O}(\log |V|)$.

Initially there are $|E|$ non-zero values of the function M, which need to be stored in $|V|$ domains, therefore the time cost of the initialization of M is $\mathcal{O}(|V| + |E|)$. Each merge requires inspecting each non-zero value of M at most once, and since the number of non-zero values of M decreases in time, the upper estimate on the number of inspections is $|E|$. Sorting and storing the non-zero values associated with the newly merged cluster requires $\mathcal{O}(|V|)$ operations (using counting sort [24], for example), therefore the total time cost of update of M after a merge is $\mathcal{O}(|V| + |E|)$.

In the FINDHEAVIEST function (algorithm 4.6), the sum in line 4 may be precomputed during cluster merges. The sum is calculated in $\mathcal{O}(|E|)$ time after each merge. All sums can be stored in a priority queue [24], with additional space cost of $\mathcal{O}(|V|)$, which allows retrieval of the "heaviest" cluster in $\mathcal{O}(\log |V|)$ time. Updates of the priority queue during cluster merges have amortized time cost of $\mathcal{O}(\log |V|)$ per merged component. Note that the costs associated with cluster merges are dominated by the cost of update of M, which is $\mathcal{O}(|V| + |E|)$.

Each call to FINDCLOSEST (algorithm 4.8) contains $\mathcal{O}(|V| + |E|)$ accesses to M, that is, a call takes $\mathcal{O}((|V| \log |V| + |E| \log |V|)$ time.

In the first pass, there are at most $|\Pi| \leq |V|$ iterations of the outer loop (line 2), since each iteration either merges a clique S (decreasing the number of available clusters by the size of the clique), or expands the set of forbidden initial cluster F. For each "failed" initial cluster, there are at most θ iterations of the inner loop (line 7), therefore at most θ calls to FindHeaviest. On the other hand, in the case of finding a sufficiently large clique, each call to FindHeaviest results in decrease in $|\Pi|$.

Therefore, there are at most $\theta|V|$ iterations of the inner loop at line 7. Each is dominated by the update of P in line 7, which takes $\mathcal{O}(|V|\log|V|)$ time in the worst case, giving $\mathcal{O}(\theta|V|^2\log|V|)$ total time. There are also at most $|V|$ cluster merges, which results in $\mathcal{O}(|V|^2\log|V| + |V||E|)$ time cost. Summing up, the total time cost of the first pass is:

$$\mathcal{O}(\theta|V|^2\log|V| + |V||E|)$$

In the second pass, every first call to FindClosest in the loop starting at line 5 returns a non-empty cluster (if everything else fails, at least the cluster returned by FindHeaviest is acceptable). Also, since returning an empty cluster ends the loop, then at least half of all calls to FindClosest return a non-empty domain. Each call to FindClosest reduces the number of clusters $|\Pi| \leq |V|$ by one, therefore there are $\mathcal{O}(|V|)$ calls to FindClosest. Note that there are as many updates of P as there are calls to FindClosest, and each update takes $\mathcal{O}(|V|\log|V|)$ time. There are $\mathcal{O}(|V|)$ calls to FindHeaviest as well. Finally, there are also $\mathcal{O}(|V|)$ cluster merges (at line 13).

Summing up, the total time cost of cluster merges in the second pass is $\mathcal{O}(|V|^2 + |V||E|)$, the total cost of calls to FindHeaviest is $\mathcal{O}(|V|^2 + |V||E|)$ the total cost of calls to FindClosest and updates of P is $\mathcal{O}(|V|^2\log|V| + |E||V|\log|V|)$. Therefore, the total cost of the second pass is:

$$\mathcal{O}(|V|^2\log|V| + |E||V|\log|V|)$$

Finally, the computational complexity of initialization and both passes, the total time cost of the "clique" heuristic is:

$$\mathcal{O}(\theta|V|^2\log|V| + |E||V|\log|V|)$$

using $\mathcal{O}(|V| + |E|)$ space.

4.3.5 Summary

We have presented a heuristic solving Minimum Partition problem, based on an approximate solution of the classic Maximum Clique problem. The applicability of the approach is limited to almost complete graphs, i.e. complete graphs in which some edges were removed (introducing incompleteness) while some other were added (possibly introducing inconsistency).

4.4 "Spatial" Approach

Our second heuristic combines the greedy strategy described in Section 4.2 and a force-based node placement.

4.4.1 Motivation

The idea is to arrange the graph's nodes in space in such a way that the distance between a pair of nodes is related to their affinity. We introduce a set of conditions: we want each pair of same-color nodes to be as far away from one another as possible, and we want each pair of the nodes connected by a link to be at a certain distance R. Later, we will assume that the closer a pair of nodes is, the more probable it is that they are equivalent. Thus, our revised task is to find such an arrangement of the graph's nodes is space that as many of the above conditions are satisfied as possible.

Having done that, a greedy approach builds the set of accepted (valid) links starting from the shortest ones: each new link is added to the set as long as it satisfies the coherence constraints.

4.4.2 Node Placement

We place the nodes in an N-dimensional space. It is done is such a way that the calculation of graph nodes' positions may be reused in in graph visualization. The choice of N is driven by a couple of factors. Graph analysis requires only that $N \geq 2$, with the classic "amount of work vs. quality of results" trade-off. Visualization is usually restricted to two or three dimensions, the latter being both more impressive and resource-consuming. Since node placement calculations are the most computationally-intensive part of the whole approach, it is reasonable to do the calculations once, and apply them to both analysis and visualization. In such a case, the choice of N is restricted to $2 \leq N \leq 3$.

Node placement uses the classic force-based paradigm, introduced by Kamada and Kawai [57]. A potential is defined, assigning each vector of node positions a scalar value, and the task is to find a sufficiently deep local minimum of the potential. The first challenge is thus to define the potential in such a way that its minima correspond to the desired node positions, and at the same time the calculation of its value and gradient at any given point is feasible.

We choose the potential to be:

$$\begin{aligned} \mathcal{P}(x_1, x_2, \ldots, x_{|V|}) := \sum_{\{v_i, v_j\} \in E} & \mu_{ij} \big(|x_i - x_j| - R \big)^2 \\ & + \lambda R^3 \sum_{\{v_i, v_j\} \in F} \frac{1}{|x_i - x_j|} \end{aligned} \tag{4.6}$$

where $\mu_{ij} := \mu(\{v_i, v_j\})$ is the weight of the edge connecting v_i and v_j, and $F := \{\{v_i, v_j\} \in 2^V | \kappa(v_i) = \kappa(v_j) \wedge i \neq j\}$ is the set of pairs of nodes of the same color. The possibility to derive the gradient $\nabla\mathcal{P}$ analytically is a huge advantage during the numerical minimization:

$$\begin{aligned}\nabla\mathcal{P}(x_1, x_2, \ldots, x_{|V|}) = 2 \sum_{\{v_i,v_j\}\in E} \mu_{ij}\big(|x_i - x_j| - R\big)\frac{\nabla|x_i - x_j|}{|x_i - x_j|} \\ -\lambda R^3 \sum_{\{v_i,v_j\}\in F} \frac{\nabla|x_i - x_j|}{|x_i - x_j|^3}\end{aligned} \tag{4.7}$$

The potential models two types of force: harmonic for linked pairs, and repulsive for nodes of the same color. The harmonic potential for a given pair achieves its minimum (= 0) when the pair is separated by distance R, the repulsive one is minimal when the given pair is separated by an infinite distance.

There are two parameters in the above equations: R and λ. The former is purely decorative, since any positive value of R yields the same minimum (up to scaling). The latter parameter controls the balance between the harmonic and repulsive forces. In practice λ should not be too small, since then the repulsive force will be too weak to separate the tight cores (impairing both the analysis and visualization). On the other hand, if the ratio is too big, then the cores will be extremely far away from one another and although the analysis will be still possible, the layout will be neither aesthetic nor suitable for visualization. Empirically, we have established that the best results for medium-sized cases (hundreds of nodes) are obtained when λ is in the range $10^1 \div 10^2$.

The minimization is performed using the Conjugate Gradients method [84], without preconditioning, with a simple gradient-only linear search.

4.4.3 Edge Ordering

Having found the vertex locations in space, we may easily sort the edges according to their length and feed them to the greedy algorithm presented in Section 4.2. This is motivated by the assumption that shorter edges are more likely to be correct. since the balance of attractive and repulsive forces acting on a short link did not manage to stretch it.

4.4.4 Computational Complexity

Although the number of iterations of the conjugate gradient optimization cannot be precisely estimated, we can determine the computational complexity of two key operations performed during this procedure, namely: the calculation of the potential's value and the calculation of potential's gradient. Both operations contain the same loops: one over the edges and one over the "forbidden" pairs. The loops over the edges clearly incur a time cost of $\mathcal{O}(|E|)$. In a pessimistic case, there may be $\mathcal{O}(|V|^2)$ forbidden pairs, and thus that is the time cost of the second type of loops. Thus, the total time cost of a calculation of potential's value or gradient is $\mathcal{O}(|V|^2)$.

Fortunately, in typical cases observed empirically, a graph in question is almost complete, that is, it can be transformed to a complete graph in $\mathcal{O}(|V|)$ edge adds or removals. Furthermore, let us assume that the tight cores are all of similar sizes $\mathcal{O}(K)$, where K is the number of colors. In such a case, there are $\mathcal{O}(K^2\frac{|V|}{K}) = \mathcal{O}(|V|K)$ edges, and $\mathcal{O}(\frac{|V|^2}{K^2}K) = \mathcal{O}(\frac{|V|^2}{K})$ forbidden pairs.

The potential's formulation exploits the fact that we already know the "forbidden" pairs which we want to separate in space. With this information, the repulsive part is cheaper to calculate: instead of processing all $\mathcal{O}(|V|^2)$ pairs, we need to compare only the same-color pairs.

Edge ordering, as any ordinary comparison sort, incurs $\mathcal{O}(|E|\log|E|)$ time cost.

4.5 "Genetic" Approach

Our next heuristic is an example of a genetic algorithm. It defines a fitness function on candidate solutions and simulates the natural selection process.

4.5.1 Motivation

Observe that the computational problem exhibits a certain kind of locality. A locally-minimal cut F_1 may contain the optimal set of cuts on a given subset V_1 of the graph, while another locally-minimal cut F_2 may contain the optimal set of cuts on another subset V_2. It is tempting to look for a method of combining the cuts in order to obtain a cut F_{12} that would be better than both F_1 and F_2. The above notions bring to mind the so-called Genetic Algorithms.

4.5.2 The Algorithm

In our approach we have chosen an unorthodox realization of the Genetic Algorithm technique. Nevertheless, the key features of the original framework have been retained. Algorithm 4.9 presents the approach.

We start by generating n pseudo-random locally-minimal cuts (line 2). The cuts are generated by running the greedy algorithm described in Section 4.2 with a random permutation of edges as input. Note that due to the method used, some cuts may be generated more frequently than other.

The remaining part of the algorithm resides in a loop (starting at line 6). First, $2n - 2m$ parents ($n - m$ pairs) are chosen using the roulette-wheel selection, with the

Algorithm 4.9 GeneticCut(G) – a genetic algorithm

```
1: for $i \in [0, n-1]$ do
2:     $H[i] \leftarrow$GreedyMerge$(G,$ Shuffle$(E))$    ▷ first generation
3: end for
4: $F, v \leftarrow E, \infty$    ▷ minimal cut and its weight initialized to $\infty$
5: $t \leftarrow 0$
6: while $t < t_{max}$ do
7:     $H' \leftarrow$Roulette$(G, H, 2(n-m))$    ▷ draw parent cuts
8:     for $i \in [0, n-m-1]$ do
9:         $H[i] \leftarrow H'[2i] \cup H'[2i+1]$    ▷ merge parent cuts
10:        $H[i] \leftarrow$GreedyMerge$(G,$ Shuffle$(E \setminus H[i]) \cup$ Shuffle$(H[i]))$
11:    end for
12:    for $i \in [n-m, n-1]$ do
13:        $H[i] \leftarrow$GreedyMerge$(G,$ Shuffle$(E))$    ▷ add random cuts to the new generation
14:    end for
15:    $v' \leftarrow \min_{i \in [0,n-1]} \|H[i]\|_G$
16:    if $v' < v$ then
17:        $v \leftarrow v'$    ▷ new best cut found
18:        $F \leftarrow \arg\min_{F' \in H} \|F'\|_G$
19:        $t \leftarrow 0$
20:    else
21:        $t \leftarrow t + 1$
22:    end if
23: end while
24: return $F$
```

fitness of a cut F defined as:

$$\phi(F) := \frac{1}{1 + \sqrt{\|F\|_G}} \tag{4.8}$$

Next, each pair of parent cuts (F_1, F_2) produces an offspring cut. In order to do this, first a cut $F = F_1 \cup F_2$ is taken. Note that this is not a locally-minimal cut, since $F_1 \subset F$ and $F_2 \subset F$. Our goal is to randomly choose a locally-minimal cut that is a subset of F. In order to do that, we run the GreedyMerge algorithm from Section 4.2 with the sequence:

$$\vec{E} := \text{Shuffle}(E \setminus F) \cup \text{Shuffle}(F) \tag{4.9}$$

as input. All the edges in the first part of the sequence are guaranteed to be accepted, thus the resulting cut is decided by the random permutation of the edges in F. Note that both F_1 and F_2 are among the potential results.

Next, we add m random cuts to the new generation. Finally, we check whether a cut better than the current best cut was found. If no new best cut is found in t_{max}

iterations of the main loop, the current best cut is returned as an answer.

Our algorithm diverges from the model GA described earlier in a couple of point. Most ostensibly, binary code is not used to encode cuts. As a consequence, the crossover operation was replaced by a offspring generation procedure that is very specific to the problem at hand. Lack of mutations is balanced by adding m random cuts to every generation.

4.5.3 Computational Complexity

The number of iterations of the algorithm cannot be predicted, we can, however, estimate the time cost of an individual iteration. Each iteration is dominated by the need to perform n GreedyMerge operations. The time cost of each iteration is thus $\mathcal{O}(n|E|K)$, where K is the number of colors in the graph.

4.6 "Girvan-Newman" Approach

The last heuristic combines the greedy strategy described in Section 4.2 and a slightly modified Girvan-Newman algorithm.

The Girvan-Newman algorithm applies to unweighted, uncolored graphs. Our slight modification renders the algorithm applicable to colored graphs. However, the restriction to unweighted graphs still holds.

4.6.1 Motivation

Recall that the Minimum Partition problem seeks to minimize the total weight of edges that are truncated by a given partition. Two vertices $v_1, v_2 \in V$ cannot be located in the same domain (i.e. every path $v_1 \leftrightsquigarrow v_2$ needs to be cut) iff they have the same color, i.e., $\kappa(v_1) = \kappa(v_2)$. Effectively, our task is to separate each vertex $v \in V$ from all the other vertices of the same color, and we want to cut as little as possible, by minimizing the sum of weights of removed edges.

One idea is to look for bottlenecks in the sum of flows between same-colored vertices and derive an analogue of the max-flow min-cut theorem [82]. If fact, a very special case of the main computational problem may be solved using the Ford-Fulkerson algorithm [41] for finding a maximum flow[3]. Unfortunately, Dahlhaus et al. have shown that generalizations of the maximum flow problem are NP-hard [25] and we have used that result while establishing the computational complexity of the Minimum Partition problem in Section 4.1.

Another concept which captures the notion of network bottlenecks is so-called centrality. In fact, a number of different loosely related measures called centrality were

[3]More precisely, the special case is a graph in which each vertex except one pair has a unique color.

developed over time [15]: closeness centrality [89], graph centrality [52], stress centrality [94], betweenness centrality [44, 4], and edge betweenness centrality [48]. All the above measures aim to quantify the extent to which a given edge or vertex is critical in the graph. We shall use the last flavor (edge betweenness), which has already been successfully used in for analyzing social [77] and biological [110] networks.

4.6.2 The Algorithm

Girvan and Newman proposed a measure called edge betweenness, and showed its application in analyzing community structure [77, 48, 72]. Edge betweenness of a given edge is defined as the number of shortest (geodesic) paths that pass through the edge. The idea of their algorithm is to iteratively find and remove the edge with the highest edge betweenness. Unfortunately, each edge removal changes the betweenness of all the edges in the connected component(s) to which the edges' endpoints belong. Therefore, each edge removal is followed by recalculation of edge betweenness. Note that this approach studies the network's topology, but does not take into account the weights of individual edges.

The output of the Girvan-Newman algorithm (sequence of edge removals) is fed, in the reversed order, to the greedy meta-algorithm presented in Section 4.2. In other words: first, the edges are removed from the graph based on edge betweenness calculations and then they are greedily added back (if possible) in the reversed order.

First optimization, a fairly obvious one, is to check, after each edge removal, whether the resulting graph is coherent, and if so, stop the Girvan-Newman algorithm and run the greedy meta-algorithm.

Another modification of the original algorithm stems from the observation that there is no need to calculate edge betweenness in the coherent components of the processed graph. Indeed, all the edges removed from a coherent component will eventually be added back, since there is no constraint that would stop the greedy algorithm from doing so. Therefore, after each edge removal we find the incoherent components and perform all the subsequent calculations on the incoherent subgraph only. Note that our modifications do not change the overall result of the combination of Girvan-Newman algorithm and greedy approach, they only allow to obtain the result faster.

Algorithm 4.10 implements the above remarks. The VERTICESOFINCOHERENT(G) function returns the vertices in G that are located in the incoherent components of the graph. The EDGEBETWEENNESS(G) returns the edges in G sorted descending by their betweenness (a part of the original Girvan-Newman algorithm), and HEAD$(\vec{E})$ returns the first element of list $\vec{E}$.

Since Algorithm 4.10 is computationally expensive, it is tempting to run EDGEBETWEENNESS only once and feed the results to GREEDYMERGE, as in Algorithm 4.11. However, this solution is clearly inferior to GIRVANNEWMANCUT, and should be used

Algorithm 4.10 GIRVANNEWMANCUT(G) – a heuristic based on the Girvan-Newman algorithm

```
1: $\vec{E} \leftarrow ( )$
2: **repeat**
3:     $I \leftarrow$ VERTICESOFINCOHERENT$(\langle V, E \setminus \vec{E}, \kappa \rangle)$
4:     **if** $I \neq \emptyset$ **then**
5:         $e \leftarrow$ HEAD(EDGEBETWEENNESS$(\langle V, E \setminus \vec{E} \rangle|_I)$)
6:         $\vec{E} \leftarrow (e) \cup \vec{E}$
7:     **end if**
8: **until** $I = \emptyset$
9: **for** $e \in E \setminus \vec{E}$ **do**
10:    $\vec{E} \leftarrow (e) \cup \vec{E}$        ▷ *put edges from the coherent components in front*
11: **end for**
12: $F \leftarrow$ GREEDYMERGE$(G, \vec{E})$
13: **return** $F$
```

only as a last resort – in the cases when problem size renders the full algorithm infeasible.

Algorithm 4.11 BETWEENNESSCUT(G) – a heuristic based on a single edge betweenness calculation

```
1: $\vec{E} \leftarrow$EDGEBETWEENNESS$(G)$
2: $F \leftarrow$ GREEDYMERGE$(G, \vec{E})$
3: **return** $F$
```

4.6.3 Computational Complexity

The pessimistic time cost of the Girvan-Newman algorithm is $\mathcal{O}(|E|^2|V|)$. This means that in the case of dense graphs, the algorithm's time complexity becomes $\mathcal{O}(|V|^5)$!

However, let us repeat the reasoning conducted in Subsection 4.4.4 and derive a more optimistic time complexity estimation. Assume, as we did before, that the input network is $\mathcal{O}(|V|)$ edge additions and removals away from a network consisting of isolated cliques of size $\mathcal{O}(K)$. Then, there are $\mathcal{O}(|V|K)$ edges and each update of edge betweenness takes $\mathcal{O}(|V|^2K)$ time. Fortunately, for a sufficiently high values of K, the $\mathcal{O}(|V|)$ links connecting different cores are removed first, and in that case the algorithm runs in $\mathcal{O}(|V|^3K)$ time.

4.7 Summary

We have shown that the computational problem studied in this thesis is NP-hard. Next, we have proposed five algorithms giving approximate solutions of the problem. Obviously, a meta-algorithm running all five algorithms and returning the best answer is a viable alternative.

Algorithm	Space	Time	Time (cores)
GreedyCut	$\mathcal{O}(VK)$	$\mathcal{O}(EK)$	$\mathcal{O}(K^2V)$
CliquesCut	$\mathcal{O}(V+E)$	$\mathcal{O}(\theta V^2 \log V + EV \log V)$	$\mathcal{O}((\theta+K)V^2 \log V)$
SpatialCut	$\mathcal{O}(V+E)$	$\mathcal{O}(tV^2 + E \log E)$	$\mathcal{O}(tV^2/K + VK(t + \log(VK))))$
GeneticCut	$\mathcal{O}(nE+VK)$	$\mathcal{O}(tnEK)$	$\mathcal{O}(tnK^2V)$
GirvanNewmanCut	$\mathcal{O}(V+E)$	$\mathcal{O}(E^2V)$	$\mathcal{O}(V^3K)$

Table 4.1: Space and time complexities of the five algorithms presented in previous sections. Here: V is the number of vertices, E, the number of edges, K the number of colors, t is the number of function evaluations (SpatialCut) or iterations (GeneticCut). Other symbols are inputs of respective algorithms. The last column shows the time cost in the case when the input is a "skeleton of cores" (see text).

All of the five heuristics are inspired by some earlier results in the respective fields. The fields are quite diverse, including: classic computer science problems, numerical optimization, graph visualization, genetic algorithms and community detection in social networks.

Note that CliquesCut and GirvanNewmanCut are not general-purpose solutions, as they assume certain level of modularity of the input network. The remaining algorithms are applicable to any problem instance.

Table 4.1 summarizes the space and time costs of the proposed algorithms. Note that we are using the $\mathcal{O}$ notation, which states the upper bound of the asymptotic behavior of a given property. The table includes time complexities in the special case of input networks of the "skeleton of cores" type. More precisely, this special case assumes that an input network consists of $\Theta(V/K)$ cores, each containing $\Theta(K)$ vertices and $\Theta(K^2)$ edges, as well as $\Theta(V)$ edges each connecting two different cores.

It should be stressed that a number of arbitrary qualitative decisions were made while designing the proposed algorithms. This fact is perhaps most clearly visible in the case of CliquesCut. Other examples include: the choice of Conjugate Gradient as the method of numerical optimization in SpatialCut, the choice of fitness function in GeneticCut. These arbitrary decisions were made using a combination of author's intuition and "trial and error" exploration based on test problem instances. Therefore, there is obviously a lot of room for improvement and more systematic calibration. Quantitative decisions, on the other hand, are extracted and expressed in the form of function parameters to be supplied upon execution.

Chapter 5

Experimental Results

In this chapter, we evaluate the approaches proposed in Chapter 4 on the network of Wikipedia's interlanguage links described in Chapter 3.

5.1 Methodology

We will be interested in the quality of cuts and running speed of the following 8 algorithms:

- **MNR** – minimum of 10 GreedyCut runs
- **AVR** – average of 10 GreedyCut runs
- **MXR** – maximum of 10 GreedyCut runs
- **CLI** – a single CliquesCut run with $\theta = 5$
- **GEN** – a single GeneticCut run with $n = 100$, $m = 10$ and $t_{max} = 5$
- **SPA** – a single SpatialCut run with $R = 1$, $\lambda = 40$, and with the number of Conjugate Gradient iterations limited to 80.
- **BET** – a single BetweennessCut run
- **G-N** – a single GirvanNewmanCut run

The first three algorithms, producing essentially random answers, serve as points of reference for the remaining algorithms.

As it was mentioned earlier, the computational problem studied in this thesis has not been explored before. Therefore, there is no real state-of-the-art algorithm to which our results could be compared. To a certain degree, the G-N algorithm, which is only a slightly modified (adapted) version of the Girvan-Newman algorithm for community detection, might be considered the closest state-of-the-art algorithm. However, the original Girvan-Newman algorithm is meant to be applied to different types of networks (i.e., colorless), and returns a different type of result (a dendrogram). Therefore, the modified version of the algorithm, adapted to a new setting, does not constitute a good benchmark.

The BET and G-N algorithms do not take edge weights into account, so in order to get a reliable comparison, we will test all the algorithms on unweighted graphs only.

Our test will be performed on the total of 86123 connected components of two networks of Wikipedias interlanguage links: one of ILLs between articles and one of ILLs between categories. The two networks have been recreated from database dumps downloaded on October 12, 2009. Each connected component shall be analyzed separately, as if it was an independent graph.

We have divided the components into three categories, based on the number of nodes (non-redirect pages) they contain:

- **small** – 85870 components having at most 300 nodes;
- **medium** – 227 components of size 301-1000;
- **big** – 26 components having more than 1000 nodes.

The largest component, consisting of 126776 nodes, is absent from the test, since it could be processed by only two fastest methods (MNR-AVR-MXR and CLI).

5.2 Results

5.2.1 Quality of Cuts

Table 5.1 shows average weights of cuts returned by all the tested algorithms. The results are grouped into three categories of components, as described above. The G-N algorithm was too slow to be applied to the medium and big components.

Category	MNR	AVR	MXR	CLI	SPA	GEN	BET	G-N
small	13.53	36.62	84.47	12.13	12.25	**10.41**	15.16	11.63
medium	1138	2131	3544	604	911	**469**	1109	N/A
big	3073	4958	7541	1214	1986	**1031**	3100	N/A

Table 5.1: Average weights of cuts returned by the proposed algorithms. The results are grouped by component classes. For each category, the best (lowest) values are bold.

The results clearly show that GEN is the best approach. In the case of small components, G-N is second, followed by CLI and SPA, each of which perform better than the "best of 10 random" – MNR. In all the cases, BET is on par with, or slightly inferior to, MNR, while both CLI and SPA are considerably better than MNR.

Category	MNR	AVR	MXR	CLI	SPA	GEN	BET	G-N
small	92.15%	33.18%	16.73%	85.33%	95.04%	**99.92%**	90.00%	92.88%
medium	1.76%	0.00%	0.00%	10.13%	10.13%	**96.47%**	4.84%	N/A
big	3.84%	0.00%	0.00%	30.76%	15.38%	**100.00%**	26.92%	N/A

Table 5.2: Fraction of cases for which a given algorithm yields the best cut. The results are grouped by component classes. For each category, the value of the most efficient algorithm is bold.

Moving on, let us now take a look at Table 5.2 which presents the fraction of cases for which a given approach returns the best result. Note that we do not know the correct solution (i.e., the minimal cut) for each test case, therefore this is not necessarily the fraction of cases for which a given approach returns the *correct* result.

GEN is the algorithm that yields the best result most often: for 99.92% of the small components, for 96.47% of the medium components, and for all the large components, GEN returned the best answer. We can see that the small components category is actually very easy: even MNR, i.e., the best of 10 random cuts, yields the best result over 90% of the time. On the other hand, in the case of medium and big components, MNR is among the best less than 5% of the time. This means that the last two categories are significantly more challenging.

5.2.2 Running Times

Running times are aggregated in Table 5.3. The greedy approach, even when repeated 10 times (MNR, AVR, MXR) is still clearly the fastest in the case of small and medium components, while in the case of big components, CLI takes the lead by a small margin.

Category	**R	CLI	SPA	GEN	BET	G-N
small	**26**	120	1 057	850	184	10 737
medium	**2 334**	5 557	50 825	282 233	45 415	N/A
big	70 920	**69 028**	531 525	5 918 445	8 645 203	N/A

Table 5.3: Average running times (in milliseconds) of the proposed algorithms. The results are grouped by component classes. For each category, the value of the fastest algorithm is bold.

On the other extreme, G-N is clearly the slowest one, and it is not feasible to run it for larger problem sizes. Even BET, which is G-N reduced to a single iteration, takes over two hours on average in the case of big components. SPA is reasonably fast, there is however a trade-off between the running time and the quality of results. In this experiment, number of iterations of Conjugate Gradient method was limited to 80, which places this particular implementation of the spatial approach on the "fast and inaccurate" side. GEN, the approach yielding the best results, is at the same time one of the slowest.

5.2.3 Selected Observations

Let us now take a look at the quality of cuts in individual test cases. Table 5.4 presents the answers produced by the 7 algorithms for all the 26 big components. As we already know from the aggregated results presented in Table 5.2, GEN returns the best answer in all the cases. BET often returns very good answers (the best result in 7 cases), and very bad answers (worse than MNR in 6 cases, in one case even worse than AVR).

One test case, namely `557FD07F`, is fairly easy: consists of big and tight clusters with very sparse connections between them. The test is, nevertheless, challenging in terms of size, as the component contains over quarter of million links.

Applicability of BET and G-N

In order to explain the reasons for occasional very poor results of BET, let us analyze a test case from the big set (`0C42DB42`), a fragment of which is visualized in Figure 5.1. The interesting fragment contains the articles on "January 26". There are two articles from the Tatar language edition competing for connection with the rest of the group (the two articles are located on the opposite sides of the figure). Recall that, per the definition of a partition (cf. Equation 1.7), at most one of the two Tatar articles may be connected to the non-Tatar articles on "January 26". Occurrences of this pattern (that is: two nodes of the same color, both well-connected to a clique) are the source of extremely poor performance of BET, as well as G-N. This is because all the nodes together form a tight community, and forcing an admissible partition (by means of the GREEDYMERGE subroutine) will randomly split the community into two parts, yielding an extremely high cost of the cut.

It should be stressed that this is not a drawback of the Girvan-Newman algorithm, Instead, the above observation proves that the adaptation of Girvan-Newman algorithm to the computational problem researched in this thesis has limited applicability.

Contrast the above case with another one, presented in Figure 5.2. The best cut in this test case partitions the nodes into four sets, which correspond to four topics: "Japanese braille", "Braille", "Cyrillic braille" and "Hebrew braille"[1]. The minimal cut is obtained by removing the links forming a bottleneck between "Braille" and "Japanese braille", removing the single link connecting "Braille" and "Hebrew braille", and disconnecting "fr:Braille cyrillique" from "Braille".

[1]Note that in general the minimal partition does not have to correspond to the "semantic" partition.

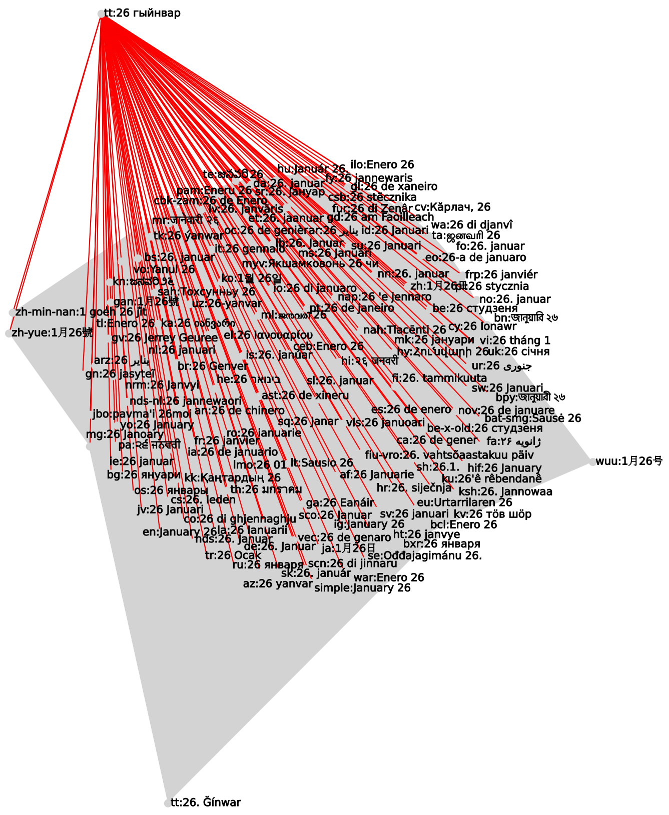


Figure 5.1: *A test case that is difficult for the G-N and BET algorithms. The model problem is a clique containing two nodes of the same color. Here, two articles in Tatar are competing for the same group of articles. Red lines represent the best cut.*

Component	Nodes	Links	MNR	AVR	MXR	CLI	SPA	GEN	BET
0C42DB42	3869	255232	295	3621	7455	**276**	281	**276**	2848
0EBA16A3	1701	8570	2038	2384	2862	2067	1517	**1462**	1604
1E0B534B	1187	20873	4474	5974	7448	3061	2810	**1687**	2607
25ECD94D	4549	106516	2272	4263	5637	1007	1169	**986**	1025
40EA68AE	2027	34794	339	818	1456	198	463	**190**	198
4B8C5F58	7832	494188	3574	10891	25081	**1035**	4459	**1035**	**1035**
4BB0A9F9	3881	248171	1009	4973	8303	**374**	**374**	**374**	3213
4E7C07B1	1091	19705	3873	4676	5632	1468	1810	**1098**	2942
534F3412	1045	18638	113	673	1455	**94**	**94**	**94**	**94**
557FD07F	4011	253246	**232**	2581	6591	**232**	**232**	**232**	**232**
56B1A372	1235	16208	2236	3025	4324	1044	1325	**918**	1393
5AA6E4BA	1302	22256	5254	6560	7464	2152	3679	**1712**	5111
5D2EA275	1689	65410	18076	21663	24124	5442	15949	**4943**	18006
6C72FBBA	5921	372930	1670	7658	16638	**1039**	1045	**1039**	16462
8672A237	1150	20255	3611	4746	6114	1271	1382	**951**	2532
96B93C7A	2274	39293	8070	9375	10394	2700	4180	**2491**	5307
A51F2E7A	1905	34103	207	531	1027	135	348	**127**	**127**
B559954A	1833	31302	203	983	1565	**154**	175	**154**	**154**
C54407E4	1395	25268	184	757	1892	119	**111**	**111**	**111**
C57187AA	1058	7401	1439	1699	2098	832	932	**704**	1156
CBA823C5	1016	18724	763	1345	2520	473	1098	**439**	575
CCCCE533	1214	64382	8675	12191	21687	1988	2857	**1834**	4064
D48EBEE8	3719	234073	938	3376	5814	**308**	313	**308**	3027
D5831EA2	2185	37712	209	1026	1713	145	145	**136**	**136**
E6C63BF8	1070	19594	4053	5383	7328	1523	1736	**1395**	2179
EDB82C74	2464	32576	6094	7752	9445	2431	3171	**2113**	4473

Table 5.4: Cut weights for the largest connected components (sizes over 1000). Each row represents a component. First three columns of the table are component's identifier, number of non-redirect pages and number of links, respectively. Next, cut weights for 7 tested algorithms are presented. For each row, the best (lowest) values are bold.

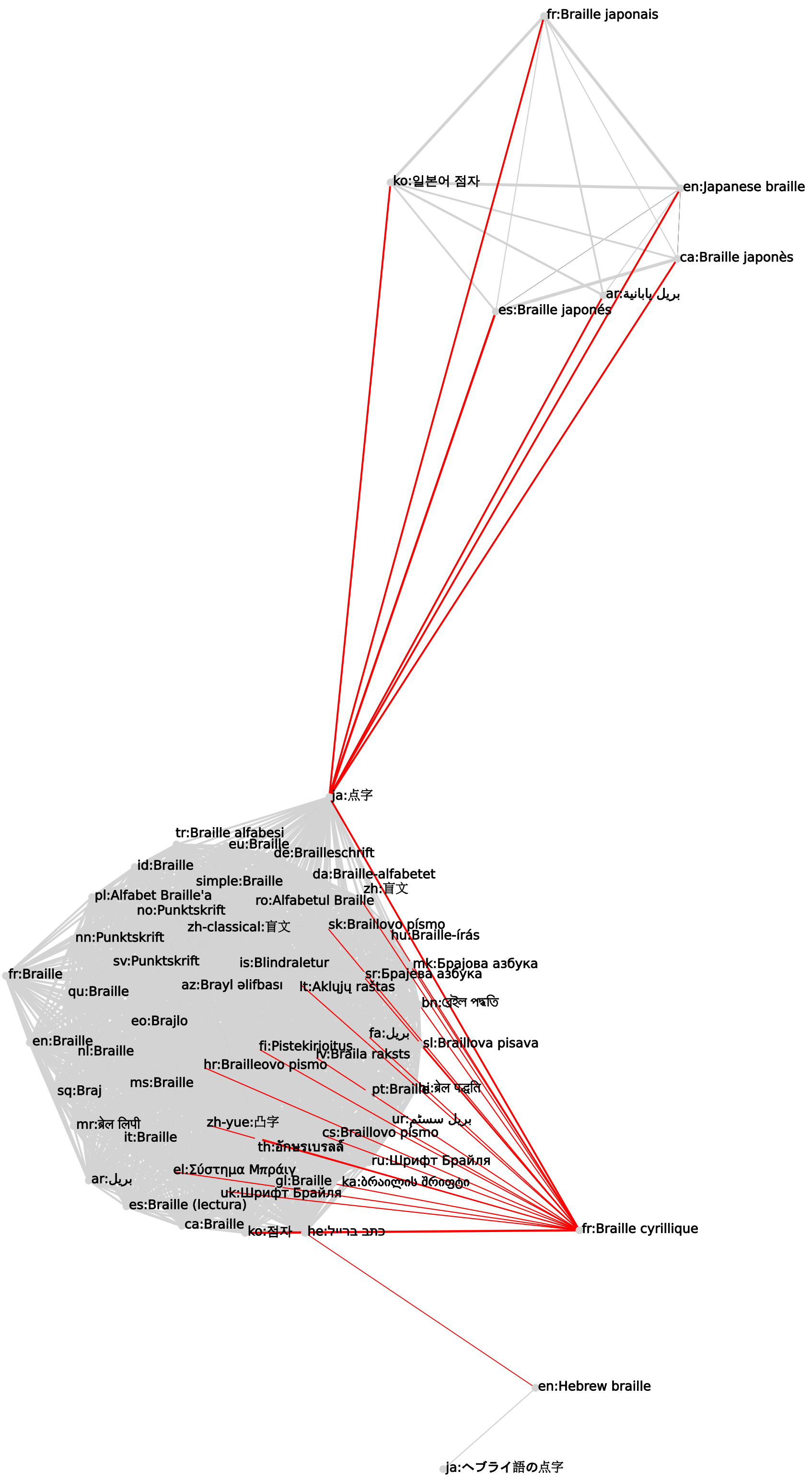


Figure 5.2: A test case that is properly solved by the G-N and BET algorithms. Red lines represent the best cut.

The non-French articles in the “Braille” cluster are well-connected to two French articles: “fr:Braille” and “fr:Braille cyrillique”, in a manner similar to the previous test case. At most one of the two French articles may be connected to the non-French articles on “Braille”. However, it this test, “fr:Braille cyrillique” has weaker connection that “fr:Braille” or any other member of the group, and this fact is reflected in edge betweenness: the edges incident to “fr:Braille cyrillique” have higher betweenness than any other edge in the cluster, and will be removed first. Therefore, the BET algorithm applied to this and similar test cases will have no problems identifying neither the bottlenecks, nor the competing same-color nodes.

Semantic drift

Let us take a look at two examples of semantic drift. Figure 5.4 presents a medium-sized semantic drift pattern spanning topics such as: “Nuclear power in Japan”, “Fusion power”, “Latent heat”, “Phase Change Material”, and “Bed warmer”. Each topic is represented by a circle whose area is proportional to the number of language editions covering the topic, and the width of segments connecting any two given topics is proportional to the number of links spanning the two topics. Figure 5.5 features a smaller example and a different presentation method: this time each individual node and link in the component is shown. This component contains politically-charged topics such as: “Kosovo”, “Metohija”, “2008 Kosovo declaration of independence”, “International recognition of Kosovo”, and “Kosovo status process”. Most of the incoherent links are incident to articles in Albanian and Serbian language editions.

The topology of connections between clusters returned by the algorithms proposed in Chapter 4 corresponds to the topology of “skeletons” generated by Algorithm 3.1 introduced in Chapter 3. Recall, however, that the latter algorithm did not return a partition of the given set of nodes and thus it was of little practical use.

It is not feasible to verify whether the best partitions returned by the algorithms proposed in Chapter 4 coincide with the true, semantic partitions of the articles and categories. Nevertheless, in the course of this research we have analyzed and critically assessed partitions of a couple of hundred components. Based on this, unorganized and definitely not methodical, probe we have concluded that the best algorithm, in vast majority of cases, yields the semantically correct partition. This result requires, however, to process weighted graphs, where edge weights reflect a coarse measure of content similarity. In our implementation, weight of an edge connecting articles A and B depends on:

- whether there are interlanguage links in both directions ($A \rightarrow B$ and $B \rightarrow A$) or not
- whether the interlanguage links point to a redirect or not

- the number of common categories of the two pages

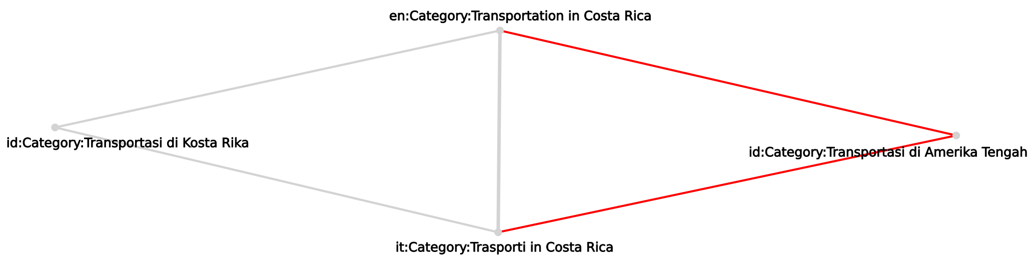

en:Category:Transportation in Costa Rica
id:Category:Transportasi di Kosta Rika
id:Category:Transportasi di Amerika Tengah
it:Category:Trasporti in Costa Rica

Figure 5.3: Illustration of an impact of edge weighting on node partitioning. In the absence of edge weights, there are two minimal partitions of this tiny incoherent component, only one of which is semantically correct. When weights associated with content affinity are introduced, the sole minimal partition is semantically correct.

Figure 5.3 shows a trivial example in which assigning weights is critical to obtaining semantically-correct cut. Without weights, disconnecting either of the two articles in Indonesian yields a minimal cut. Once weights are assigned, the "id:Category:Transportasi di Kosta Rika" category is better connected to the English and Italian categories than "id:Category:Transportasi di Amerika Tengah".

It goes without saying that even the most sophisticated weight assignment can only, at best, *reduce* the number of semantically-incorrect partitions. It is by no means a surefire solution generating semantically-correct partitions.

Network dynamics

Finally, let us compare certain characteristics of the network of interlanguage links observed at three different points in time. The largest connected component has risen from approx. 48 thousand articles in March 2008, to approx. 72 thousand in August 2008, to over 126 thousand in October 2009. Recall that the size of coherent components should not exceed 265 articles (i.e., the number of language editions). The number of incoherent connected components[2] has risen from approx. 60 thousand in August 2008 to over 85 thousand in October 2009.

There is little rotation in the set of incoherent components – new incoherent cases appear, but the old ones are rarely resolved. Instead, smaller incoherent components merge into larger ones in a process resembling coagulation. Unfortunately, it is difficult to propose and validate a convincing model of network dynamics based on two or three snapshots. All we can do at this moment is report the handful of observations that we have made.

[2]In the case of interlanguage links between articles.

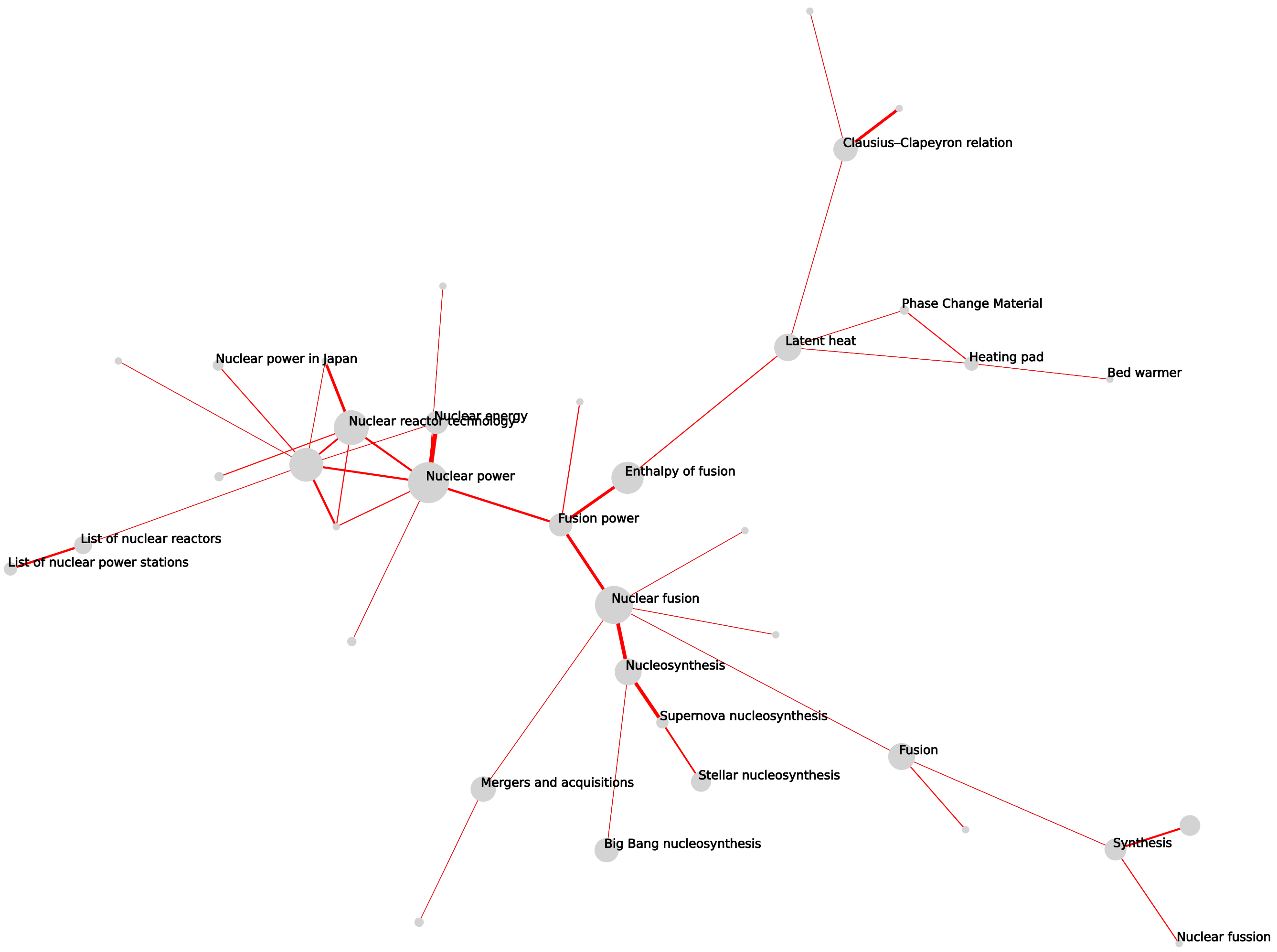


Figure 5.4: A medium-sized semantic drift pattern. A single connected component contains articles on several different topics, such as: "Nuclear power in Japan", "Fusion power", "Latent heat", "Phase Change Material" and "Bed warmer".

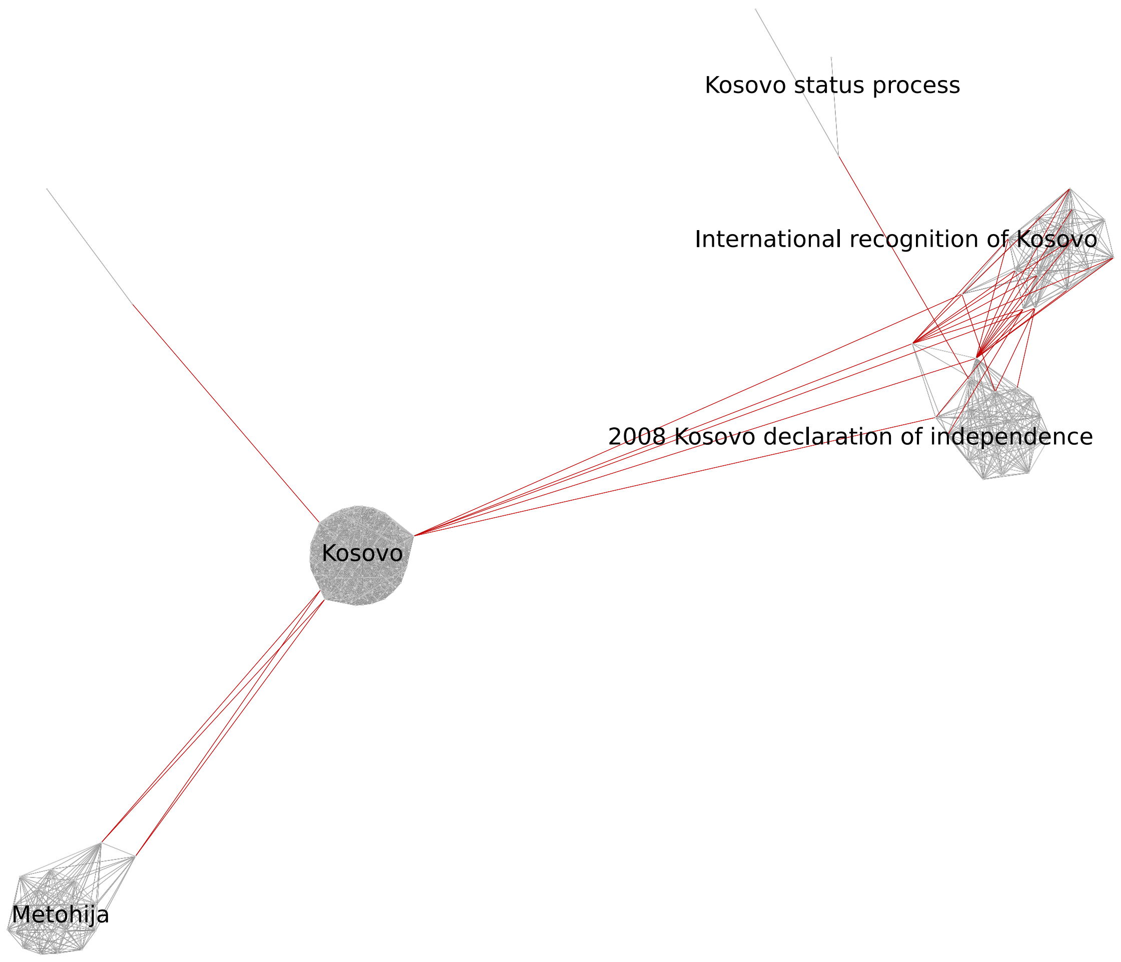

Kosovo status process
International recognition of Kosovo
2008 Kosovo declaration of independence
Kosovo
Metohija

Figure 5.5: A small-sized semantic drift pattern. The component contains articles on several topics related to the situation in Kosovo. A lot of incoherent links (red) are incident to articles in either Albanian or Serbian.

Chapter 6

Summary, Conclusions and Outlook

6.1 Summary

In this thesis we have researched a problem that is increasingly present in the global knowledge infrastructure.

The framework of our research consisted of six steps:

1. formulate the optimization problem;
2. understand the state of the art in related areas;
3. study an important example;
4. find feasible solutions;
5. validate the results experimentally;
6. draw conclusions.

Due care has been taken to separate theoretical work (steps 1 and 4) from empirical results, obtained in steps 3 and 5.

In the beginning (Chapter 1), we have presented the wider context of the problem in a top-down manner, starting with the big picture: the paradigm shift triggered by the Information Age. Next, we have sketched the landscape of Semantic Web, and focused on one particular challenge regarding equivalence relations in semantic networks. We have listed a number of real-world occurrences of the challenge, and offered a longer description of one particular example: semantic drift in the network of so-called interlanguage links in Wikipedia. Having done that, we have expressed the problem using formal, graph- and set-theoretic language: a subspace of "stable" graphs has been defined, and the optimization problem asked for the closest "stable" graph to the given "perturbed" one.

Next (Chapter 2), we have surveyed the relevant literature and discussed state-of-the-art approaches in related areas, including:

- summary of similar computational problems;
- review of network growth models;
- examples of occurrences of power-law distributions;
- advances in social network analysis (community detection in particular);
- review of several techniques used in this research, such as: clique and core finding, genetic algorithms, graph visualization;
- survey of academic studies focusing on Wikipedia;
- and summary of efforts to create multilingual dictionaries.

Moving on (Chapter 3), we have studied an example of a network in which the researched problem is particularly acute: the network of interlanguage links in Wikipedia. We have given a thorough description of the network's topology, made a number of surprising observations, and developed an algorithm that distills semantic drift patterns.

Next (Chapter 4), we have proved the researched problem to be NP-hard and offered five algorithms that attack the problem from five considerably different angles:

- a greedy algorithm;
- an iterated finding of the maximum clique;
- a numerical minimization of a multi-dimensional potential;
- a genetic algorithm;
- an adaptation of a community finding algorithm.

The computational complexity of each approach has been established as well.

Following that (Chapter 5), we have applied the proposed algorithms to over 85 thousand problem cases extracted from the network of interlanguage links in Wikipedia. Both the methodology and technical details were documented (the latter in Appendix A).

6.2 Conclusions

Results of the conducted research allow us to formulate the following conclusions.

First of all, **the optimization problem asking for the semantic drift reduction using as few cuts as possible is NP-hard**. This result has a profound impact on methods of reducing semantic drift. Instead of a polynomial-time algorithm yielding exact solutions, one has to settle for heuristics returning approximate answers.

Secondly, **interlanguage links in Wikipedia suffer from extensive semantic drift**. We have demonstrated that this is a growing problem, and that current approaches (surveyed in Chapter 2) fail to address it properly. The largest connected components are three orders of magnitude larger than expected. In the case of coherent components, we have noticed an impact of mass-generated articles and categories on the shape of component size distributions.

Thirdly, **the genetic algorithm presented in Section 4.5, applied to the interlanguage links, is very often capable of finding the correct semantic partition of articles.** This means that, to a large extent, it is possible to extract articles' meanings by analyzing only the structure of links – we do not have to analyze texts written in over 200 languages (which is clearly beyond current state of the art). Nevertheless, results of the genetic algorithm should be interpreted as mere educated guesses pending validation by a human.

6.3 Outlook

The research documented in this thesis answered a number of questions, but at the same time a number of problems were left untackled, and new challenges were outlined. Let us briefly describe possible extensions of this research.

6.3.1 Applications in Wikipedia

One of the obvious applications of the obtained results is correction of the incoherent interlanguage links in Wikipedia. This is a complex and interesting challenge for a couple of reasons. We assume that each edit recommendation generated by our algorithms should be validated by a user before it is committed[1]. Therefore, a well-designed web service is called for, one in which a user could efficiently browse through 85 thousand problem cases, and intuitively navigate, visualize, discuss, update and approve edit recommendation generated for each individual component (the largest of which contains well over 100 thousand nodes). Accountability should be thought through, since

[1]This statement contains an implicit assumption that the validating user has both necessary linguistic competences and good will. Frequent cases of vandalisms and blatant mistakes render this assumption dubious.

Wikipedians would probably like to know *who* approved *what* change, and *why*. Last but not least, the suggested web service should ideally be language-neutral (i.e. either all the messages should be localized or users should be able to interact with the system by means of universally understood icons).

Moreover, there is a non-trivial social challenge, namely, how to organize the community of casual editors and bot operators so that they would act in a coherent way. Right now, there is hardly even a consensus on the interpretation of interlanguage links, and human-bot or bot-bot "edit wars" regarding interlanguage links are not uncommon. Perhaps a way to achieve this goal is to raise the level of awareness of the problem and focus on defining common rules and guidelines regarding editing interlangauge links.

An alternative solution, much simpler but at the same time quite controversial, is to apply all the recommendations generated by our algorithm without human oversight, knowing full well that such an operation may and will locally do more harm than good. The goal is to "divide and conquer", that is, eliminate one gigantic problem and introduce a number of small, but easily manageable ones.

We have shared the results of our research with the Wikipedia community and initiated a cooperation aiming at validating and applying the edit recommendations generated by the algorithms presented in this thesis.

6.3.2 Model of Network Dynamics

In Chapters 3 and 5 we have seen two snapshots of the network of interlanguage links in Wikipedia, taken one year apart. Basing on this limited material, we have noted a dramatic increase of the number of problem cases and their complexity. We have also offered a hypothesis that the dynamics of interlanguage links are governed by a coagulation-like process.

In our opinion, a thorough study of the dynamics of interlanguage links would not only be an interesting and challenging academic endevour, but might help create solutions *preventing creation* of semantic drift patterns. It is probable that, in order to be accurate, a model of interlanguage links dynamics would have to assume multiple types of actors (e.g. vandals, bots, experts) and multiple types of page relations (e.g. equivalent, broader/narrower meaning, common theme, homonyms, false friends, unrelated). On the other hand, a more generic model is more likely to be useful outside the scope of Wikipedia.

6.3.3 Algorithm Refinements and New Approaches

As noted earlier, the heuristics presented in Chapter 4 contain a lot of arbitrary choices which have not been justified, and may probably be refined. Naturally, there is also room for radically new approaches.

Particularly hopeful is the field of community detection, from which the Girvan-Newman algorithm comes. Advances in this area result in an influx of new algorithms, some of which may be adapted to the our computational problem the way we have adapted the Girvan-Newman algorithm.

Appendix A

Implementation

In order to evaluate the theoretical results presented in Chapter 4, we have applied the algorithms to analyze the graph of articles from 265 language versions of Wikipedia. This chapter documents the data import process and an open-source implementation of the proposed methods. The results of the analysis and conclusions shall be presented in Chapter 5.

A.1 Technical Background

All the text content of all the language versions of Wikipedia is published under GNU Free Documentation License[1]. Wikimedia Foundation facilitates access to the content by publishing regular database dumps[2].

To understand format and content of the dumps, we start with a brief overview of the engine which powers Wikipedia and the structure of the underlying database. Next, we present the dumps and explain how and why did we import and postprocess them.

The description in this section includes a number of technical details, which may uninteresting for most readers, but valuable for ones willing to reproduce our results, or conduct similar research.

A.1.1 MediaWiki Engine

Each language version of Wikipedia is powered by an independent instance of the MediaWiki engine. The engine, written in PHP specially for the Wikipedia, is now an independent project. Apart from all the Wikimedia Foundation projects, MediaWiki is used by hundreds of other websites[3].

[1]See: `http://en.wikipedia.org/wiki/Wikipedia:Copyrights`

[2]See: `http://en.wikipedia.org/wiki/Wikipedia:Database_download`

[3]See: `http://www.mediawiki.org/wiki/Sites_using_MediaWiki`

By default, MediaWiki works with MySQL as database backend, but starting with version 1.8 released in October 2006, it fully supports PostgreSQL too.

A.1.2 MediaWiki Database Structure

Let us briefly describe the key tables in the database. We shall focus on MySQL backends, since this backend is used by all Wikipedia language versions.

The page table contains, among others, page ID, namespace and title. The ID is referenced in many other tables.

```
CREATE TABLE page (
  page_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
  page_namespace INT NOT NULL,
  page_title VARCHAR(255) BINARY NOT NULL,
  ...
  PRIMARY KEY page_id (page_id),
  UNIQUE INDEX name_title (page_namespace,page_title),
  ...
);
```

The pagelinks table contains page-to-page links between the latest revisions of the pages. Note that the target page is referenced by title instead of ID, links in Wikipedia may lead to nonexistent pages (encouraging creation of the missing article).

```
CREATE TABLE pagelinks (
  pl_from INT UNSIGNED NOT NULL DEFAULT ’0’,
  pl_namespace INT NOT NULL DEFAULT ’0’,
  pl_title VARCHAR(255) BINARY NOT NULL DEFAULT ’’,
  ...
  UNIQUE KEY pl_from (pl_from,pl_namespace,pl_title),
  ...
);
```

The langlinks table contains links to pages in other language versions. Again, the target page is referenced by title instead of ID, since the target language version of Wikipedia does not need to be, and often isn’t, stored in the same database. Note that it is by design impossible for a single page to contain two interlanguage links to the same language version (by the UNIQUE KEY) constraint.

```
CREATE TABLE langlinks (
  ll_from int UNSIGNED NOT NULL DEFAULT ’0’,
  ll_lang VARBINARY(20) NOT NULL DEFAULT ’’,
  ll_title VARCHAR(255) BINARY NOT NULL DEFAULT ’’,
  ...
  UNIQUE KEY (ll_from, ll_lang),
  ...
);
```

The `redirect` table stores page redirects. Note that, in theory, circular redirects are possible, as well as redirects to different namespaces, although none of these are recommended.

```
CREATE TABLE redirect (
  rd_from INT UNSIGNED NOT NULL DEFAULT '0',
  rd_namespace INT NOT NULL DEFAULT '0',
  rd_title VARCHAR(255) BINARY NOT NULL DEFAULT '',
  ...
  PRIMARY KEY rd_from (rd_from),
  ...
);
```

The `categorylinks` table stores categories of individual pages. Note that each category is a page itself, located in namespace 14 (`NS_CATEGORY`).

```
CREATE TABLE categorylinks (
  cl_from int unsigned NOT NULL default '0',
  cl_to varchar(255) binary NOT NULL default '',
  ...
  UNIQUE KEY cl_from (cl_from,cl_to),
  ...
);
```

There is a number of other important tables, most notably `revision` storing metadata for a given page revision (page, author, comment, date), and `text` storing text content for each revision.

In fact, all the "link" tables are redundant, since information contained in them can be recreated from the text of the latest revisions of pages. This, however, requires complex parsing, and thus, for the sake of efficiency, these auxiliary tables are used.

A.1.3 Wikipedia Database Dumps

For each language version of Wikipedia, there are several database dump files available, in different formats and containing different types of content. The file format is either compressed XML (Extensible Markup Language) suitable for import through the MediaWiki engine which powers Wikipedia, or compressed SQL (Structured Query Language) for direct import into the underlying database. Some files are compressed using *gzip* algorithm, others using *bzip2*. The largest dumps files are additionally available in the *7z* compression format.

The most important XML dump files are:

- `pages-meta-history.xml.{bz2,7z}` — *All pages with complete edit history*. These are the most complete dumps available, and the largest ones. For example, as of mid-February 2008, the English version compressed using bzip2 takes 133 GB and is approximately 20 times larger after decompression.
- `pages-meta-current.xml.bz2` — *All pages, current versions only*. These files include only the most recent revisions. Technical pages, such as discussion and user pages are included in these dumps.
- `pages-articles.xml.bz2` — *Articles, templates, image descriptions, and primary meta-pages*. These dumps contain the most recent revisions of articles. Wikipedia mirrors typically use these files.

SQL dumps store contents of individual tables in the database. Therefore, most (if not all) of these dumps can be recreated from the XML files, in the same way that "links" tables can be reconstructed from the texts of latest revisions of pages.

Some users need only one or two tables, and rebuilding them from the XML files can be extremely time-consuming. For that reason, some of the database tables are stored in the form of SQL dumps. Some of the dumps are listed below:

- `categorylinks.sql.gz` — *Wiki category membership link records*.
- `langlinks.sql.gz` — *Wiki interlanguage link records*.
- `page.sql.gz` — *Base per-page data (id, title, old restrictions, etc)*.
- `pagelinks.sql.gz` — *Wiki page-to-page link records*.
- `redirect.sql.gz` — *Redirect list*.

For the complete list of available dumps, go to: `http://download.wikimedia.org/`.

A.1.4 Import Alternatives

The canonical method of importing a language version of Wikipedia is installing the MediaWiki engine and running the *importDump* maintenance script with an XML dump as input, followed by optional rebuild of certain indexes. However, this method is painfully slow, especially for the larger language versions.

There are at least two reasons for the sluggishness of this procedure. First, the import function is written in PHP, an interpreted language, thus inevitably it executes less efficiently than a code generated by an optimizing C/C++ or Java compiler. Secondly, such import performs, among other things, parsing of the page text followed by updates of a number of auxiliary tables.

For many purposes, a tool extracting the crucial data from an XML dump and storing it as an SQL table is sufficient. One such practical tool, a Java program called *mwdumper*, reads an XML dump and writes `page`, `revision`, and `text` tables.

Finally, one can simply download the tables of interest instead of recreating them.

A.2 Database Import Process

A.2.1 Required and Optional Dumps

The minimum data requirements for our research consisted of `page`, `langlinks`, and `redirect` tables for each language version. Using this data, we could reconstruct the graph of interlanguage links between all the language versions: `page` tables provided nodes, `langlinks` tables provided links, and `redirect` tables were used to merge certain nodes.

However, in order to assign weights to the links, additional information about the nodes is needed. We have decided to calculate, for the interlanguage links that required such data, the number of common intrawiki links and common categories for the two endpoints, which requires `pagelinks` and `categorylinks` tables, respectively.

In this context, the number of "common" links for a given pair of articles (a, b) connected by an interlanguage link is understood as the number of intrawiki neighbors of a which are connected by interwiki links with neighbors of b (the definition is, of course, symmetric wrt. a and b).

A.2.2 Data Import and Preprocessing

We have written a script in the Python programming language that imports and preprocesses all the available dumps of a multi-language, MediaWiki-powered wiki, such as Wikipedia.

The script parses raw MySQL database dumps, therefore it does not require MySQL to be installed. The network of pages and links is stored in a database. The connected components of the network are identified and indexed. This way, each individual connected component may be processed independently, as a fast access to pages and links associated with the request component is guaranteed. The import process also filters out pages from namespaces other than the article namespace (0) or the category namespace (14). Similarly, links that are incident to the ignored pages are ignored as well.

A detailed design of the database holding the network shall be presented in Section A.3. We have created a software package hosted on the Google Code website[4], licensed

[4]See: `http://code.google.com/p/interwiki-analysis/`

under GNU General Public License version 3. The import script described above is part of this package.

A.3 Software Design and Implementation

A.3.1 Network Storage Model

Figure A.1 presents a Unified Modeling Language [38] class diagram of the idealized network storage. *Page* is the central class in the model, representing both articles and categories. It contains all the basic facts about a page, such as title, language, and namespace. A *Component* consists of one or more *Page*s. It is possible to derive several interesting facts about a given component: whether it is coherent or not, what is the namespace of the pages it contains, how many non-redirect pages does it contain.

A *LangLink* represents an interlanguage link between a pair of pages. It is possible to determine the *Component* in which a given *LangLink* resides. Next, *CategoryLink* and *PageLink* classes model the individual links of other types: a category link from X to Y says that article X belongs to category Y, while a page link from X to Y is the plain wiki link between two articles within a given language edition.

Next, *PagePosition* describes the position of a given *Page* in space. This information is used both by the "Spatial" approach described in Section 4.4, and in visualization. Finally, *PageMeaning* describes the cluster to which a given *Page* is assigned by a given *Approach*.

A.3.2 Database Design

The network storage design was optimized towards the specific needs of the analysis performed in this work. The following features were identified to be mandatory:

1. fast access to details of each node;
2. fast extraction of both incoming and outgoing links of any given node;
3. fast extraction of any given connected component.

The network data is stored in a PostgreSQL-backed database.

Each connected component is assigned a unique identifier (UUID, i.e., Universally Unique Identifier). The `network_comp` relation stores certain characteristics of every connected component having more than one node. The characteristics include the namespace of the nodes comprising the given component (0 for articles, 14 for categories – the codes are the same as in the MediaWiki engine), whether the component is coherent or not, and its size. While this relation is essentially redundant (all the values can be inferred from the remaining relations), it is maintained for an accelerated access

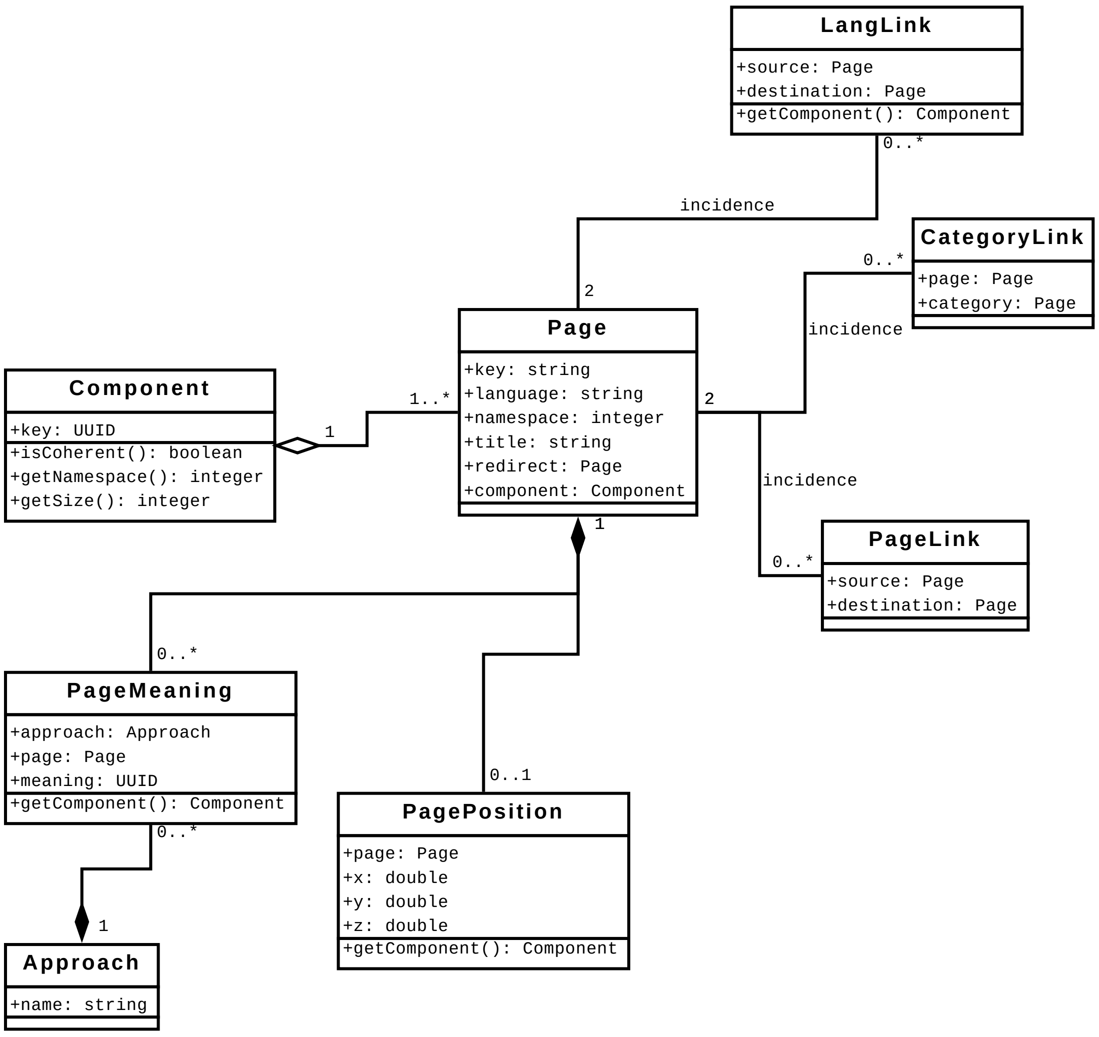


Figure A.1: UML class diagram of network storage.

to components of a given type. The components of size one are not recorded in this relation because of space considerations. The SQL schema for the `network_comp` is as follows:

```
CREATE TABLE network_comp (
    key varchar(36) NOT NULL PRIMARY KEY,
    namespace INTEGER NOT NULL,
    coherent BOOLEAN NULL,
    size INTEGER NULL
);
```

Each page (an article or a category) in any of the language editions has a corresponding record in the `network_page` relation. The record holds page's identifier, language and title, namespace (article or category), and informs whether the page is a redirect to another one. For the sake of optimization it also contains (redundantly) the identifier of the connected component.

```
CREATE TABLE network_page (
    key VARCHAR(32) NOT NULL PRIMARY KEY,
    lang VARCHAR(16) NOT NULL,
    namespace INTEGER NOT NULL,
    title varchar(1024) NOT NULL,
    redirect_id VARCHAR(32) NULL,
    comp_id VARCHAR(36) NULL REFERENCES network_comp (key)
);
```

Each interlanguage link between a pair of articles or a pair of categories has a corresponding record in the `network_langlink` relation. Again, the identifier of the connected component to which the given ILL belongs is redundantly stored:

```
CREATE TABLE network_langlink (
    id SERIAL NOT NULL PRIMARY KEY,
    src_id VARCHAR(32) NOT NULL REFERENCES network_page (key),
    dst_id VARCHAR(32) NOT NULL REFERENCES network_page (key),
    comp_id VARCHAR(36) NULL REFERENCES network_comp (key)
);
```

Links between articles and their categories, as well as links between articles within a given language edition, are stored in the `network_categorylink` and `network_pagelink` relations, respectively. The primary purpose of storing the above links is to assign weights to the interlanguage links based on the number of common categories and outgoing links of a pair of articles.

```
CREATE TABLE network_categorylink (
    id SERIAL NOT NULL PRIMARY KEY,
    page_id VARCHAR(32) NOT NULL REFERENCES network_page (key),
    category_id VARCHAR(32) NOT NULL REFERENCES network_page (key)
);

CREATE TABLE network_pagelink (
    id SERIAL NOT NULL PRIMARY KEY,
    src_id VARCHAR(32) NOT NULL REFERENCES network_page (key),
    dst_id VARCHAR(32) NOT NULL REFERENCES network_page (key)
);
```

Page positions in space are stored in the `network_pageposition` relation. Each page has a corresponding record in the relation, containing the x, y and z coordinates of the location in $\mathbb{R}^3$. A reference to the component containing the page is redundantly stored as well.

```
CREATE TABLE network_pageposition (
    id SERIAL NOT NULL PRIMARY KEY,
    page_id VARCHAR(32) NOT NULL REFERENCES network_page (key),
    x DOUBLE PRECISION NULL,
    y DOUBLE PRECISION NULL,
    z DOUBLE PRECISION NULL,
    comp_id VARCHAR(36) NOT NULL REFERENCES network_comp (key)
);
```

Finally, the `network_pagemeaning` relation stores the partitions generated by the approaches described in Chapter 4. A record in the relation describes the location of an individual page within a partition generated by a given approach. Such record contains the identifier of the approach, the identifier of the meaning to which the page is assigned, and – as usual – the identifier of the component to which the page belongs.

```
CREATE TABLE network_pagemeaning (
    id SERIAL NOT NULL PRIMARY KEY,
    auth VARCHAR(30) NOT NULL,
    page_id VARCHAR(32) NOT NULL REFERENCES network_page (key),
    meaning VARCHAR(36) NOT NULL,
    comp_id VARCHAR(36) NOT NULL REFERENCES network_comp (key)
);
```

As we have already pointed out, most of the relations have a redundant reference to the component containing a given entity. Thanks to this, assuming that in each of the case there is a database index on the `comp_id` column, it is feasible to fetch a single component and all the entities associated with it. This way, one may process a network component-by-component, dramatically improving the performance.

Another optimization-driven redundancy is present in the `network_comp` relation, where component characteristics are redundantly stored. This way it is possible to efficiently iterate over a set of components with given properties. Again, this optimization yields a dramatic reduction of the processing time.

A.3.3 Software Package

We have developed a software package containing the database import procedure and implementations of all the approaches presented in Chapter 4. The package is hosted on the Google Code website: `http://code.google.com/p/interwiki-analysis/` and is available under the GNU General Public License version 3.

The code has been written in the Python programming language. We have used the NumPy and SciPy scientific libraries to perform the Conjugate Gradient routine referenced in the "Spatial" approach (Section 4.4). We have chosen PostgreSQL as the database backend, and the psycopg package provides the Python-PostgreSQL connectivity.

A proof-of-concept website has been created, where incoherent components and selected analysis results may be explored. The website is written using the Django framework (a Python library). The website's address is: `http://wikitools.icm.edu.pl/`

Appendix B

Glossary and Symbols

B.1 Glossary

The following is a list of recurring terms used throughout this thesis:

clique A set of vertices in a graph such that every pair of vertices from the set is connected by an edge.

cluster An element of a partition (a subset of vertices).

coherent graph A graph in which each connected component contains at most one vertex of any given color.

complete graph A coherent graph in which every connected component is a clique.

connected component A set of vertices in a graph such that for each pair of vertices in the set there is a path (a sequence of edges) that connect the two vertices.

cut In a given graph, any set of edges such that their removal renders the graph coherent.

edge A connection between two vertices.

graph A set of vertices and edges. In the context of this thesis, unless stated otherwise, a graph is assumed to be vertex-colored (a color is assigned to each vertex), weighted (a positive real number is assigned to each edge), and undirected (each edge simply connects two vertices v and w without specifying whether it is a connection from v to w, or from w to v).

ILL See: interlanguage link.

interlanguage link In Wikipedia, a link from a page in one language edition to a corresponding page in another language edition.

link See: edge.

network See: graph.

node See: vertex.

partition In a given graph, partition of the set of vertices into subsets (clusters) in such a way that no two vertices of the same color are in the same subset.

vertex An elementary building block of a graph. A graph consists of vertices connected by edges. A vertex may have additional properties, such as color.

B.2 Symbols

The following is a list of recurring symbols used throughout this thesis:

E all the edges of a graph.

e typical symbol for an individual edge.

F typical symbol for a cut.

$f|_X$ if f is a function and X is a set, then $f|_X$ is a new function with the domain restricted to $dom(f) \cap X$, and equal to f everywhere.

$G = \langle V, E, \kappa, \mu \rangle$ A graph consisting of vertices V and edges E, with vertex colors κ and edge weights μ.

$G|_\pi$ if G is a graph and π is a partition of this graph, then $G|_\pi$ is a truncated graph, that is, a graph G with all the edges incoherent with π removed.

V all the vertices of a graph.

v, w typical symbol for individual vertices.

κ vertex colors. It is a function that, for a given vertex v, returns its color $\kappa(v)$.

μ edge weights. It is a function that, for a given edge e, returns its weight $\mu(e)$.

Π partition of the set of vertices V into disjoint subsets covering V.

π partition of the set of vertices V into disjoint subsets covering V. For a given v, $\pi(v)$ returns the subset of V containing v.

Acknowledgements

First of all, I would like to thank my wife Duygu for her tremendous support and motivation, without which this thesis would never have been completed.

Next, I would like to thank Prof. Marek Niezgódka for his supervision of this thesis and creating a comfortable environment for research. I would also like to thank my colleagues at the Interdisciplinary Centre for Mathematical and Computational Modelling, whose to-the-point remarks and fruitful discussions were invaluable: Piotr Bała, Dominik Batorski, Magdalena Gruziel, Jarosław Kalinowski, Kerstin Kantiem, Michał Łopuszyński, Aleksander Nowiński, Krzysztof Nowiński, Michał Politowski, Franciszek Rakowski, Tomasz Rosiek, Wojciech Sylwestrzak, and Anna Trykozko. Finally, I would like to thank members of the Wikipedia and Wikia communities: Masti and Tor for their cooperation and fruitful discussion.

This research was possible because of a wide range of free and open source software, including: GNU bc, Django framework, Eclipse IDE, Fedora GNU/Linux distribution, Firefox, gnuplot, MediaWiki, MySQL, NetBeans, OpenJDK, PostgreSQL, Python, GNU R, Referencer, Subversion, TeX Live, Texmaker, GNU Textutils, Ubuntu GNU/Linux distribution, and Vim. I deeply appreciate the effort of countless authors of these excellent products.

Bibliography

[1] Adafre, S.F., de Rijke, M.: Discovering missing links in Wikipedia. In: *Proceedings of the 3rd international workshop on Link discovery*. (2005) 90–97

[2] Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. *Reviews of Modern Physics* **74** (2002) 47–97

[3] Almeida, R.B., Mozafari, B., Cho, J.: On the Evolution of Wikipedia. In: *Proceedings of the International Conference on Weblogs and Social Media*. (2007)

[4] Anthonisse, J.M.: The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam (1971)

[5] Auer, S., Bizer, C., Lehmann, J., Kobilarov, G., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: *6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)*. (2007) 715–728

[6] Barabási, A.L., Albert, R.: Emergence of scaling in random networks. *Science* **286** (1999) 509–512

[7] Barabási, A.L., Ravasz, E., Vicsek, T.: Deterministic Scale-Free Networks. *Physica A* **299** (2001) 3

[8] Barak, A.: *Psychological Aspects of Cyberspace: Theory, Research, Applications*. Cambridge University Press, New York, NY, USA (2008)

[9] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* (2001)

[10] Berners-Lee, T.: Tim Berners-Lee on the next Web. TED Talks (2009)

[11] Bollobás, B.: *Random Graphs*. Cambridge University Press (2001)

[12] Bomze, I., Budinich, M., Pardalos, P.M., Pelillo, M.: The maximum clique problem. In Du, D.Z., Pardalos, P.M., eds.: *Handbook of Combinatorial Optimization (Supplement Volume A)*. Volume 4. Kluwer Academic Publishers, Boston, MA (1999)

[13] Bondy, J.A., Murty, U.S.R.: *Graph theory with applications*. Elsevier Science (1976)

[14] Boudet, V., Rastello, F., Robert, Y.: Alignment and distribution is NOT (always) NP-hard. In: *Proceedings of the International Conference on Parallel and Distributed Systems*. (1998)

[15] Brandes, U.: A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology* **25** (2001) 163–177

[16] Brandes, U., Lerner, J.: Visual analysis of controversy in user-generated encyclopedias. *Information Visualization* **7** (2008) 34–48

[17] Bryant, S.L., Forte, A., Bruckman, A.: Becoming Wikipedian: transformation of participation in a collaborative online encyclopedia. In: *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*. (2005) 1–10

[18] Bunde, A., Havlin, S.: Power-law persistence in the atmosphere and in the oceans. *Physica A* **314** (2002) 15–24

[19] Callaway, D.S., Newman, M.E.J., Strogatz, S.H., Watts, D.J.: Network Robustness and Fragility: Percolation on Random Graphs. *Phys. Rev. Lett.* **85** (2000) 5468–5471

[20] Capocci, A., Servedio, V.D.P., Colaiori, F., Buriol, L.S., Donato, D., Leonardi, S., Caldarelli, G.: Preferential attachment in the growth of social networks: the case of Wikipedia (2006)

[21] Capocci, A., Servedio, V.D.P., Caldarelli, G., Colaiori, F.: Detecting communities in large networks. *Physica A* **352** (2005) 669–676

[22] Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70** (2004) 066111

[23] Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-law distributions in empirical data (2007)

[24] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms, Second Edition*. The MIT Press (2001)

[25] Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The Complexity of Multiterminal Cuts. *SIAM J. Comput.* **23** (1994) 864–894

[26] Danon, L., Duch, J., Diaz-Guilera, A., Arenas, A.: Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment* (2005) P09008–09008

[27] Dorogovtsev, S.N., Goltsev, A.V., Mendes, J.F.F.: Pseudofractal scale-free web. *Phys. Rev. E* **65** (2002) 066122

[28] Dorogovtsev, S.N., Goltsev, A.V., Mendes, J.F.F.: k-core organization of complex networks. *Physical Review Letters* **96** (2006) 040601

[29] Dorogovtsev, S.N., Mendes, J.F.F.: Scaling properties of scale-free evolving networks: Continuous approach. *Physical Review E* **63** (2001) 056125

[30] Dorogovtsev, S.N., Mendes, J.F.F., Oliveira, J.G.: Frequency of occurrence of numbers in the World Wide Web. *Physica A* **360** (2006) 548

[31] Duch, J., Arenas, A.: Effect of random failures on traffic in complex networks. *Proc. SPIE* **6601** (2007)

[32] Eades, P.: A Heuristic for Graph Drawing. *Congressus Numerantium* **42** (1984) 149–160

[33] Eastman, R., Eastman, E.: Iquito Syntax. In Elson, B.F., ed.: *Studies in Peruvian Indian languages: I*. SIL of the University of Oklahoma (1963) 145–192

[34] Ebel, H., Mielsch, L.I., Bornholdt, S.: Scale-free topology of e-mail networks. *Physical Review E* **66** (2002) 035103

[35] Erdmann, M., Nakayama, K., Hara, T., Nishio, S.: Lecture Notes in Computer Science. In Haritsa, J.R., Ramamohanarao, K., Pudi, V., eds.: *DASFAA*. Volume 4947., Springer (2008) 380–392

[36] Erdős, P., Rényi, A.: On random graphs, I. *Publicationes Mathematicae* **6** (1959) 290–297

[37] Erdős, P., Rényi, A.: On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* **5** (1960) 17–61

[38] Eriksson, H.E., Penker, M.: *UML toolkit*. John Wiley & Sons, Inc., New York, NY, USA (1998)

[39] Everett, D.L.: Cultural Constraints on Grammar and Cognition in Pirahã. *Current Anthropology* **46** (2005)

[40] Flake, G., Lawrence, S., Giles, C.L.: Efficient Identification of Web Communities. In: *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA (2000) 150–160

[41] Ford, L.R., Fulkerson, D.R.: *Flows in Networks*. Princeton University Press, Princeton, NJ (1962)

[42] Frank, M.C., Everett, D.L., Fedorenko, E., Gibson, E.: Number as a cognitive technology: Evidence from Pirahã language and cognition. *Cognition* **108** (2008) 819–824

[43] Freeman, L.C.: A Set of Measures of Centrality Based on Betweenness. *Sociometry* **40** (1977) 35–41

[44] Freeman, L.C.: Centrality in social networks: Conceptual clarification. *Social Networks* **1** (1979) 215–239

[45] Fruchterman, T.M.J., Reingold, E.M.: Graph Drawing by Force-directed Placement. *Software - Practice and Experience* **21** (1991) 1129–1164

[46] Garey, M.R., Johnson, D.S., Stockmeyer, L.J.: Some Simplified NP-Complete Graph Problems. *Theor. Comput. Sci.* **1** (1976) 237–267

[47] Garey, M.R., Johnson, D.S.: *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA (1990)

[48] Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. *Proc. Natl. Acad. Sci. U.S.A.* **99** (2002) 7821–6

[49] Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional (1989)

[50] Gulbahce, N., Lehmann, S.: The art of community detection. *BioEssays* **30** (2008) 934–938

[51] Hansen, S., Berente, N., Lyytinen, K.: Wikipedia, Critical Social Theory, and the Possibility of Rational Discourse. *The Information Society* **25** (2009) 38–59

[52] Harary, F., Hage, P.: Eccentricity and centrality in networks. *Social Networks* **17** (1995) 57–63

[53] He, G., Liu, J., Zhao, C.: Approximation algorithms for some graph partitioning problems. *Journal of Graph Algorithms and Applications* **4** (2000) 1–11

[54] Hill, T., Lundgren, A., Fredriksson, R., Schiöth, H.B.: Genetic algorithm for large-scale maximum parsimony phylogenetic analysis of proteins. *Biochim Biophys Acta* **1725** (2005) 19–29

[55] Holland, J.H.: *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA (1992)

[56] Hong, B.H., Lee, K.E., Lee, J.W.: Power Law in Firms Bankruptcy. *Physics Letter A* **361** (2007) 6

[57] Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. *Inf. Process. Lett.* **31** (1989) 7–15

[58] Karp, R.M.: Reducibility among combinatorial problems. In Miller, R.E., Thatcher, J.W., eds.: *Complexity of Computer Computations*. Plenum Press (1972) 85–103

[59] Kopf, R., Ruhe, G.: A computational study of the weighted independent set problem for general graphs. *Foundations of Control Engineering* **12** (1987) 167–180

[60] Korte, B., Lovász, L., Schrader, R.: *Greedoids*. Springer-Verlag (1991)

[61] Laherrère, J.H.: Distributions de type fractal parabolique dans la nature. *Comptes Rendus de l'Acadèmie des Sciences* **322** (1996) 535–541

[62] Lewis, P.O.: A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data. *Mol Biol Evol* **15** (1998) 277–283

[63] Li, J., Chen, M.: Index Domain Alignment: Minimizing Cost of Cross-reference between Distributed Arrays. In: *Proceedings of the third Symposium on Frontiers of Massively Parallel Computation*. (1990) 424–433

[64] Luccio, F., Sami, M.: On the Decomposition of Networks in Minimally Interconnected Subnetworks. *IEEE Transactions on Circuit Theory* **16** (1969) 184–188

[65] Magnus, P.D.: On Trusting WIKIPEDIA. *Episteme* **6** (2009) 74–90

[66] Mantegna, R.N., Buldyrev, S.V., Goldberger, A.L., Havlin, S., Peng, C.K., Simons, M., Stanley, H.E.: Linguistic Features of Noncoding DNA Sequences. *Phys. Rev. Lett.* **73** (1994) 3169–3172

[67] Marsili, M., Zhang, Y.C.: Interacting Individuals Leading to Zipf's Law. *Phys. Rev. Lett.* **80** (1998) 2741–2744

[68] Milgram, S.: Behavioral study of obedience. *Journal of Abnormal and Social Psychology* **67** (1963) 371–378

[69] Mitchell, M.: *An Introduction to Genetic Algorithms*. The MIT Press (1998)

[70] Miyazima, S., Lee, Y., Nagamine, T., Miyajima, H.: Power-law Distribution of Family Names in Japanese Societies. *Physica A* **278** (2000) 282 – 288

[71] Newman, M.E.J.: Scientific collaboration networks. I. Network construction and fundamental results. *Phys. Rev. E* **64** (2001) 016131

[72] Newman, M.E.J.: Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Phys. Rev. E* **64** (2001) 016132

[73] Newman, M.E.J.: Detecting community structure in networks. *European Physical Journal* **B 38** (2004) 321–330

[74] Newman, M.E.J.: Fast algorithm for detecting community structure in networks. *Physical Review E* **69** (2004) 066133

[75] Newman, M.E.J.: Power laws, Pareto distributions and Zipf's law. *Contemporary Physics* **46** (2005) 323

[76] Newman, M.E.J.: Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* **103** (2006) 8577–8582

[77] Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. *Physical Review E* **69** (2004) 026113

[78] Newman, M.E.J., Park, J.: Why social networks are different from other types of networks. *Physical review. E* **68** (2003) 036122

[79] Nunes, S., Ribeiro, C., David, G.: WikiChanges - Exposing Wikipedia Revision Activity. In: *WikiSym'08: Proceedings of the 2008 international symposium on Wikis.* (2008)

[80] Oreg, S., Nov, O.: Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values. *Comput. Hum. Behav.* **24** (2008) 2055–2073

[81] Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435** (2005) 814–818

[82] Papadimitriou, C.H., Steiglitz, K.: *Combinatorial optimization: algorithms and complexity*. Prentice-Hall (1982)

[83] Papadimitriou, C.M.: *Computational complexity*. Addison-Wesley, Reading, Massachusetts (1994)

[84] Press, W.H.: *Numerical recipes: the art of scientific computing*. 3 edn. Cambridge University Press (2007)

[85] Pushkin, D.O., Aref, H.: Bank mergers as scale-free coagulation. *Physica A* **336** (2004) 571–584

[86] R. Albert, H. Jeong, A.L.B.: Error and attack tolerance of complex networks. *Nature* **406** (2000) 378–482

[87] Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. *Proc Natl Acad Sci USA* **101** (2004) 2658–2663

[88] Ravasz, E., Barabási, A.L.: Hierarchical organization in complex networks. *Phys. Rev. E* **67** (2003) 026112

[89] Sabidussi, G.: The centrality index of a graph. *Psychometrika* **31** (1966) 581–603

[90] Saito, K., Yamada, T., Kazama, K.: Extracting Communities from Complex Networks by the k-Dense Method. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **E91-A** (2008) 3304–3311

[91] Sanger, L.M.: The Fate of Expertise after WIKIPEDIA. *Episteme* **6** (2009) 52–73

[92] Schroer, J., Hertel, G.: Voluntary Engagement in an Open Web-Based Encyclopedia: Wikipedians and Why They Do It. *Media Psychology* **12** (2009) 96–120

[93] Seidman, S.: Network structure and minimum degree. *Social Networks* **5** (1983) 269–287

[94] Shimbel, A.: Structural parameters of communication networks. *Bulletin of Mathematical Biology* **15** (1953) 501–507

[95] Stone, H.S.: Multiprocessor Scheduling with the Aid of Network Flow Algorithms. *IEEE Trans. Softw. Eng.* **3** (1977) 85–93

[96] Suh, B., Chi, E.H., Pendleton, B.A., Kittur, A.: Us vs. Them: Understanding Social Dynamics in Wikipedia with Revert Graph Visualizations. In: *IEEE Symposium on Visual Analytics Science and Technology*. (2007) 163–170

[97] Tollefsen, D.P.: WIKIPEDIA and the Epistemology of Testimony. *Episteme* **6** (2009) 8–24

[98] Travers, J., Milgram, S.: An Experimental Study of the Small World Problem. *Sociometry* **32** (1969) 425–443

[99] Tyers, F.M., Pienaar, J.A.: Extracting bilingual word pairs from Wikipedia. In: *Proceedings of the SALTMIL Workshop at Language Resources and Evaluation Conference, LREC08*. (2008) 19–22

[100] van Batenburg, F.H., Gultyaev, A.P., Pleij, C.W.: An APL-programmed genetic algorithm for the prediction of RNA secondary structure. *J Theor Biol* **174** (1995) 269–280

[101] Vose, M.D.: *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, USA (1998)

[102] Wagner, K., Wechsung, G.: *Computational Complexity*. Springer (2001)

[103] Wasserman, S., Faust, K.: *Social Network Analysis: Methods and Applications*. Cambridge University Press (1994)

[104] Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. *Nature* **393** (1998) 440–2

[105] Wierzbicka, A.: *Semantics, culture, and cognition : universal human concepts in culture-specific configurations*. Oxford University Press (1992)

[106] Wierzbicka, A.: *Semantics: Primes and Universals*. Oxford University Press (1996)

[107] Wierzbicka, A.: *Emotions Across Languages and Cultures: Diversity and universals*. Cambridge University Press (1999)

[108] Willett, P.: Genetic algorithms in molecular recognition and design. *Trends in Biotechnology* **13** (1995) 516–521

[109] Wray, B.K.: The Epistemic Cultures of Science and WIKIPEDIA: A Comparison. *Episteme* **6** (2009) 38–51

[110] Yoon, J., Blumer, A., Lee, K.: An algorithm for modularity analysis of directed and weighted biological networks based on edge-betweenness centrality. *Bioinformatics* **22** (2006) 3106–8

[111] Zlatic, V., Bozicevic, M., Stefancic, H., Domazet, M.: Wikipedias: Collaborative web-based encyclopedias as complex networks. *Physical Review E* **74** (2006) 016115