2012

Characterizing Conflict in Wikipedia

Nathaniel Miller
Macalester College, nmiller@alumni.macalester.edu

Characterizing Conflict in Wikipedia

Submitted to the Department of Mathematics, Statistics and Computer Science in partial fulfillment of the requirements for the degree of Bachelor of Arts

Nathaniel Miller

Advisor: Prof. Shilad Sen
Second Reader: Prof. Andrew Beveridge
Third Reader: Prof. Susan Fox

Macalester College

May 1, 2012

1

Abstract

Wikipedia serves as the Internet's most widely viewed reference. In order to ensure its success, editors who create and maintain articles must resolve conflicts over appropriate article content. Previous research has measured Wikipedia conflict at two levels: single articles and categories of pages. I observe conflicts within small groups of articles, identifying their frequency, size, and intensity. Additionally, I identify individual conflicts spanning multiple articles and effects of conflict upon users' editing habits. I analyze cross-article conflict in three stages. First, I cluster a group of 1.4 million Wikipedia articles. Next, I find individual user conflicts within each article cluster using a list of reverts. Finally, I characterize the individual conflicts and analyze population statistics. While most conflicts are low-intensity and take place in a single article, high-intensity conflicts frequently span multiple articles.

Contents

2

Introduction

Wikipedia serves as the Internet's most widely viewed reference, garnering more than 7.7 billion page views per month.[1] Remarkably, volunteers have created and maintained Wikipedia's 3.89 million articles.[2] Beyond the quantity of its articles, readers also value Wikipedia for its quality, which is comparable to Encyclopedia Britannica's [5]. Wikipedia volunteers perform about 3.6 million edits to the encyclopedia every month. To maintain article quality, users who have never met must collaborate - even if they disagree passionately. These disagreements can be beneficial, helping mold policies and best practices for the encylopedia. However, conflict can also be detrimental, discouraging volunteers and leading them to stop contributing. For these reasons, conflict can shape the Wikipedia volunteer community and the articles those volunteers maintain.

Several researchers have studied conflict in Wikipedia, generally through two separate lenses. One popular method of viewing conflict has been at the individual article level. For example, Brandes and Lerner apply a technique for visualizing conflict to single articles [2]. Others have studied which categories of Wikipedia pages conflict is concentrated in [7]. However, very little research has been conducted on the structure of individual conflicts that have spread beyond a single article.

I attempt to address that gap in the research by investigating the presence of conflicts that span related articles. This paper investigates the char-

[1]http://stats.wikimedia.org/EN/TablesPageViewsMonthly.htm

[2]http://stats.wikimedia.org/EN/SummaryEN.htm

acteristics of conflict within Wikipedia at a low level by observing individual differences of opinion. At the same time, the analysis extends over a large number of Wikipedia articles and can also be viewed at a high level as describing Wikipedia as a whole. I make conclusions about the characteristics of conflict in Wikipedia, including the size, frequency, and intensity of conflicts, as well as their effects on editing.

There are two primary obstacles to analyzing conflict on Wikipedia: the amount of data that must be analyzed and finding an appropriate method of identifying users carrying on a conflict and the articles involved. Over the past year there have been almost 50 million edits to English Wikipedia articles, if we extrapolate from the statistics above. Recent technological advances such as Hadoop MapReduce have made this barrier easier to overcome, though it is still a substantial obstacle. Identifying groups of users who carry on a conflict in more than one article is also difficult. In order for the process to be scalable, we need a method of identifying other articles that are good candidates for the conflict to spread to, as well as a way of determining that the conflict really is the same on both pages.

I conducted my analysis in three stages:

1. **Article Clustering**:
   I studied conflict within groups of articles, or article *clusters*. These clusters were created based upon random walks on a graph of links between the articles.

2. **User Clustering**:
   I studied conflicts among users by examining *reverts*, or reversed edits between two users, within each article cluster. Each conflict was a connected component of reverts. I clustered the involved users into two sides by approximating the maximum cut for the component.

3. **Conflict Characterization**:
   Finally, I described conflict in Wikipedia by observing how conflict affected users' contributions to Wikipedia as well as the frequency with which users carried on conflicts in multiple articles.

This paper is organized as follows: in Section 3 I describe prior research on social networks within the Internet and Wikipedia, methods of clustering graphs, and tools for working with large datasets. In Section 4, I describe how I clustered both directed and undirected article graphs using random walks as well as the results of these clusterings. In Section 5, I describe how I clustered users using maximum cut approximation and discuss the results of both the directed and undirected user clusterings. In Section 6, I describe how I used this information to characterize conflict in Wikipedia using bytes added and deleted by users as well as the frequency with with users carry on a conflict in multiple articles and discuss the results of my analyses.

3

Background and Related Work

This project interacts with a diverse group of topics in computer science, mathematics, and psychology. The article and user clustering stages involve both mathematics, which is necessary to choose appropriate methods of clustering, and computer science, which allows us to deal with these problems on a large scale. The third stage, characterizing conflict, involves social networks and the Internet, which is an intersection of computer science and psychology. Research on Wikipedia is also relevant. With this in mind, this section is divided into five subsections: one about social networks on the Internet, one about group psychology, one on social interactions within Wikipedia, one on clustering graphs, and one about processing large datasets.

3.1 Social Networks on the Internet

There are many websites dedicated to public or semi-public communication between users, examples of which include Facebook, Reddit, Slashdot, StackExchange, and Twitter. Research on these websites comes in many different forms, but many view them as graphs, with users as vertices and interactions between users as edges. Some of this research examines users' popularity, unpopularity, or importance to a website's community. There is also research into the theoretical implications of the differences between the ways websites represent relationships between their users for research methods.

Kunegis, Lommatzsch, and Bauckhage identify popular, unpopular, and central users on the website Slashdot utilizing information provided by the network of users' friend and foe relationships [9]. They are able to easily identify popular and unpopular users based upon their number of friends and foes. However, they find that central users, those who are have more connections and are closer to other users on average, are not necessarily the ones with most friends or foes and that metrics specifically designed to measure centrality, like PageRank, are better for this purpose. This suggests that while common graph-based metrics may provide some insight into conflict, specialized metrics might better capture conflict's characteristics within Wikipedia and its affect upon users' contributions.

Leskovec, Huttenlocher, and Kleinberg investigate how social networks with signed relationships, such as foes and friends on Slashdot, as opposed to unsigned relationships, such as friends on Facebook, require different investigatory techniques [11]. They find that signed networks are better analyzed by theories of relationships that include social hierarchy, essentially the idea that some users may be more popular than others. Unsigned networks, on the other hand, are better analyzed by theories of relationships that do not include social hierarchy. Unlike Leskovec, Huttenlocher, and Kleinberg, our data had only positive edges or only negative edges, so techniques that apply to unsigned relationships are appropriate for us.

Brandes and Lerner introduce a graph-like visualization of conflict between multiple parties [1]. The visualization is best for conflicts between two parties and is difficult to use for more groups. Brandes and Lerner apply their visualization to Wikipedia [2]. The data in each graph is based upon the similarity of contributions, including both additions and deletions, by editors of an article. Editors are represented as vertices, which are located near other users they tend to agree with. Every graph uses data from only one article, but they demonstrate that there is no theoretical barrier to using one graph for multiple articles. Brandes and Lerner find that certain articles are edited more intensely at certain times than others [3]. Specifically, they find that edits of news events spike immediately after the event. Additionally, they find that certain pairs of articles have more common editors than others. Their research sparked our interest in looking at conflict in larger groups of articles and we considered using their algorithm in the user clustering stage. We decided not to use it because of the complexity of implementation and the difficulty of using its output to create user clusters.

Kittur, Chi, and Suh analyze the distribution of conflict throughout Wikipedia, using data from July, 2006 [7]. They find that certain types of articles are more likely to have conflict, including those about religion and philosophy. While we study conflict within groups of articles, this research is focused on controversy within article categories.

On an even more general level, Zlatic, Bozicevic, Stefancic, and Domazet look at the basic network structures of Wikipedias in different languages and find they are similar in several ways [18]. For example, they find the degree distributions of intra-Wikipedia links, including in-degree, out-degree, and undirected degree, follow very similar power laws in the different languages. This suggests intra-Wikipedia links contain important information about Wikipedia's structure.

3.2 Group Psychology

When studying conflict between groups of users, the reasons for users' actions and their impacts upon other users must be considered. Therefore, our research must be informed by research on group psychology.

Van Alstyne and Brynjolfsson theorize that people will choose to socialize with those who most closely share their ideas given the choice [17]. They further suggest that the internet will enable people to significantly decrease their interactions with people they disagree with.

Sunstein reviews several fascinating psychology experiments and concludes that a group gains more confidence in its collective opinion after discussion [16]. The experiments demonstrate that people's judgments are influenced by the judgments of those around them, even when those judgments are contradicted by readily available evidence.

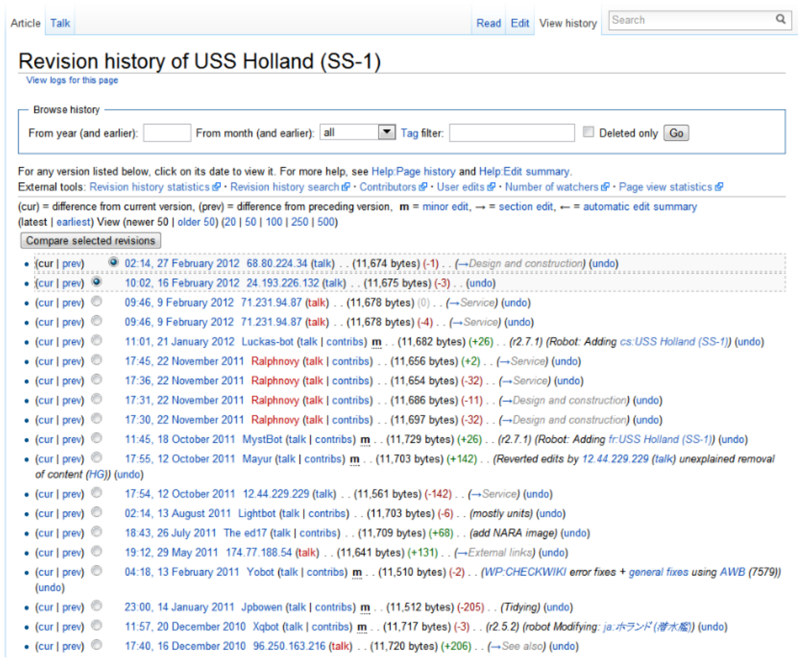
Understanding the implications of this research requires additional background on Wikipedia and therefore they will be discussed below, in Section 3.3.

3.3 Wikipedia

Wikipedia is a volunteer-maintained encyclopedia that is publicly available online. Contributions to a Wikipedia page, or *article*, are also called *edits*. Information about each edit, including the name of the user who made the edit and the time at which the edit was made, can be viewed by clicking on the *View History* tab near the top of any page, as seen in Figure 3.1. Each page also has an associated *Discussion* page, also referred to as *Talk* pages, on which users may discuss issues affecting the article. On Wikipedia, a *namespace* is a section of the website that contains pages of a certain type: the user namespace contains user pages and the project namespace contains pages about Wikipedia policies, for example. My research focuses on the *main* namespace, which contains only articles. Research on Wikipedia has examined a variety of different topics, including the purpose of volunteer contributions, the structure of Wikipedia discussions, and interactions between Wikipedia volunteers both within Discussion pages and within articles.

Kittur, Suh, Pendleton, and Chi investigate the purposes of edits to Wikipedia over time [8]. They find that an increasing percentage of edits are directed to non-article pages, like Discussion pages, as opposed to the articles themselves. The authors suggest this indicates coordination costs, or the difficulties of overcoming obstacles to successful collaboration, are rising for Wikipedia. Conflict between users is another component of coordination costs in Wikipedia.

Laniado et al. investigate Discussion (also called Talk) pages on Wikipedia by introducing a new tree structure to model them [10]. They find several interesting results about Wikipedia's community structure. They show talk pages are used differently across different categories of article. According to their research, discussion pages about sports articles tend to draw fewer users and be less in-depth than discussion pages about philosophy articles. One primary method of conflict resolution is through discussion, so differing uses of the discussion pages suggest that conflict is distributed unevenly and may be dealt with differently across articles. They also find evidence there are two primary groups of talk page editors: those who respond to infrequent talk page users and those who respond to the most frequently responded-to users. This research may be an example of Van Alstyne and Brynjolfsson's theories from Section 3.3 in action.


Article Talk Read Edit View history Search

Revision history of USS Holland (SS-1)

View logs for this page

Browse history

From year (and earlier): From month (and earlier): all Tag filter: Deleted only Go

For any version listed below, click on its date to view it. For more help, see Help:Page history and Help:Edit summary.
External tools: Revision history statistics · Revision history search · Contributors · User edits · Number of watchers · Page view statistics

(cur) = difference from current version, (prev) = difference from preceding version, **m** = minor edit, → = section edit, ← = automatic edit summary
(latest | earliest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

Compare selected revisions

- (cur | prev) 02:14, 27 February 2012 68.80.224.34 (talk) . . (11,674 bytes) (-1) . . *(→Design and construction)* (undo)
- (cur | prev) 10:02, 16 February 2012 24.193.226.132 (talk) . . (11,675 bytes) (-3) . . (undo)
- (cur | prev) 09:46, 9 February 2012 71.231.94.87 (talk) . . (11,678 bytes) (0) . . *(→Service)* (undo)
- (cur | prev) 09:46, 9 February 2012 71.231.94.87 (talk) . . (11,678 bytes) (-4) . . *(→Service)* (undo)
- (cur | prev) 11:01, 21 January 2012 Luckas-bot (talk | contribs) **m** . . (11,682 bytes) (+26) . . *(r2.7.1) (Robot: Adding cs:USS Holland (SS-1))* (undo)
- (cur | prev) 17:45, 22 November 2011 Ralphnovy (talk | contribs) . . (11,656 bytes) (+2) . . *(→Service)* (undo)
- (cur | prev) 17:36, 22 November 2011 Ralphnovy (talk | contribs) . . (11,654 bytes) (-32) . . *(→Service)* (undo)
- (cur | prev) 17:31, 22 November 2011 Ralphnovy (talk | contribs) . . (11,686 bytes) (-11) . . *(→Design and construction)* (undo)
- (cur | prev) 17:30, 22 November 2011 Ralphnovy (talk | contribs) . . (11,697 bytes) (-32) . . *(→Design and construction)* (undo)
- (cur | prev) 11:45, 18 October 2011 MystBot (talk | contribs) **m** . . (11,729 bytes) (+26) . . *(r2.7.1) (Robot: Adding fr:USS Holland (SS-1))* (undo)
- (cur | prev) 17:55, 12 October 2011 Mayur (talk | contribs) **m** . . (11,703 bytes) (+142) . . *(Reverted edits by 12.44.229.229 (talk) unexplained removal of content (HG))* (undo)
- (cur | prev) 17:54, 12 October 2011 12.44.229.229 (talk) . . (11,561 bytes) (-142) . . *(→Service)* (undo)
- (cur | prev) 02:14, 13 August 2011 Lightbot (talk | contribs) . . (11,703 bytes) (-6) . . *(mostly units)* (undo)
- (cur | prev) 18:43, 26 July 2011 The ed17 (talk | contribs) . . (11,709 bytes) (+68) . . *(add NARA image)* (undo)
- (cur | prev) 19:12, 29 May 2011 174.77.188.54 (talk) . . (11,641 bytes) (+131) . . *(→External links)* (undo)
- (cur | prev) 04:18, 13 February 2011 Yobot (talk | contribs) **m** . . (11,510 bytes) (-2) . . *(WP:CHECKWIKI error fixes + general fixes using AWB (7579))* (undo)
- (cur | prev) 23:00, 14 January 2011 Jpbowen (talk | contribs) **m** . . (11,512 bytes) (-205) . . *(Tidying)* (undo)
- (cur | prev) 11:57, 20 December 2010 Xqbot (talk | contribs) **m** . . (11,717 bytes) (-3) . . *(r2.5.2) (robot Modifying: ja:ホランド (潜水艦))* (undo)
- (cur | prev) 17:40, 16 December 2010 96.250.163.216 (talk) . . (11,720 bytes) (+206) . . *(→See also)* (undo)

Figure 3.1: The revision history of the Wikipedia article on the USS Holland (SS-1).[1]

Combined with the research from Section 3.2, these theories suggest that Wikipedia may develop one dominant group of frequent users with high levels of agreement about content and policy while marginalizing users with different views or simply discouraging them from contributing. However, the presence of some users who choose to interact with infrequent users, as found by Laniado et al. may mitigate this effect [10].

Unlike many popular open-source version control systems, like Google Code[2] or Github [3], any user may commit changes to most pages of Wikipedia, not just those who have been selected as project members. This system makes Wikipedia more vulnerable to attempts to disrupt, or *vandalize*, its pages. Kittur, Chi, and Suh assigned broad categories to Wikipedia articles and then used indicators of conflict to figure out which articles are more likely to have conflict based upon their assigned category [7]. They found that articles about religion and philosophy were disproportionately likely to have conflicts. To counteract this vulnerability to vandalism, Wikipedia has a few conflict minimization strategies. Disruptive users may be prohibited from editing pages, or *banned*. Additionally, pages may be *protected* so that only certain users may edit them if they attract an unusually high amount of disruptive activity.

The Wikimedia Foundation provides downloads of its content, or *dumps*, from certain dates. Some types of dumps include just the most recent version of certain types of pages: just the most recent versions of articles, for example. One dump contains every version, or the 'full revision history,' of every Wikipedia page. These can be very helpful for collecting data on Wikipedia and they are quite large - the full revision history dumps are over 5 TB uncompressed.

3.4 Tools for Big Data

The ability to analyze large data sets was crucial to this project. The MapReduce framework and Hadoop, a Java library that implements MapReduce,

[1]http://en.wikipedia.org/w/index.php?title=USS_Holland_(SS-1)&action=history
[2]http://code.google.com/p/support/wiki/Permissions
[3]https://github.com/features/projects

helped me to deal with the vast amount of data I had to analyze. I used two other libraries to deal with the complex task of parsing Wikipedia dumps. One, Cloud9, is specifically built to deal with Wikipedia dumps including just the current versions of pages. Another, wikipedia-map-reduce, is capable of analyzing full revision history dumps.

Hadoop MapReduce is a Java library that implements MapReduce, a framework designed to allow easy processing of large datasets through distributed computing. Each MapReduce job consists of two phases: the map phase and the reduce phase. In the map phase, the data is input as key-value pairs and intermediate key-value pairs are output. The input to the reduce phase is the intermediate keys and values. Between the map and reduce phases, the intermediate data is merged. Pairs with the same key are collected together and a single copy of the key is associated with a list of the collected values. The output of the reduce phase is also key-value pairs.

I used MapReduce to create graphs of Wikipedia articles based upon their link structure. MapReduce was also the method I used to obtain information about the number of bytes added or removed by users when they edited an article.

The Cloud9 library allows users to easily parse Wikipedia articles from current version Wikipedia dumps, though not from full history dumps. It has many useful built-in functions, including a method that finds every link within a Wikipedia page to another Wikipedia page, which I used to create a graph of articles as vertices, with the links between them as edges, in the article clustering stage.

The wikipedia-map-reduce library allows MapReduce jobs to be run on full history Wikipedia dumps. It allows the user to view each revision of an article in succession, which is very useful for data analysis. I used wikipedia-map-reduce to find the total bytes added or removed by each individual author to a Wikipedia article. Additionally, I contributed code to the wikipedia-map-reduce library that finds the total bytes added or removed to a Wikipedia article by each individual author.

3.5 Graph Clustering

In this project I focus on clustering graphs consisting of vertices and edges. Within the set of algorithms for this type of graph clustering, there are two large subcategories: local clustering and global clustering. Local clustering algorithms cluster vertices together based only on the knowledge of vertices in a small subgraph, whereas global clustering algorithms use knowledge of the entire graph while making decisions. Local clustering algorithms can be more scalable, because they do not require knowledge of the entire graph. They are also less likely to require that a clustering be symmetrical. A symmetrical clustering is one in which if vertex v is in vertex u's cluster, then vertex u is in vertex v's cluster. Clusterings that are not symmetrical do not necessarily have this property. Schaeffer gives a good overview of both types of clustering algorithms [14].

In this project I used only global clustering algorithms. Global clustering algorithms were desirable because I wanted symmetrical clusterings that were based upon all available information. While clustering articles, I used a global clustering method based upon random walks on the graph of links between articles as described in Section 3.5.1 [15]. In the user clustering stage, I utilized a method based upon the approximation of a maximum cut as described in Section 3.5.2 [13].

There are many alternatives to the algorithms I chose, some of which simply were not practical for my purposes. For example, Jesus and Lehmann create bicliques, or complete bipartite graphs, of Wikipedia articles and users [6]. Edges are present if a user has edited an article at least a certain number of times. This approach requires the minimum number of edits be set, which was not a desirable property for my purposes, especially because the ideal number may not be the same across all of Wikipedia.

3.5.1 Article Clustering

As discussed in Section 3.7, the full history of English Wikipedia is a very large dataset. For this reason it was desirable to do computations upon smaller subsets of this data when possible because this allows for more in-depth analyses. Clustering is one way that we can break the data into smaller groups for analysis. Importantly, clustering allows us to group data in sensible ways. For example, I wanted to obtain small groups of articles with related topics in order to investigate the possibility that individual conflicts spanned multiple articles.

The random walk algorithm I used to cluster the article graphs was suggested by Steinhaeuser and Chawla [15]. They use random walks explicitly to find nearby vertices and then agglomerate vertices that appear in the same random walks into clusters. Additionally, they compare the quality and time complexity of their new method with several other methods: fast modularity, WalkTrap, and Markov clusters, and find that it performs quite well. Finally, they suggest ways to improve the scalability of their algorithm. This approach was desirable to me because the algorithm was efficient, required relatively low memory use, definitively clusters vertices, and was easy to implement.

Lizorkin, Medelyan, and Grineva took another approach to clustering Wikipedia articles [12]. They divided Wikipedia articles into subgraphs using the Girvan-Newman method and then attempt to categorize these clusters by their most central vertex. Girvan-Newman removes edges from a graph in order to create a dendrogram of its vertices. However, Girvan-Newman does not definitively cluster vertices, but instead suggests a number of possible clusters without providing any indicator of appropriateness.

Gibson, Kumar, and Tomkins describe an alternative heuristic-based algorithm for finding clusters specifically designed for use in very large directed graphs [4]. In their algorithm, groups of vertices that have out edges to similar groups of vertices are clustered together. Unusually, their algorithm can be set up to stream data to and from files, which dramatically lowers the memory required. This property is quite advantageous. However, we did not discover this approach until after we had completed this stage.

We also considered using Markov clusters, as discussed by Steinhaeuser and Chawla, which simulate random walks through matrix operations [15]. We concluded the matrix operations would be prohibitively expensive for us.

3.5.2 User Clustering

I used a more unusual approach to cluster groups of users in conflict. In order to separate users into groups with lots of conflict between groups but little conflict within groups, I found an approximation of the maximum cut. If a graph's vertices are divided into two groups, the cut is the total weight of the edges between the groups. The maximum cut is the greatest possible cut. However, finding the maximum cut is quite difficult, so finding an approximation was the practical solution.

Sahni and Gonzalez introduced a basic maximum cut approximation algorithm with a performance guarantee of 0.5 [13], where vertices are assigned to clusters greedily. I adapted this method slightly for my purposes as described below in section 5.2.

This contrasts with Brandes and Lerner's work on Wikipedia [2]. They measure the similarity of users' positions through the similarity of their edits, including both the addition and deletion of text. Our more simplistic approach is computationally less intensive, especially for larger groups of users. Beyond the comparative ease of approximating the maximum cut, being able to split the graph into components further reduces the complexity of the problem. It is probable, though not certain, that there would be fewer unconnected graph components with Brandes and Lerner's method given the larger number of contributing factors to their measure of similarity. Additionally, Brandes and Lerner's method positions users in space, but does not necessarily assign them a side, something we required for our later analyses.

3.6 Conflict Characterization

I wanted to address several questions about conflict in Wikipedia in this paper, including whether individual conflicts spanned more than one article and how conflicts affected editing behavior. For the purposes of this paper, I defined a *conflict* as a connected component of the revert graph within a single article cluster. In other words, if we viewed the set of all reverts within every article in an article cluster as an undirected graph with edges representing reverts and vertices representing users, each connected component would be a conflict. A *connected component* is a subgraph such that each pair of vertices has a path between them and there are no paths from any of the vertices within the subgraph to any vertices outside the subgraph.

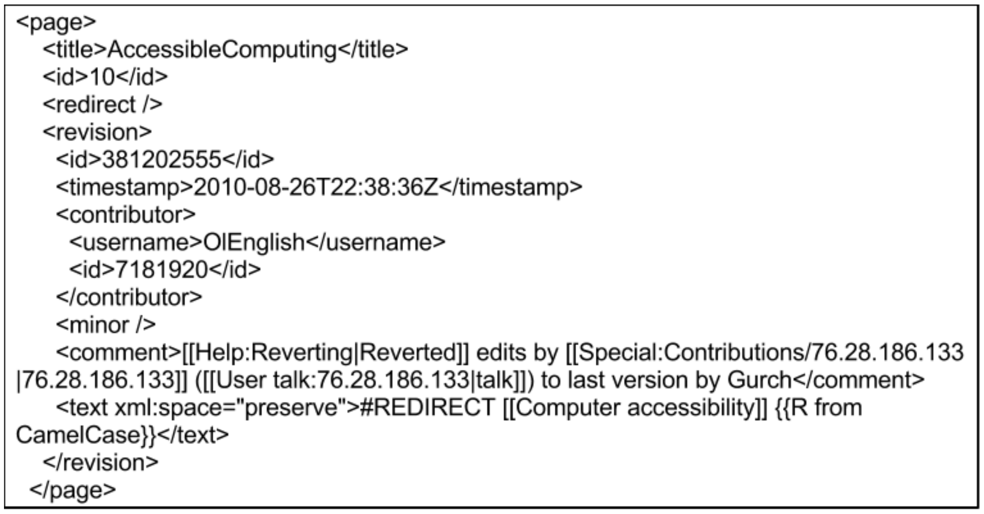
I characterized conflict in several dimensions: its span, its impact upon editing, its size, and its form. I investigated conflict span by determining how many articles were part of each individual conflict. I observed the impact of conflicts upon editors through the total number of bytes contributed by users involved in conflicts and users uninvolved in conflicts within the same article cluster and comparing those to the contributions of users in other article clusters. I measured conflict size by the number of users involved. I looked at conflict form through the properties of the conflict graphs, with a specific focus on how appropriate the my two-sided model of a conflict was for these graphs.

My work contrasts with that of Kittur, Pendleton, Suh, and Chi, who viewed conflict through the labelling of article versions as controversial [8]. My methods allow finding conflicts in pages that are not usually controversial but where conflict may be for brief periods of time.

3.7 Dataset

I used both full history and current version only Wikipedia dumps to conduct this research. I used the full history dump from December 2011 and the current version only dump from September 2011. This current version only dump was approximately 7.1 gigabytes while compressed with LMZA encoding. When uncompressed, its size was around 31 gigabytes. We encoded the full history dump in a specific manner so that it could be used with Hadoop MapReduce, such that each article and its entire revision history was represented on one line. As mentioned above, when uncompressed, the full history data dump is over 5 terabytes. We limited the research to the 1,451,634 main namespace articles visited during the 20 busiest hours for Wikipedia of a day in the summer of 2011.

Wikipedia data dumps are encoded in XML, as shown in Figure 3.2. Both the Cloud9 library and the wikipedia-map-reduce library are capable of parsing this data (Welch, 2010).



```
<page>
   <title>AccessibleComputing</title>
   <id>10</id>
   <redirect />
   <revision>
     <id>381202555</id>
     <timestamp>2010-08-26T22:38:36Z</timestamp>
     <contributor>
       <username>OlEnglish</username>
       <id>7181920</id>
     </contributor>
     <minor />
     <comment>[[Help:Reverting|Reverted]] edits by [[Special:Contributions/76.28.186.133
|76.28.186.133]] ([[User talk:76.28.186.133|talk]]) to last version by Gurch</comment>
     <text xml:space="preserve">#REDIRECT [[Computer accessibility]] {{R from
CamelCase}}</text>
   </revision>
  </page>
```

Figure 3.2: Sample XML Record

4

Clustering Articles

In this stage my objective was to make it easier to work with the Wikipedia dataset by decreasing the problem size. By looking at conflict within small groups of articles as opposed to within the entire website, I made it feasible to conduct more in-depth analyses on individual conflicts.

4.1 Desired Article Clustering Features

There were three types of clustering to choose between: behavioral, lexical, and semantic. For example, a behavioral clustering might be based upon co-occuring edits, or whether articles were edited by the same users, while lexical clustering might be based upon the textual similarity between articles, and semantic similarity could be based upon links between articles. We chose to use a semantic measure of similarity because we thought it would give the best results for the purposes of finding conflicts. We rejected lexical clustering because we thought it was likely to lead to large and overly diverse clusters. We rejected behavioral clustering because we thought this would likely lead to clusters based around editing habits as opposed to topics and might be overly influenced by the editing habits of a few highly active users. We chose clustering based on link structure because it met our criteria of including each vertex in only one cluster, was topic-based as opposed to behavior-based, and was relatively easy to implement.

4.2 Creating the Article Link Graph

In addition to choosing a clustering algorithm, we also had to construct graphs of articles to cluster. We wanted to cluster denser graphs as opposed to sparser ones in order to decrease the number of resulting clusters and increase their size. We especially wanted to avoid large number of isolated vertices, which would become clusters of size one, significantly reducing the value of the clustering. Larger article clusters would also allow us to find more cross-article conflicts, whereas we would be unable to find conflicts spanning multiple articles in clusters with only one article. For these reasons, we chose to cluster a subgraph of Wikipedia containing about 1.45 million main namespace articles that met the popularity criteria described in section 3.7.

First, I obtained a graph containing articles as vertices with edges between articles if one article linked to another. I tried several approaches to this task. First, I attempted to use the wikipedia-map-reduce library. This approach was problematic for two reasons. First, due to human error there were several incorrectly coded links or links with misspellings. Secondly, wikipedia-map-reduce is designed to work on full history Wikipedia dumps, while this task only required one version of each article. This meant that using wikipedia-map-reduce made the task much more resource intensive. For these reasons, I switched to using the Cloud9 library, which is built to be used with current version only Wikipedia dumps, which contain only the current versions of pages, as opposed to their entire history. Utilizing its built-in functions, I was able to write MapReduce jobs that provided me with both directed and undirected link graphs of Wikipedia.

Figure 4.1 demonstrates the differences between directed and undirected graphs. As shown, edges in a directed graph are directed from one vertex to another, whereas edges in an undirected graph have no direction. In a directed graph, the number of edges directed to a vertex is its *in-degree* and the number of edges directed from a vertex are its *out-degree.* In an undirected graph, the number of edges a vertex has is simply called its *degree.*

Prior to clustering the graphs, it was unclear whether there would be an advantage to clustering the directed or undirected graph or whether subtle differences between the clusterings would emerge in later stages. The di-

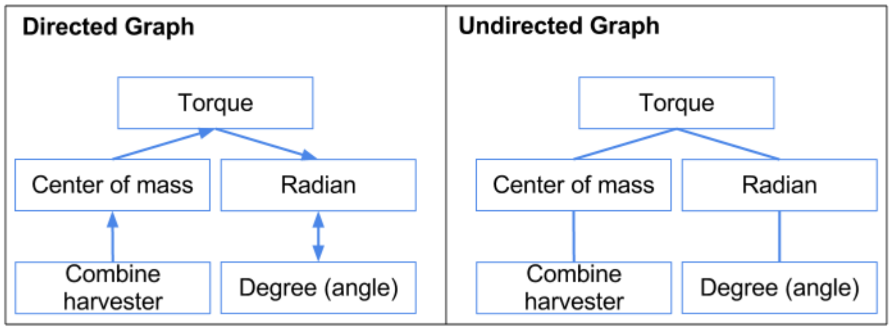


Figure 4.1: A directed graph on the left and an undirected graph at right.

Graph	Vertices	Edges	Average Degree	Median Degree	Median In-Degree	Median Out-Degree
Directed, All Pages	17,243,428	119,310,822	6.91	N/A	1	1
Directed, Main Namespace and Ranked Only	1,438,428	34,319,866	23.85	N/A	6	14
Undirected, All Pages	17,265,111	113,028,472	13.09	2	N/A	N/A
Undirected, Main Namespace and Ranked Only	1,438,793	31,292,409	43.49	20	N/A	N/A

Table 4.1: Statistics for directed and undirected article graphs including all pages and just those in the main namespace and ranked category. Average degree for the directed graphs is for in- or out- degree, not in- and out- degree.

rected edges in the directed graph may contain more useful information than the undirected edges. However, the undirected graph is much denser than the directed graph, which could lead to larger clusters - which is better so long as they retain their semantic meaning.

As seen in Table 4.1, the all-article graphs each had around 17.25 million vertices.[1] I have also compiled statistics for the subgraphs I described

[1]Due to inconsistent formatting in the files used to translate article names to Wikipedia article IDs some articles were incidentally omitted from the Main Namespace and Ranked Only category, which should have had 1,451,634 articles in both the directed and undirected graphs. Articles missing include, but are not necessarily limited to, many articles with dashes in their titles. Approximately 2,000 more articles disappear during article clus-

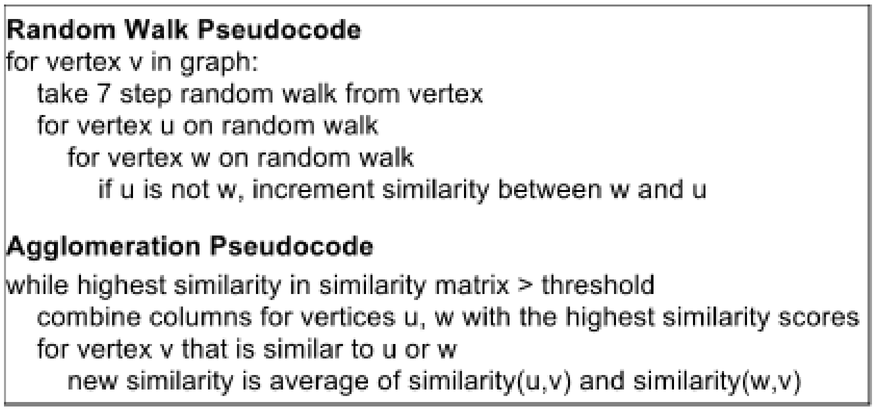
above. The directed graph contained about 11.9 million edges, with each vertex averaging about 6.9 in- or out- degrees. The median in-degree was 1, which happened to also be the median out-degree. Approximately 7.4 million vertices had an out-degree of zero, while about 4.1 million vertices had an in-degree of zero. The undirected graph contained about 11.3 million edges, or an average of around 13.1 degrees for each vertex. The median degree was 2. The similarity in the number of edges suggests there are few vertex pairs in the directed graph that have edges in both directions between them: every vertex pair with an edge in each direction in the directed graph is one fewer edge in the undirected graph. This explains why the undirected graph is denser than the directed graph, though both graphs are very sparse.

As mentioned above, the subgraph of the main namespace articles that met the popularity criteria was significantly denser than the full graph. The directed subgraph contained about 34.3 million edges, or an average in- or out-degree of approximately 23.9. The median in-degree was 6, while the median out-degree was 15. 123,727 vertices had no in-edges, while only 242 had no out-edges. The undirected subgraph contained around 31.3 million edges, or an average degree of about 43.5 per vertex. The median degree was 20. Similarly to the full graphs, the undirected subgraph has a similar number of edges to the directed subgraph, meaning that each individual vertex has a higher degree. Additionally we can see that there is a large difference between the median in- and out- degrees for the median graph, suggesting that while out-edges are relatively evenly distributed, there are a relatively small number of articles with large numbers of in-edges.

4.3 Clustering Algorithms

We considered several approaches to generating graphs for article clustering. Our most important constraints were a desire for hard clusterings: each article should be in only one cluster, and a desire for a flexible cluster size depending on the density of the local graph. Allowing articles to be in more than one cluster could lead to article clusters sharing large portions of their content, which could result in duplicate conflicts later on. Flexible cluster sizes were important because different cluster sizes might be appropriate in

tering and the source of this problem has not yet been conclusively identified. Fortunately these two problems affect a total of only 1.1% of our data.



Random Walk Pseudocode

```
for vertex v in graph:
    take 7 step random walk from vertex
    for vertex u on random walk
        for vertex w on random walk
            if u is not w, increment similarity between w and u
```

Agglomeration Pseudocode

```
while highest similarity in similarity matrix > threshold
    combine columns for vertices u, w with the highest similarity scores
    for vertex v that is similar to u or w
        new similarity is average of similarity(u,v) and similarity(w,v)
```

Figure 4.2: Pseudocode of the article clustering algorithm

different parts of Wikipedia: different topics might have different numbers of articles relating to them, for example.

I chose to cluster the article link graphs by utilizing random walks, a method introduced by Steinhaeuser and Chawla [15]. I had difficulty successfully implementing their method due to the high memory requirements of their algorithm. As can be seen in Figure 4.2, the algorithm first takes a random walk of seven steps from each vertex in the graph. When a random walk is completed, it increments the similarity between each pair of vertices on the walk by one. After all the random walks are complete, the agglomeration stage begins. Initially each article is its own cluster, but during agglomeration, the two most similar clusters are merged so long as the greatest similarity is larger than a predetermined threshold.

The major limiting factor was the number of steps taken in the random walk. As the number of steps increases, the amount of space necessary to store information about which vertices appear in the same the random walks increases quadratically. Although the random walk stage was the most memory intensive, I found the agglomeration stage took more time to run.

Due to memory constraints, I tried three different data structure solutions for storing the data generated by the random walk during the agglomeration stage. My first solution involved using two hashtables. The first mapped pairs of vertices to their similarities and the second mapped scores to sets of

Graph	Articles	Clusters	Average Cluster Size	Median Cluster Size	Size One Clusters
Directed	1,436,625	424,012	3.38	2	195,229
Undirected	1,436,666	249,587	5.75	5	25,437

Table 4.2: Statistics for clusterings of the directed and undirected article graphs. Each clustering used a 7 step random walk and a threshold of 0.2.

vertex pairs and was sorted to make finding the vertex pair with the highest score easy. However, there was no way to easily determine which vertices were adjacent to which, and it was necessary to check each vertex pair to see if it contained one of the merging vertices. This lead agglomeration to be $O(n^2)$, which was impractically slow. My second solution added a new hashtable containing adjacency lists. Unfortunately, this required more memory than was available on the server I was using. Finally, I eliminated the hashtable of vertex pairs to scores and added hashtables within the adjacency lists to similarity scores for each adjacency. These different versions can be seen in Figure 4.3.

I was able to run the algorithm on both the directed and undirected graphs after I changed the data structures and reduced the number of steps taken in the random walk from 10 to 7. The directed graph requires about 6.7 gigabytes of memory and took two hours and ten minutes, while the undirected graph requires 7.4 gigabytes of memory and took over two and a half hours on a dual CPU Intel X5650 server with 32GB of memory.

4.4 Results of Article Clustering

I conducted article clusterings on both the directed and undirected graphs. For each graph, 7 random steps were taken during the random walk stage and the threshold for merging two clusters was 0.2.

As seen in Table 4.2, the directed article clustering produced more than 400,000 article clusters with an average cluster size of about 3.39 and almost 200,000 articles with their own clusters. Given that 123,727 articles in the directed graph with only main namespace and ranked articles had an in-

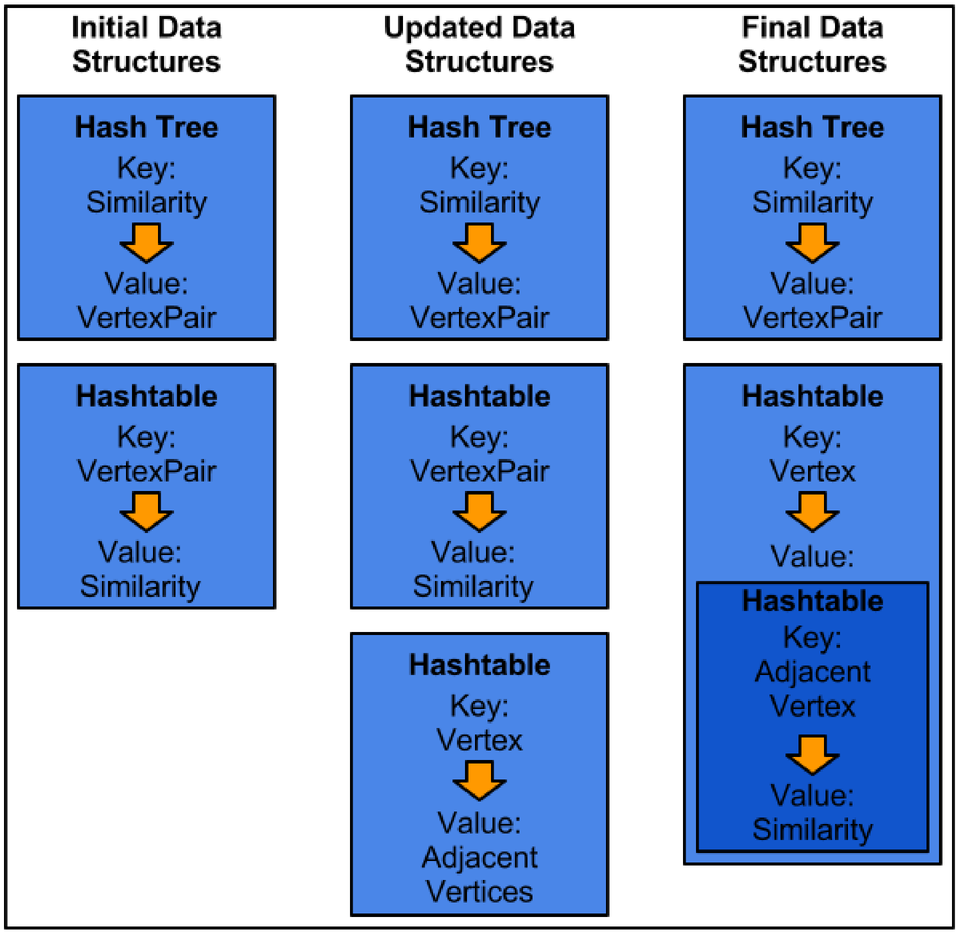


Figure 4.3: Data structures used in different versions of the article clustering algorithm

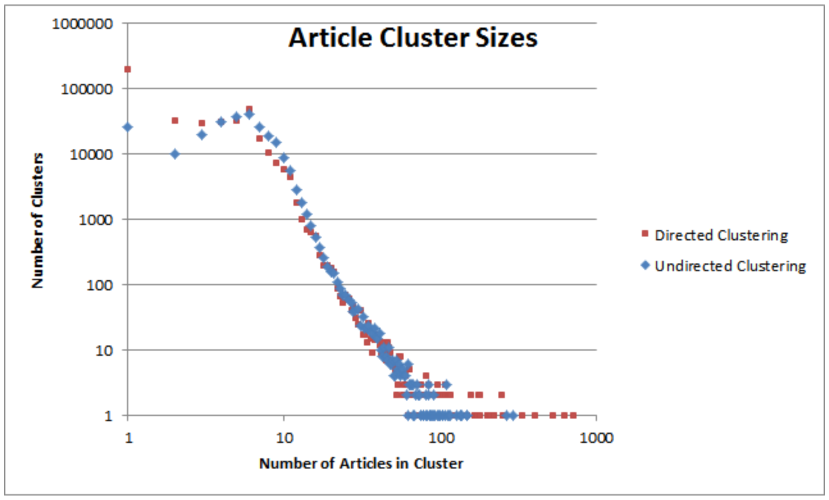


Figure 4.4: The number of clusters with a certain size for both the directed and undirected clusterings and trendlines.

degree of zero, this is not an unreasonably high number of size one clusters. As a result, most clusters were quite small; the median cluster contained two articles.

The undirected article clustering produced less than 250,000 article clusters with an average cluster size of approximately 5.76 and around 25,000 articles in clusters of their own, a huge decrease. The median cluster was of size 6.

As can be seen in Figure 4.4, the directed clustering and the undirected clustering have similar distributions of article cluster sizes. It should be unsurprising that the undirected graph would have larger clusters and lower numbers of singletons, given how much easier it is to traverse during the random walk and the higher similarity scores that would result from that. Both distributions have a local maximum at six articles. The threshold of 0.2, combined with our choice of using a weighted average to merge similarity scores, probably had a large impact on this. This threshold allows article clusters to merge when the average similarity scores of their members is as low as 0.2, which is equal to one fifth and greater than one sixth. This makes it more difficult to form clusters of more than six articles, because the next article added to the cluster must have had a combined initial similarity of at least two with the articles in the cluster, as opposed to the combined initial similarity of one that is required for adding to smaller cluster sizes.

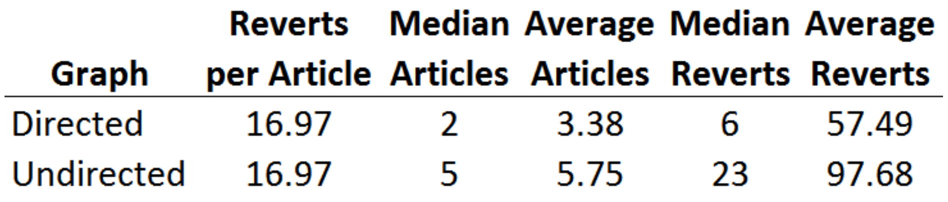
5

Clustering Users Within Article Clusters

After creating clusters of articles, I next identified specific conflicts among users within article clusters and split those involved into sides. For this clustering algorithm I required a method capable of handling large amounts of data efficiently with regards to both memory and time complexity. Additionally, the algorithm had to produce a hard clustering.

5.1 Creating the Revert Graph

I clustered users using a graph of reverts from each article cluster. We generated an undirected graph of all reverts within an article cluster, where vertices represented users and edges represented reverts between a pair of users. The edges were weighted by the number of reverts between the users.



Graph	Reverts per Article	Median Articles	Average Articles	Median Reverts	Average Reverts
Directed	16.97	2	3.38	6	57.49
Undirected	16.97	5	5.75	23	97.68

Table 5.1: Statistics on the number of reverts within an article cluster in the directed and undirected article clusterings

To create the graph, we used a method introduced by Kittur, Suh, Pendleton, and Chi and our WikiMiner library [8]. There are two basic components to this method: one that relies on users identifying reverts through their revision comments and the other based upon the equality of the MD5 hashes of different revisions. If the user identified their revision as a revert, then we counted it as such. An MD5 hash is a checksum commonly used to verify that files are identical. In this case, if a new revision of an article has the same MD5 hash as an older revision it is very likely the revisions are identical and a revert has occurred. Kittur, Suh, Pendleton, and Chi found that the MD5 hash equality method identified a larger number of reverts than comments, but that the methods tended to agree [8].

As seen in Table 5.1, the average article had almost 17 reverts. The average article cluster in the directed graph had 3.38 articles and 57.49 reverts, while the average article cluster in the undirected graph had 5.75 articles and 97.68 reverts. Median revert numbers were much lower, suggesting that a few article clusters experienced a large proportion of all reverts. The larger differences between the median and the average articles and reverts for the directed graph suggest conflict is more heavily concentrated within a few clusters in that graph than in the undirected graph.

5.2 Clustering Algorithm and Implementation

I divided each revert graph into *connected components*, which I considered as to be individual conflicts. A connected component is a subgraph such that each pair of users within the subgraph has a path of reverts between them and no user within the subgraph has a path of reverts to any user outside the subgraph. I clustered the users in each conflict by splitting them into two sides with the objective of having as many reverts as possible be between users on opposing sides. In order to implement my approach, I had to be able to find the maximum cut of a graph. Ideally, finding the maximum cut in a revert graph component separates the component into two groups of users such that there are few reverts between users in the same group, but many reverts between users in different groups. This implies disagreement between the groups. Unfortunately, finding the maximum cut is an NP-complete problem, so I used an approximation [13]. The approximation is obtained by assigning each vertex to one of two groups. The first vertex is assigned randomly and then its neighbors are assigned with the objective of increasing the cut as much as possible. The cut is the total weight of the edges between the two groups. The algorithm continues increasing the cut by moving a user from one group to another until no further increasing moves are possible. By assigning a vertex's neighbors immediately after it is assigned we can guarantee optimal results for all bipartite graphs.

My implementation of the algorithm conducts multiple user clusterings simultaneously, with each revert graph given its own thread. My implementation uses a separate thread to write results, avoiding race conditions while writing to the output file. I utilize a thread-safe queue to store results between their computation and when they are written. Additionally, one thread is responsible for reading input revert graphs and creating user clustering tasks. A visual representation of this architecture is given in Figure 5.1. We implemented this parallelism using the LinkedBlockingQueue, ThreadPoolExecutor, and Runnable classes from Java's java.util.concurrent library.

Initially my implementation of the maximum cut approximation algorithm found one cut for each provided revert graph, but I later revised it to find a cut for each connected component of the revert graph. Figure 5.2 demonstrates the advantages of calling each connected component of the revert graphs a separate conflict as opposed to viewing each revert graph as a conflict. Because there is no information within the revert graph indicating whether users in different connected components should be considered to be in conflict or on the same side of a conflict it is better to assume that they are not involved in the same conflict.

5.2.1 Limitations of the Maximum Cut Algorithm

The maximum cut clustering algorithm has a few drawbacks. First, it limits the number of possible sides to two and therefore cannot accurately represent a conflict with three or more sides. Secondly, time is not accounted for. If a single user is involved in two different conflicts occurring at separate times in the article history, this is treated as one conflict. Third, this method does not take into account the difference between the reverter and the reverted.

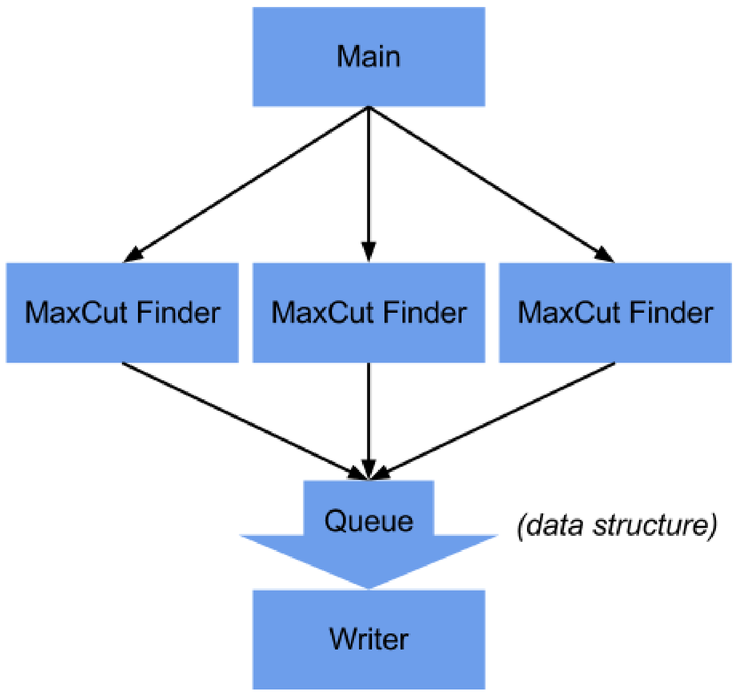


Figure 5.1: Example of maximum cut parallelism

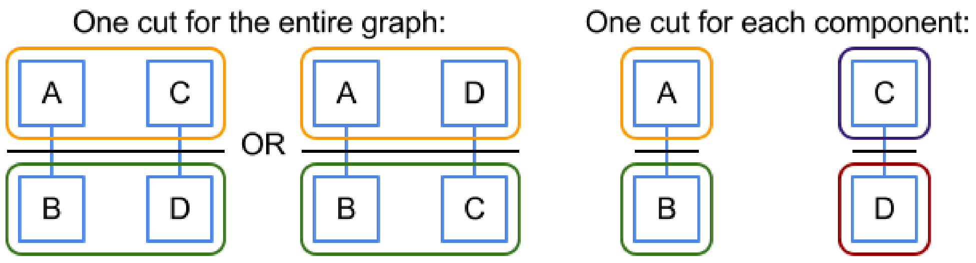


Figure 5.2: Example of the difference between one maximum cut for a revert graph as opposed to a connected component. Note that there is no reason for A or B to be on the same side as C or D, nor is there a reason for them opposed to C or D. For this reason the one cut for each component solution is preferable.

Graph	Article Clusters	Conflicts	Average Conflicts in a Cluster	Median Conflicts in a Cluster	Average Users in a Conflict	Median Users in a Conflict
Directed	424,012	6,984,960	16.47	3	3.59	2
Undirected	249,587	7,217,465	28.91	12	3.53	2

Table 5.2: Statistics for user clusterings within the directed and undirected article clusterings

Reverts are directed edges, though I consider them as undirected edges. As suggested by Leskovec, Huttenlocher, and Kleinberg, a method utilizing the status implications of reverts to find conflicts might be more perceptive than the method I used, which is based in what they refer to as balance theory [11].

5.3 Results of User Clustering within Article Clusters

User clusterings on the directed and undirected graphs turned out to be quite similar. Basic statistics on the user clusterings can be viewed in Table 5.2. The number of conflicts in the directed graph user clustering differed from the number of conflicts in the undirected graph user clustering by less than 250,000, or about 3.5%. The difference in the average number of users in a conflict was even smaller, and the median number of users in a conflict was the same for each user clustering: two.

While the majority of conflicts are very small - over 5.5 million of the conflicts in the undirected graph involve just two users, there are also a number of quite large conflicts. In the undirected graph user clustering, there are over 12,000 conflicts involving more than 100 users each and more than 800 conflicts involving over 1,000 users each.

In Figure 5.3 we can see the number of conflicts containing a certain number of users. For ease of viewing, both axes are using logarithmic scales. In the upper left we can see that there are a very large number of conflicts with just two users, and under one million with three users. In the bottom right of

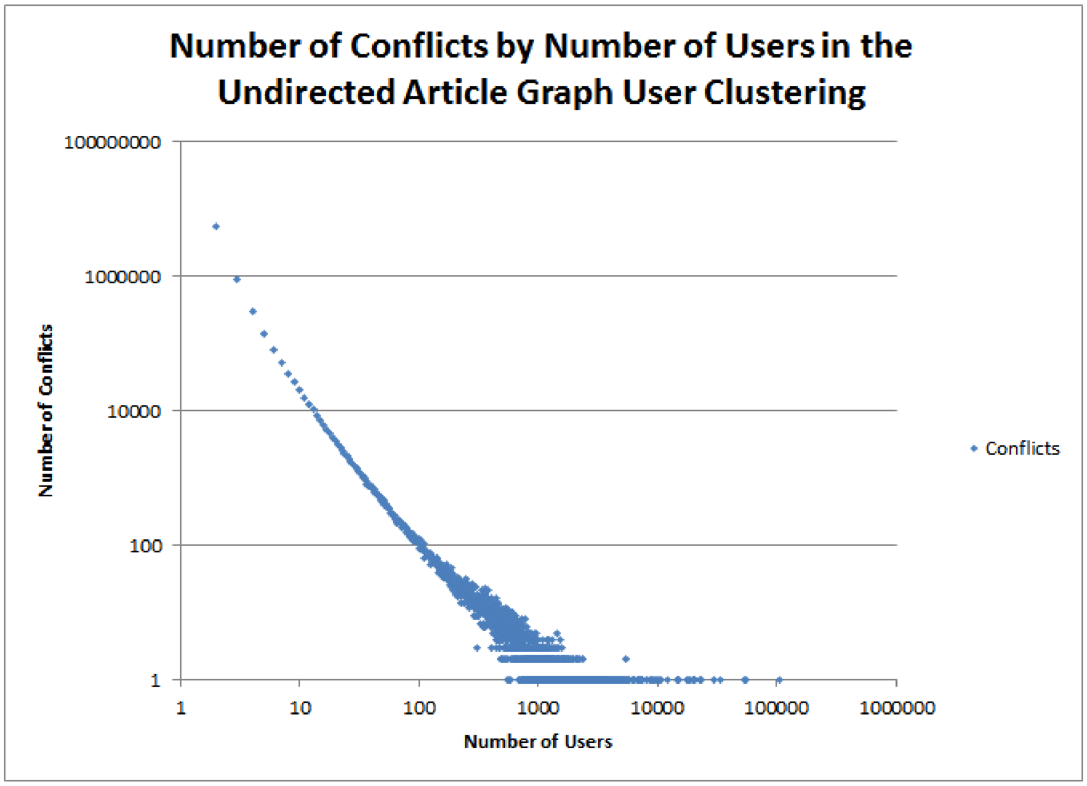


Figure 5.3: The number of conflicts in which a certain number of users are involved. The data were binned to the nearest power of $e^{0.2}$. All higher values of conflicts were binned to $e^{10.4}$.

the figure we can see that there is one conflict containing more than 100,000 users. This figure shows that the vast majority of conflicts involve very small numbers of users, but there are also small numbers of conflicts that involve very large numbers of users.

The number of conflicts in a cluster is also distributed very unevenly. In the undirected graph user clusters, there are more than 65,000 article clusters that have no conflicts. In the directed graph user clusters, the number is dramatically higher, such that the median number of conflicts in a cluster is zero. The undirected graph user clusters have about 20 conflicts, on average, with a median of five. In Figure 5.4 we can see that conflicts are also not distributed evenly on a per article basis. Some articles have many more re-

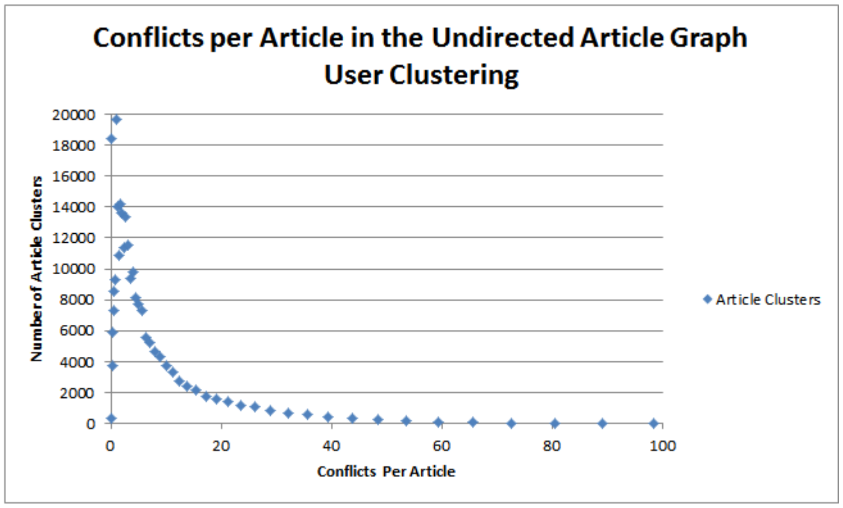


Figure 5.4: The distribution of conflicts among article clusters. The data were binned to the nearest power of $e^{0.1}$, with a shift of one to account for article clusters with zero conflicts. All higher values of conflicts were binned to $e^{4.6} - 1$.

verts than average, causing their article clusters to have many more conflicts than average. One article cluster contains over 90 conflicts per article.

Importantly, the conflicts in both the directed and undirected graphs are also either bipartite or very close to bipartite, in general. Examples of trees, bipartite graphs and non-bipartite graphs may be seen in Figure 5.5. Briefly, a tree is a connected graph without cycles and a bipartite graph is a graph such that all the edges are between two groups of vertices and none of the edges are within a group. Bipartiteness is the total edge weight in a cut divided by the total edge weight in the graph. Of the 7,217,465 conflicts in the undirected user clustering, 7,162,847, or 99.2%, were trees. Of the remaining 54618 conflicts, another 16989, or 0.2%, were bipartite, leaving only 37629, or 0.5%, non-bipartite conflicts. These non-bipartite conflicts had an average bipartiteness of about 0.92 by our maximum cut approximation algorithm. In other words, even in 0.5% of graphs where not every revert is between two sides, an average of nine out of ten reverts are between the two sides. This suggests that having two sides in every conflict is not much of a limitation.

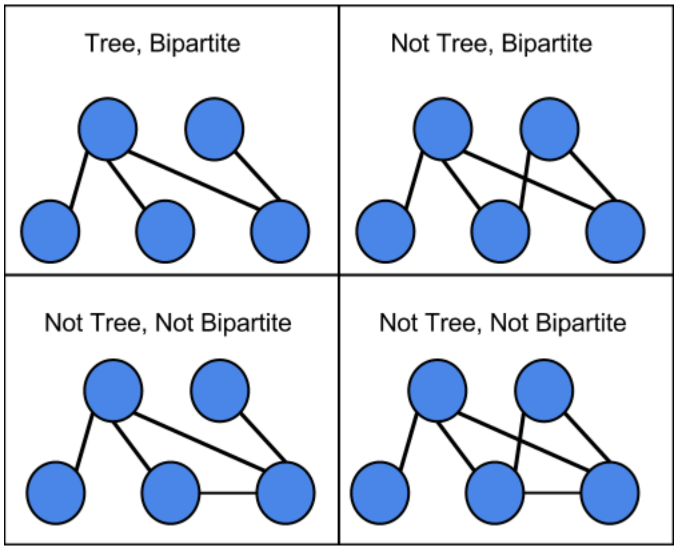


Figure 5.5: Trees, Bipartite Graphs, and non-Bipartite Graphs

One problem with attempting to measure the connectedness of a connected component by using average degree is that larger connected components have larger minimum average degrees than smaller ones. For example, a tree with two vertices has average degree equal to one, whereas a tree of size three has average degree 1.5. As the number of vertices in the tree approaches infinity, the average degree approaches two. In order to correct for this bias, I calculated the *jungle density*, which was defined as the number of unique reverts divided by the number of users involved minus one, for each conflict and rounded it to the nearest tenth. As a rule of thumb, a user can expect to have a revert relationship with a number of other users about equal to twice the jungle density of a conflict because the jungle density is close to half the average degree of a graph. Trees have jungle density of one. Examples of jungle density are presented in Figure 5.6. It should be clear that jungle density increases linearly with the number of unique reverts present. This is a reasonable assumption for most of the non-trees, which have a median size of 25 users in the undirected user clustering.

The largest jungle density is 2, which suggests the average user in a conflict is reverting or being reverted by a small number of other users, even if they revert or are reverted many times. Alternatively, the average user

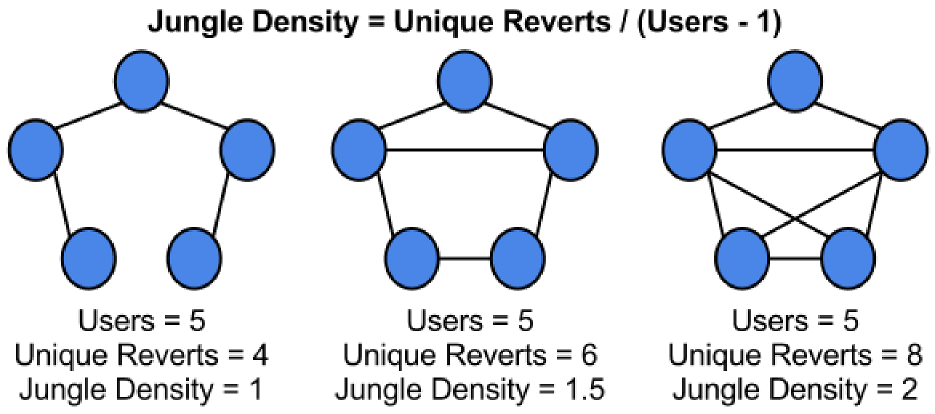


Figure 5.6: Jungle density formula and example graphs

comes into conflict with a small number of opposing users even in conflicts involving many users. In Figure 5.7 we can notice a large dip in bipartiteness at jungle density of 1.5 and also a corresponding uptick in the number of conflicts with jungle density of 1.5 in Figure 5.8. I suspect that many of these conflicts are triangle graphs. The bipartiteness of the conflicts with jungle density of 2.0 is also relatively low, but there is only one of these. The remaining data points support the conclusion that bipartiteness decreases very slowly as jungle density increases, suggesting that some conflicts with higher jungle density might be better suited to models of conflict allowing more than two sides. However, there do not appear to be very many of these conflicts.

As I would expect, Figure 5.8 shows there are are a large number of conflict with jungle density of 1. The number of conflicts with jungle density of above 1.5 is less than 200, or less than 0.001% of all conflicts, and less than 0.005% of non-tree conflicts.

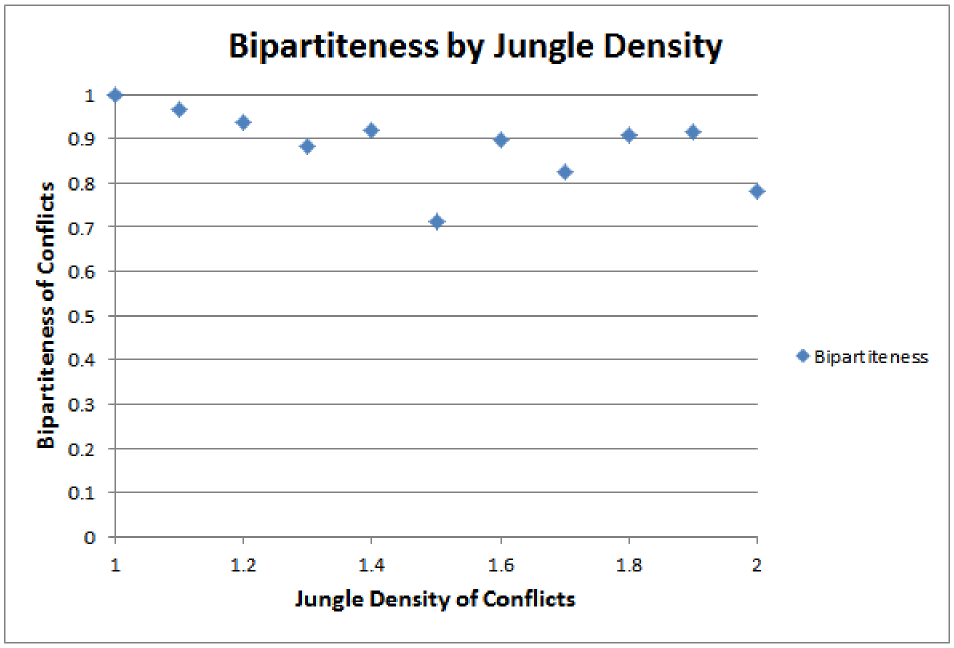


Figure 5.7: Average Bipartiteness by Jungle Density of Conflicts

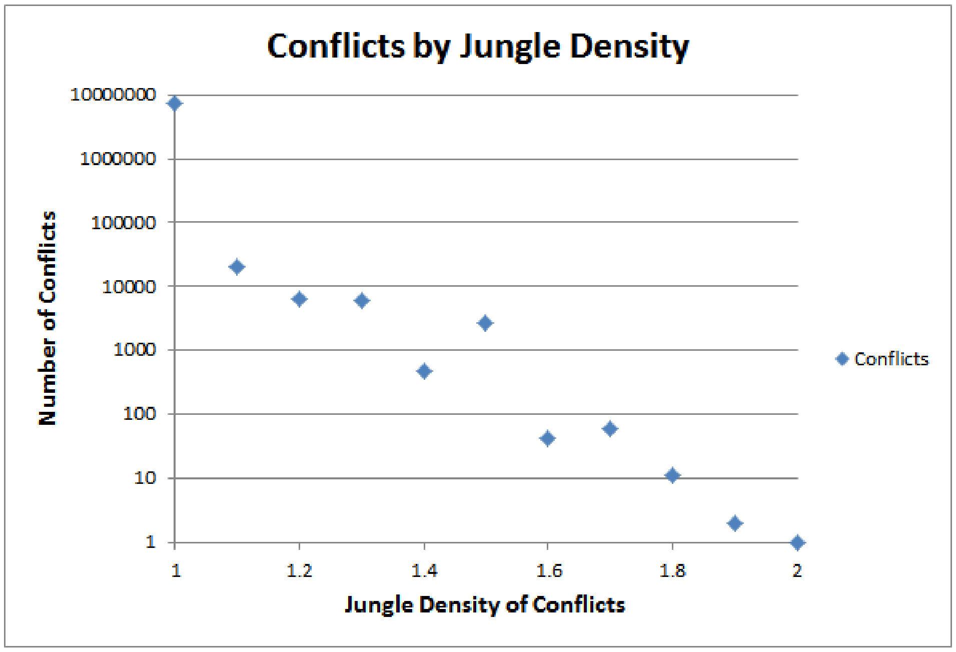


Figure 5.8: Conflicts by Jungle Density

6

Characterizing Conflict

This section addresses the effects of conflict upon user contributions to Wikipedia, the presence of individual conflicts spanning multiple Wikipedia articles, and the frequency with which conflicts spanning multiple articles occur. There were two main measures I used to analyze the user clustering data and achieve these objectives: the amount of content added or deleted and the number of articles involved in a conflict.

6.1 Conflict Size by Articles

One of my main research questions was to determine how often individual conflicts on Wikipedia spanned more than one article. I analyzed this by determining how many articles were involved in each conflict. I defined a conflicts' span as the articles in which its reverts occur.

The process of finding each conflicts' span was parallelized using the same techniques as in user clustering (Section 5.2). First, information about conflicts and their reverts is read in to memory. Each thread charged with determining conflict span is given all the conflicts and reverts from a single article cluster and there is one additional thread that outputs all the results to a file.

This data was used to analyze how conflict span varied as the number of users and reverts involved in the conflict changed. This allows analysis of how often conflicts take place in more than one article as well as comparisons between conflicts that occur in only one article and those that take place in

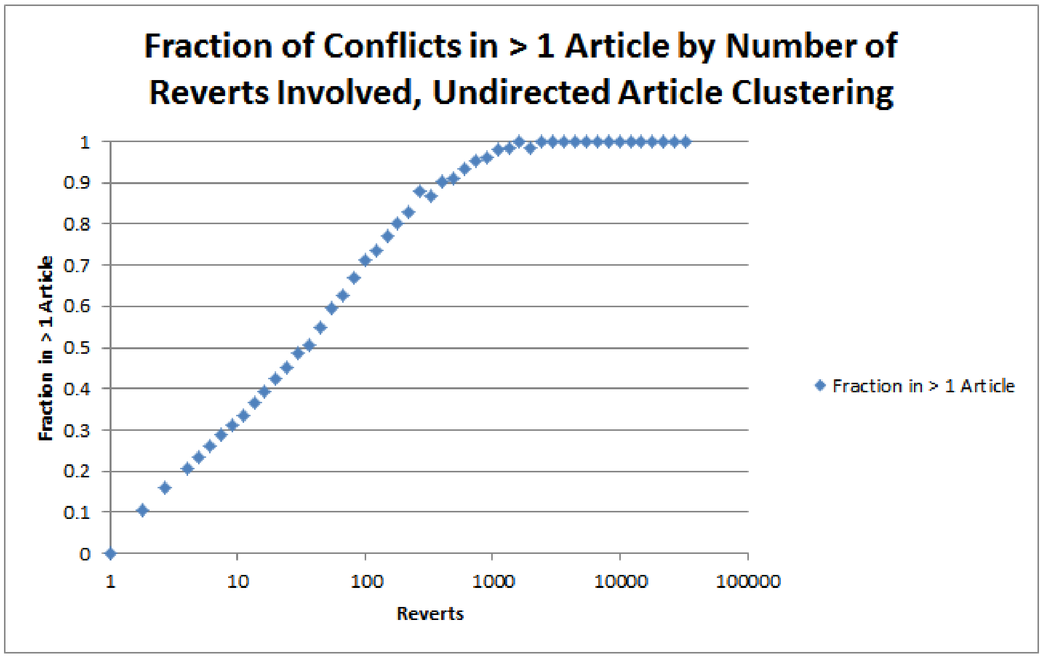


Figure 6.1: The fraction of conflicts spanning more than one article by the number of reverts in the conflict. The data were binned to the nearest power of $e^{0.2}$. All higher values of conflicts were binned to $e^{10.4}$.

more than one article.

Very few conflicts span more than one article. 74.1% of conflicts involve only two users and one revert. As the number of reverts increases from one and the number of users increases from two, the percentage of conflicts that span more than one article increases logarithmically until around 1000 reverts or users in a conflict, as seen in Figure 6.1. I have binned the data logarithmically in order to reduce the amount of noise. Additionally, the graphs for reverts and for users (Figure 6.3) are very similar. This is because 99.2% of conflicts are trees with one fewer reverts than users, as mentioned above in Section 5.3.

We can see in Figure 6.2 that the numbers of conflicts with a certain number of reverts dips below 10 between 100 and 1000 reverts, and for this reason the data for more than 100 reverts would have a high variance if they

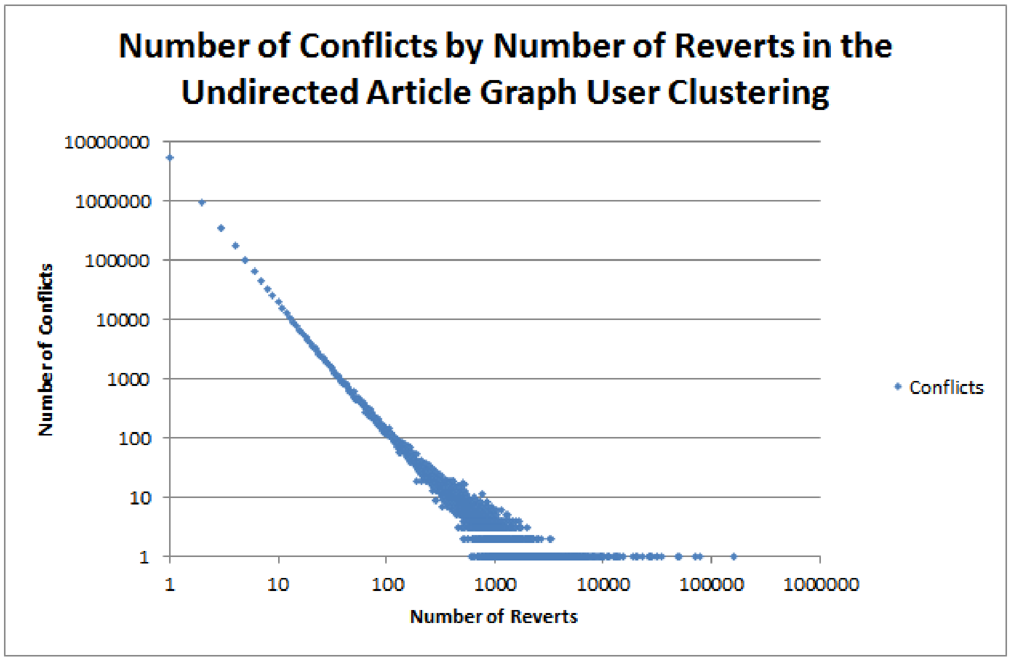


Figure 6.2: The conflicts by the number of reverts involved. The data were binned to the nearest power of $e^{0.2}$. All higher values of conflicts were binned to $e^{10.4}$.

were not binned.

Figure 6.3 plots the percentage of conflicts that span more than one article against the number of users involved in the conflict. This plot is very similar to the one plotted against reverts. We can see that the fraction of conflicts that span more than one article increases logarithmically until around 1000 users, at which point almost all conflicts span more than one article. These data have also been logarithmically binned, for the same reasons as mentioned above.

Figure 6.4 shows the average number of articles in a conflict involving a certain number of users. The number of articles involved in a conflict increases quite slowly, but steadily, as the number of users involved increases. I used logarithmic binning to ensure this figure was easily viewed.

Our data suggest that the complete analysis of large or intense conflicts

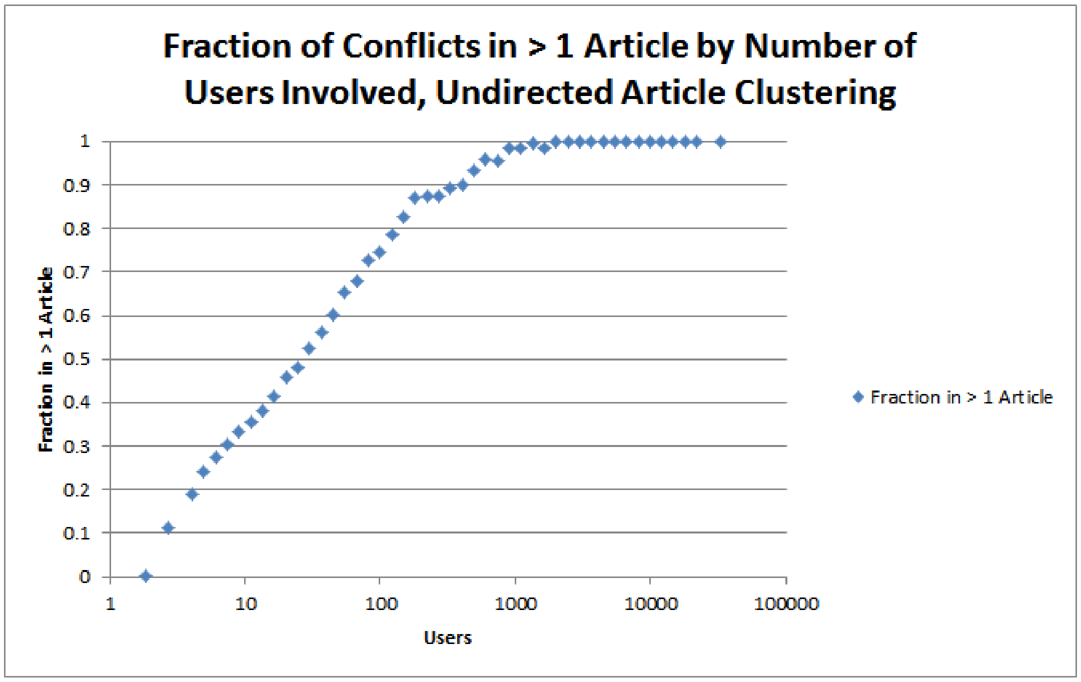


Figure 6.3: The fraction of conflicts spanning more than one article by the number of reverts in the conflict. The data were binned to the nearest power of $e^{0.2}$. All higher values of conflicts were binned to $e^{10.4}$.

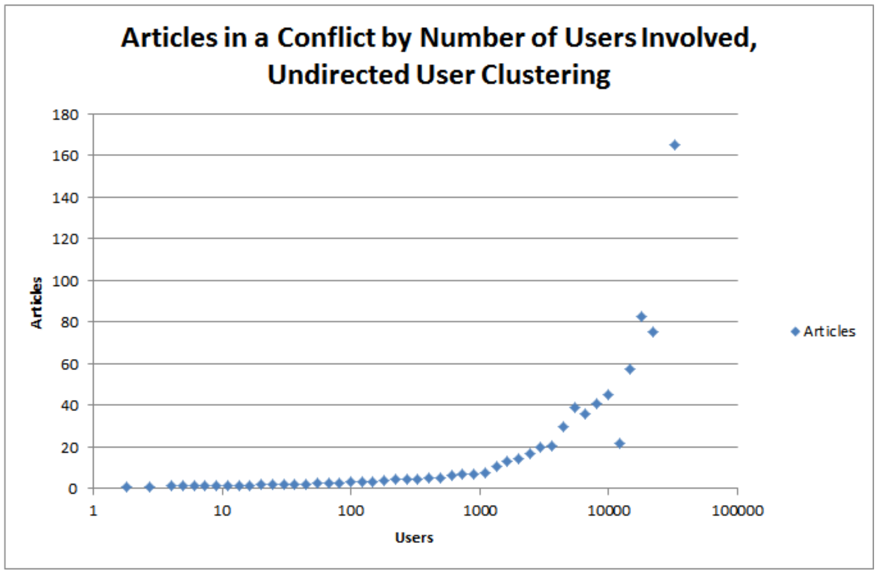


Figure 6.4: The average number of articles in a conflict involving a certain number of users. The data were binned to the nearest power of $e^{0.2}$. All higher values of conflicts were binned to $e^{10.4}$.

on Wikipedia requires the ability to analyze an individual conflict in more than one article.

6.2 Content Added or Deleted

Kittur, Suh, Pendleton, and Chi argue that one threat to Wikipedia is increasing coordination costs, or difficulty of working constructively with other users [8]. Conflicts between users are a prime contributor to coordination costs. Conflict can impose coordination costs on users directly involved, but it can also increase the coordination costs of users who wish to remain outside the conflict. Therefore, I needed to observe not only the differences between users in conflicts and users not in conflicts, but also how productivity changes as the amount of conflict within an article cluster varies. One way to compare users' productivity is by the amount of material they have contributed to an article. Comparing users' contributions as the level of conflict changes addresses the question of whether conflict reduces or enhances users' productivity.

Graph	Average Bytes Changed	Users in Conflict	Users not in Conflict	Users in Conflict - Users not in Conflict
Directed	142.14	176.14	129.41	46.73
Undirected	141.67	180.13	127.05	53.08

Table 6.1: Statistics about average bytes changed by users in conflicts as well as users not in conflicts for both the directed and undirected user clusterings.

In order to conduct these analyses, I first had to obtain data on users' contributions to Wikipedia. I used the wikipedia-map-reduce library to find the total bytes added or removed by each user within an article. Then I aggregated this data to the cluster level for both the directed and undirected graphs. I used the cluster level data to find the average number of bytes added or removed by a user within the cluster. I also used the cluster level data to determine the average number of bytes added or removed by a user on each side of each conflict.

I parallelized the last parts of this process in the same way I did user clustering (Section 5.2). After data from each cluster was read, it was given its own thread for finding the cluster and conflict averages. A single thread was responsible for writing all the output.

Statistics for users' average contributions can be found in Figure 6.1. Average bytes changed is the average contribution of a user to an article cluster, counting only users who contributed to at least one article in the cluster. Users in Conflict is the average contribution in bytes of a user involved in a conflict in the article cluster. Users not in Conflict is the average contribution in bytes of a user not involved in a conflict in the article cluster. The last number is simply the difference between Users in Conflict and Users not in Conflict. The positive numbers mean that users directly involved in conflicts contributed more to the article cluster than users not involved in a conflict in the article cluster on average.

The statistics are quite similar for both the directed and undirected graphs. Especially noteworthy is the fact that users involved in conflicts contribute more than the average user while users not in conflicts contribute less. This is a surprising result, because removing text is counted as a negative contribution of bytes, users in a conflict must be involved in a revert, and the majority of reverts are a net contribution of zero bytes by the two users involved. This suggests users involved in conflicts may tend to be more active contributors than users who are not in conflicts.

As can be seen in Figures 6.5 and 6.6, the number of conflicts in an article cluster had a very large effect on the average number of bytes contributed by a user. Figure 6.5 graphs average bytes changed by a user within an article cluster against conflicts in the article cluster. Figure 6.6 graphs average bytes changed by a user within an article cluster against conflicts per article. This figure clearly shows that article clusters with few conflicts relative to their number of articles had larger average user contributions. It would seem that as users spend more time reverting other users, they spend less time adding new content or cancel out their additions by removing the work of others. Vandalism contributes to this effect by forcing one user to negate another's contribution. Given that users in conflicts tend to contribute more than users not in conflicts, Figure 6.6 suggests that conflicts have a deterent effect upon the contributions of all users.

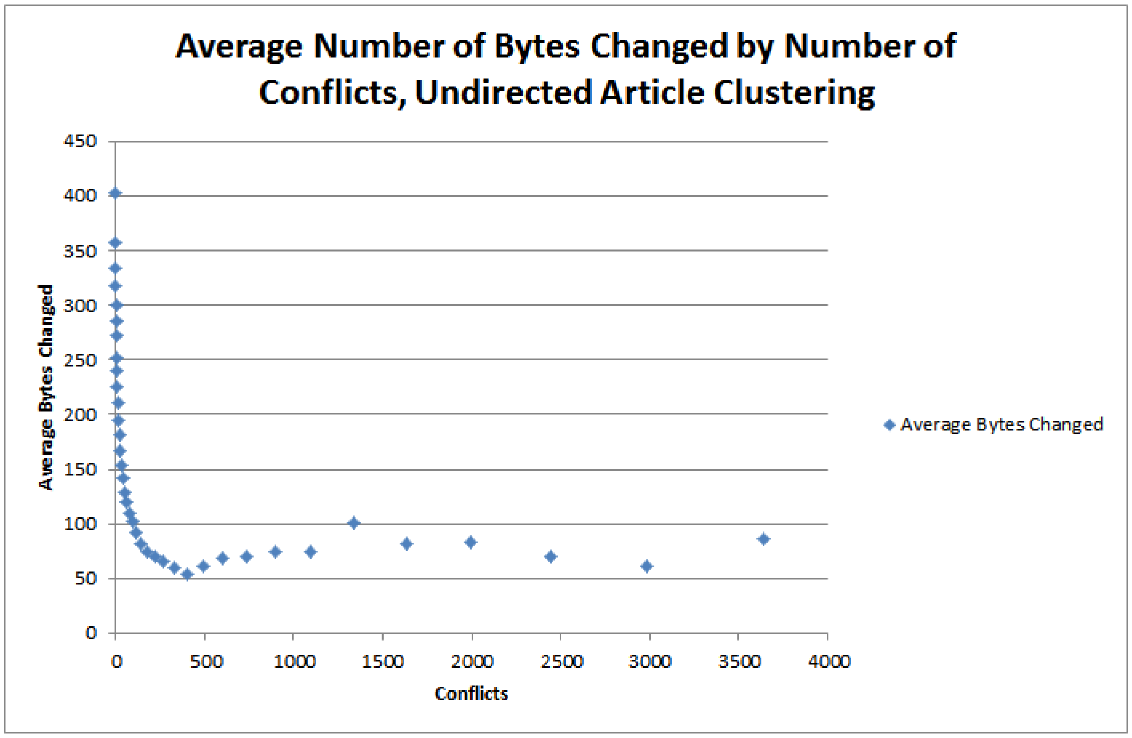


Figure 6.5: Average bytes changed by number of conflicts in an article cluster. The data were binned to the nearest power of $e^{0.2}$, with a shift of one to account for article clusters with zero conflicts. All higher values of conflicts were binned to $e^{8.2} - 1$.

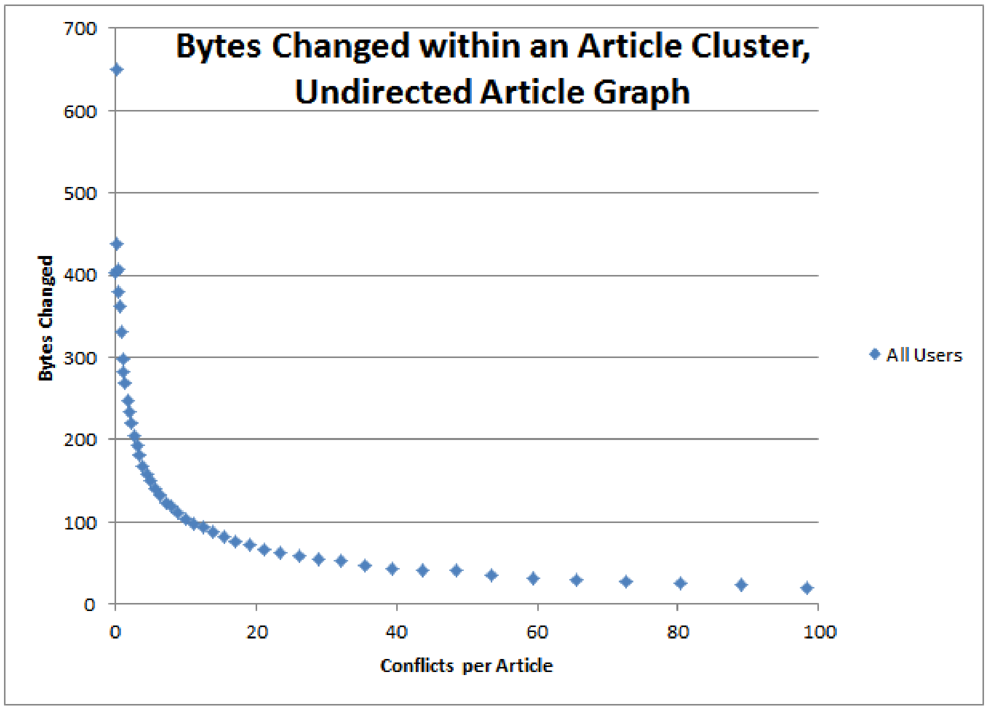


Figure 6.6: Average bytes changed by number of conflicts per article in an article cluster. The data were binned to the nearest power of $e^{0.1}$, with a shift of one to account for article clusters with zero conflicts. All higher values of conflicts per article were binned to $e^{4.6} - 1$.

7

Conclusion

In this paper I introduced a new method for finding conflicts in Wikipedia that span multiple articles and characterized several dimensions of these conflicts, including how many reverts and users were involved, how many articles they occured in, their impact upon the editing habits of users, and their bipartiteness.

I examined conflicts at the level of topic-based article clusters, which I created by using an algorithm based upon random walks on a graph of Wikipedia articles and the intra-Wikipedia links between them. I found conflicts within these article clusters as opposed to within single articles. I observed the size and form of conflicts in Wikipedia given this data. Based upon information about the reverts, I determined which articles each conflict existed in and observed how frequently conflicts spanned multiple articles. I also obtained data about the number of bytes contributed to each article cluster by every user within the cluster, and observed some effects of conflict upon editing habits.

These analyses lead to make several conclusions. The vast majority of conflicts are very small, but there are still thousands of conflicts involving at least one hundred users. Conflicts between small numbers of users, or with small numbers of reverts, tend to span only one article, whereas larger conflicts tend to span more than one article. My results suggest that research focusing on conflicts of a substantial size, perhaps 30 users or larger, could be substantially improved by allowing conflicts to span multiple articles.

Conflict within an article cluster is associated with smaller average contributions from users within that cluster. Additionally, contributions from users uninvolved in conflicts are even lower than those involved in conflicts. This indicates that users may be deterred from contributing to areas with high concentrations of conflict.

Lastly, most conflicts are trees. Among the rest, the number that are not bipartite is small. This suggests that most conflicts are not very intense and that conflicts in Wikipedia usually have two sides and not more. Conflicts that are not trees tend to involve more users.

8

Bibliography

[1] U. Brandes, D. Fleischer, and J. Lerner. Summarizing dynamic bipolar conflict structures. *IEEE Transactions on Visualization and Computer Graphics*, 12(6), 2006.

[2] U. Brandes, P. Kenis, J. Lerner, and D. van Raaij. Network analysis of collaboration structure in wikipedia. *World Wide Web Conference*, 2009.

[3] U. Brandes and J. Lerner. Revision and co-revision in wikipedia: Detecting clusters of interest. *European Semantic Web Conference*, 2007.

[4] D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. *Proceedings of the 31st Very Large Data Bases Conference*, 2005.

[5] J. Giles. Internet encyclopaedias go head to head. *Nature*, 438:900–901, 2005.

[6] R. Jesus, M. Schwartz, and S. Lehmann. Bipartite networks of wikipedia's articles and authors: a meso-level approach. *International Symposium on Wikis and Open Collaboration*, 2009.

[7] A. Kittur, E.H. Chi, and B. Suh. What's in wikipedia? mapping topics and conflict using socially annotated category structure. *ACM Conference on Human Factors in Computing Systems*, 2009.

[8] A. Kittur, B. Suh, B.A. Pendleton, and E.H. Chi. He says, she says: Conflict and coordination in wikipedia. *ACM Conference on Human Factors in Computing Systems*, 2007.

[9] J. Kunegis, A. Lommatzsch, and C. Bauckhage. The slashdot zoo: Mining a social network with negative edges. *World Wide Web Conference*, 2009.

[10] D. Laniado, R. Tasso, Y. Volkovich, and A. Kaltenbrunner. When the wikipedians talk: Network and tree structure of wikipedia discussion pages. *International AAAI Conference on Weblogs and Social Media*, 2011.

[11] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. *ACM Conference on Human Factors in Computing Systems*, 2010.

[12] D. Lizorkin, O. Medelyan, and M. Grineva. Analysis of community structure in wikipedia (poster). *World Wide Web Conference*, 2009.

[13] S. Sahni and T. Gonzalez. P-complete approximation algorithms. *Journal for the Association of Computing Machinery*, 25(3):555–565, 1976.

[14] S. E. Schaeffer. Graph clustering. *Computer Science Review 1*, pages 27–64, 2007.

[15] K. Steinhaeuser and N.V. Chawla. Identifying and evaluating community structure in complex networks. *Pattern Recognition Letters*, 2009.

[16] C. R. Sunstein. The law of group polarization. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=199668, 1999.

[17] M. W. Van Alstyne and E. Brynjolfsson. Global village or cyberbalkans: Modeling and measuring the integration of electronic communities. *Management Science*, 52:851–868, 2005.

[18] V. Zlatic, M. Bozicevic, H. Stefancic, and M. Domazet. Wikipedias: Collaborative web-based encyclopedias as complex network. *Phys. Rev. E*, 74, 2006.