Arc Diagrams: Visualizing Structure in Strings

Martin Wattenberg
IBM Research
One Rogers Street
Cambridge MA 02142
mwatten@us.ibm.com

Abstract

This paper introduces a new visualization method, the arc diagram*, which is capable of representing complex patterns of repetition in string data. Arc diagrams improve over previous methods such as dotplots because they scale efficiently for strings that contain many instances of the same subsequence. This paper describes design and implementation issues related to arc diagrams and shows how they may be applied to visualize such diverse data as music, text, and compiled code.*

Keywords: string, sequence, visualization, arc diagram, music, text, code

1. Introduction

From text to DNA to melodies, many data sets come in the form of a string, or sequence of symbols. Just as with quantitative data, it is often desirable to perform graphical exploratory analysis on a string to find important structural features.

One way to reveal a string's structure is to exploit the fact that sequences often contain significant repeated subsequences. Melodies, for instance, are usually based on combinations of smaller repeated musical passages; text has repeated words and phrases. A natural way to visualize structure is to use these repeated units as signposts.

Several existing methods use repetition to visualize string structure, but each has significant drawbacks for complex strings. In this paper we introduce the *arc diagram*, a new visualization method for representing sequence structure by highlighting repeated subsequences. We describe how arc diagrams can find patterns in text, compiled code and, most fruitfully, in musical compositions.

2. Existing methods for string visualization

Many methods have been proposed to display string structure visually. The H-Curve and W-Curve [HR83, W93] transform sequences into curves in 3D space. Although such curves are capable of showing fine detail, they can be hard to interpret and it can be difficult to spot smaller repeated substrings. The "Chaos Game" representation of a sequence [J90], in effect a 2D histogram depicting the frequencies of various motifs, is efficient for showing which small substrings are frequently repeated, but can run into difficulties distinguishing long subsequences with similar beginnings. Moreover, chaos game representations remove much ordering information, making them unsuitable for domains where ordering matters.

Another method, popular in analyzing DNA sequences, is the dotplot [CH92]. A dotplot is a visual auto-correlation matrix; in its simplest form, a string of n symbols $a_1a_2\ldots a_n$ is represented by an n x n image in which the pixel at coordinates (i,j) is colored black if $a_i=a_j$ and white otherwise. This image often provides a good picture of the string's structure, with repeated subsequences showing up clearly as diagonal lines. In many respects dotplots are an excellent visualization method: They can handle very large data sets, are resistant to noise, and can show both small and large-scale structures. However, the matrix-style presentation of a dotplot means that if a substring is repeated n times, it will give rise to n^2 corresponding visual features. As a result, dotplots can be confusing when applied to strings with frequently repeated substrings.

One non-visual method of describing a long string is to summarize it by describing which subsequences are repeated. For instance, musicians have long described the global structure of musical compositions by summaries such as "AABB" (meaning a subsequence, denoted by A, is repeated and then followed by a different subsequence, B, that is also repeated.) This simple symbolic notation is easy to understand and provides a broad overview of the data, but obliterates smaller details.

It is natural to seek a visual analogue of this notation. Music theorists, starting with Heinrich Schenker, have used a system of hand-drawn arcs to indicate structural units (see, for example, [S69]). However Schenkerian diagrams, which are intrinsically subjective and manual, are unsuitable for automation or for showing features on multiple scales. One commercial software package, TimeSketch [T02], uses half-disks to delineate different sections of a piece, coloring related passages with the same color to indicate musical form. The TimeSketch software requires human definition of related passages, and because it uses color for differentiation does not scale well for sequences that have many different related passages.

3a. The arc diagram

This paper introduces the *arc diagram*. An arc diagram generalizes the musical AABB notation by using a pattern-matching algorithm to find repeated substrings, and then representing them visually as translucent arcs. Unlike a TimeSketch diagram, an arc diagram can be constructed automatically and can represent the structure of a sequence with many different repeated subsequences and multiple scales of repetition. Unlike a dotplot, it can efficiently represent sequences where individual subsequences are repeated many times.

An arc diagram is built around the idea of visualizing only a subset of all possible pairs of matching substrings. By choosing to highlight just the subsequences essential to understanding the string's structure, the method can convey all critical structure while avoiding the quadratic scaling problem of a dotplot. We now define these "essential" substring pairs for a given string *S*.

Definition 1. A *maximal matching pair* is a pair of substrings of *S*, *X* and *Y,* which are:

1. *Identical*. *X* and *Y* consist of the same sequence of symbols.
2. *Non-overlapping*. *X* and *Y* do not intersect.
3. *Consecutive*. *X* occurs before *Y*, and there is no substring Z, identical to X and Y, whose beginning falls between the beginning of X and the beginning of Y.
4. *Maximal*. There do not exist longer identical non-overlapping subsequences *X*' and *Y*' with *X*' containing *X* and *Y*' containing *Y*'.

For example, in the sequence "123a123", the two "123" substrings form a maximal matching pair, but the two "12" substrings do not.

It is tempting to base a visualization method on maximal matching pairs alone, but an awkward situation arises when a pattern is repeated many times in immediate succession. For instance in the string 10101010, the only maximal matching pair consists of the first and last "1010" substrings, implying that the string has two main structural components. In a sense, this division into two large substrings is spurious; it would be more accurate to describe the string as composed of four small repeated units. This is the motivation for the following two definitions.

Definition 2. A *repetition region R* is a substring *R* of *S* such that *R* is made up of a string *P* repeated two or more times in immediate succession. Each repetition of P is called a *fundamental substring* for *R*.

For example, in the string ABC010101, the substring "010101" is a repetition region. Each of the "01" substrings is a fundamental substring.

The next definition specifies the precise set of substrings that will be used to construct an arc diagram.

Definition 3. An *essential matching pair* is a pair of substrings of *S*, *X* and *Y,* which are:

1. A maximal matching pair not contained in any repetition region,
2. *Or*, a maximal matching pair contained in the same fundamental substring of any repetition region that contains it,
3. *Or,* two consecutive fundamental substrings for a repetition region.

We are now ready to define the arc diagram for a string *S* of length *N*. First, define a mapping from the string to the *x*-axis, with the position of the *m*th symbol at the point (*m/N*,0). Under this mapping, a substring *T* of *S* corresponds to an interval on the *x*-axis. Now, for each essential matching pair (*X*,*Y*) in the string, connect the corresponding intervals on the line with a thick semi-circular arc (Figure 1). Precisely, the interior semi-circle connects the end of the interval for *X* with the beginning of the interval for *Y*, and the exterior semi-circle connects the beginning of the interval for *X* with the end of the interval for *Y.* The height of the resulting arc is thus proportional to the distance between the two substrings.

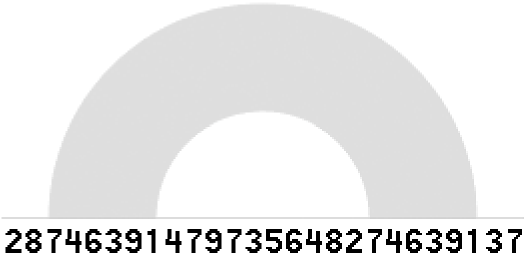

Figure 1. Connecting an essential matching pair

The arc makes it obvious where the repeated subsequences are. (By comparison, imagine finding this repetition without the arc as a cue; it would be laborious.)

Because of the “consecutive” condition of Definition 1, if a particular subsequence is repeated more than once, the diagram connects only consecutive repetitions with an arc. (See Figure 2.) The fact that only consecutive pairs are connected rather than every possible pair is what allows arc diagrams to scale more efficiently than dotplots: when a subsequence is repeated *n* times, an arc diagram will contain *n*-1 arcs, while a dotplot would display n^2 diagonal lines. (See section 3b for an example.)

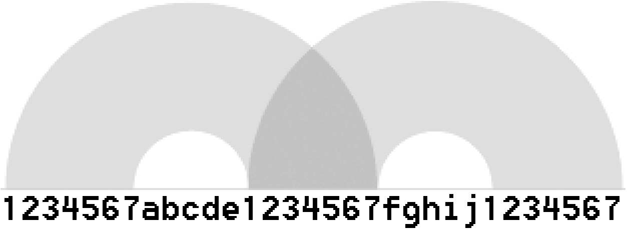

Figure 2. A substring repeated three times

As a simple example of visualizing strings with repetition regions, take the sequence 1010101010101010. Here the only essential matching pairs are those that satisfy part 3 of Definition 3—that is, they are the fundamental substrings of the form “10”. The diagram for this sequence would look like the picture in Figure 3.

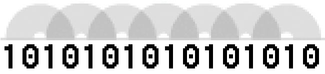

Figure 3. Immediate repetition

Most sequences will contain several different repeated subsequences, including overlapping sequences at multiple scales. To make a diagram for such a sequence, we overlay all the appropriate arcs with a degree of translucency so no match is completely obscured. For instance the sequence abcd111110000011111abcd produces the diagram in Figure 4:

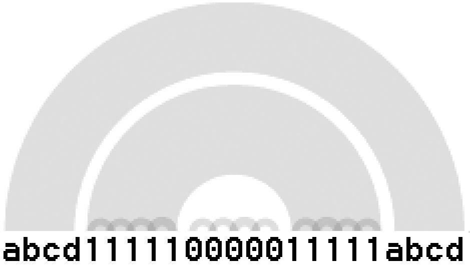

Figure 4. Repetition of different substrings

In Figure 4 we begin to see how a pattern of matches can provide a bird's-eye view of the sequence's structure. It is visually obvious that at the macro level the sequence is symmetric. The translucency further reveals a highly repetitive substructure, without interfering with macro-level interpretation. This technique is similar to that used in Jerding and Stasko’s Information Mural [JS95].

A long sequence made from a small set of symbols will always contain many small repeated sequences, which may be of no significance. Worse, the arcs connecting these small sequences may obscure significant large-scale repetitions. Figure 5 shows an example where too many small repeated subsequences cause an uninformative jumble.

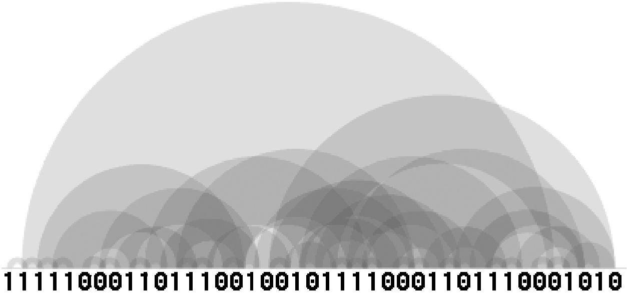

Figure 5. Too much detail

One way of reducing this complexity is to filter by subsequence length, displaying only repeated subsequences that are longer than a given limit. For instance, Figure 6 shows the same sequence but this time set to filter out repeated subsequences of fewer than 10 symbols. The result is a simple diagram that highlights a single repeated region of 15 symbols—the kind of large-scale repetition that is unlikely to occur by chance.

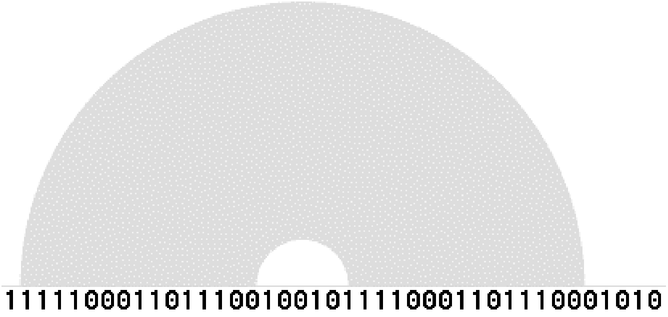

Figure 6. Displaying only large-scale repetition

3b. Comparison with a dotplot

As mentioned above, the reason we do not connect every possible matching subsequence is so that the resulting diagram scales efficiently from a visual perspective. To see how this works, consider two visualizations of the same string, a dotplot (Figure 7) and an arc diagram (Figure 8). The string visualized contains many repetitions of two substrings, which results in considerable visual clutter in the dotplot. Two other substrings are each repeated once, a feature that is difficult to spot in the dotplot. The arc diagram, however, shows all the repeated structures clearly.

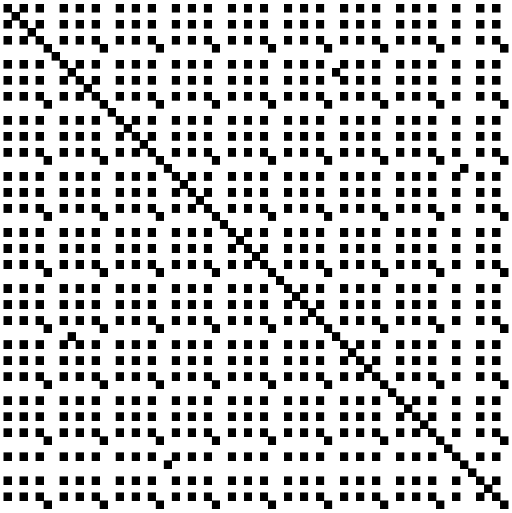

Figure 7. Dotplot of a synthetic sequence

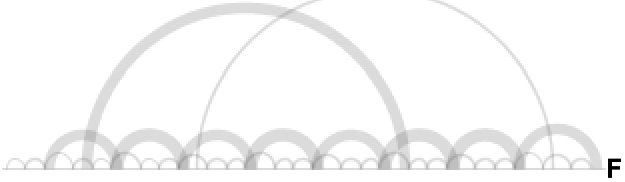

Figure 8. Arc diagram, same sequence as Fig. 8

3c. Implementation

The program used to create the arc diagrams in this paper is written in Java, runs efficiently on a low-end (266 Mhz Pentium II) machine, and can create diagrams of sequences of several thousand symbols within seconds. To enumerate repeated patterns, a suffix tree is constructed and traversed twice. In the first pass, repetition regions are identified and in the second pass potential matching substring pairs are tested to see whether they are essential according to the criteria of Definition 3. The arc diagram code has also been used in an online applet, "The Shape of Song" [W01], that allows users to create arc diagrams for any MIDI format music file available on the web.

4. Applications to music

One of the most promising applications of arc diagrams is to reveal structure in musical compositions. An example of a musical arc diagram is shown in Figure 9, which represents the first line of the song *Mary Had a Little Lamb*.

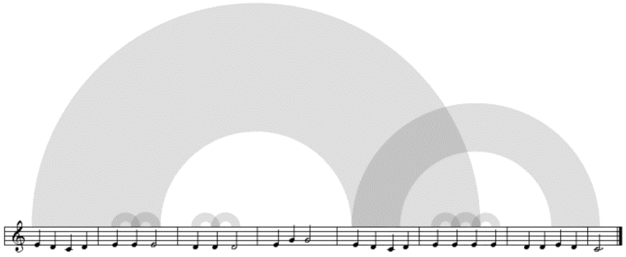

Figure 9. Arc diagram for music

Each arc connects two matching passages, where a "match" means that they contain the same sequence of pitches. The diagram shows repeated subsequences of three or more notes. To clarify the connection between the visualization and the song, I have displayed the score beneath the arcs.

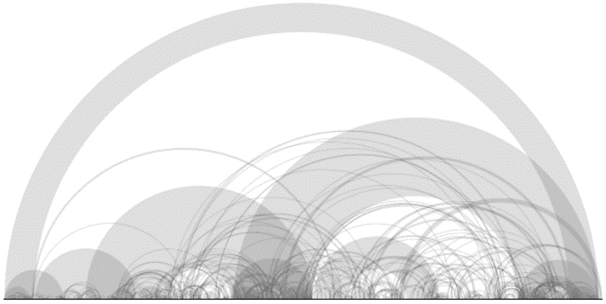

Figure 10. Arc Diagram of ***Für Elise***

Figure 10 visualizes Beethoven's *Für Elise*. (In this and subsequent diagrams, the source sequences are too long to display legibly.) Again, matches are based on equality of pitch; where chords occur we consider only the top note. Despite this extremely limited definition of musical similarity, the resulting matching diagram reveals an intricate and beautiful structure. The picture shows how the piece begins and ends with the same passage, while a longer version of that passage repeats throughout at increasing intervals. Equally illuminating is the long stretch in the second half of the piece where that passage is not repeated at all and the structure looks distinctly different, which corresponds well to what you hear when you listen to the music.

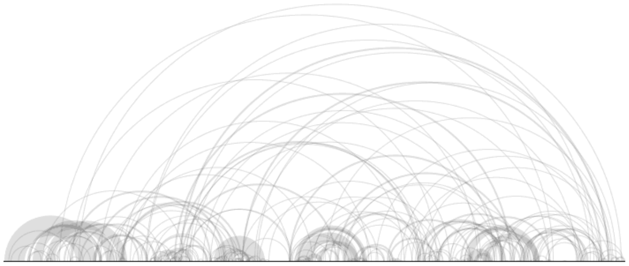

Figure 11. Toreador, Carmen

Not all pieces show as much large-scale repetition as *Fur Elise*. For instance, the "Toreador" song from Carmen (Figure 11) looks completely different. Instead of a few long passages repeated over and over again, it contains many repeated smaller phrases.

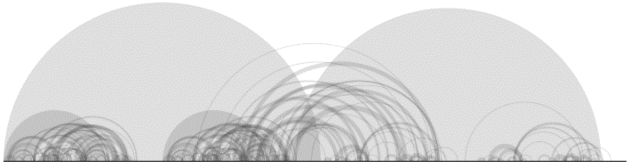

Figure 12. Minuet in G Major, Bach

As a final example, consider Bach's *Minuet in G Major* (Figure 12). The arc diagram shows that the piece divides into two main parts, each made of a long passage played twice: what a musician would call an "AABB" structure. AABB is, in fact, the classic structure of a minuet, which shows that the matching algorithm is picking out structures that correspond to conventional musical analysis. The pictorial representation, however, provides much more detailed information than the simple "AABB" notation. For instance, you can see that the A and B passages are loosely related, as shown by the bundle of thin arcs connecting the two halves of the piece. And the fact that the two main arcs overlap shows that the end of the A passage is the same as B's beginning.

For musical compositions it is natural to consider the sequence of differences between successive notes as well as the notes themselves. Figure 16 (at the end of the paper) shows two arc diagrams for *Für Elise*, juxtaposed: the top is a large version of Figure 10 and the bottom, flipped diagram shows additional matching substrings based on intervals between successive notes.

5. Finding structure in text and compiled code

Arc diagrams are well suited to the analysis of highly structured data such as musical compositions, but they also can be effective in exploring other less well-structured data. Three examples we consider are compiled computer code, a web page, and a nucleotide sequence from DNA.

To see how matching diagrams can find. structure in sequences that might otherwise be difficult to decipher, consider Figure 13, which shows the bytecode for the main Java class of the diagram-generating application itself:

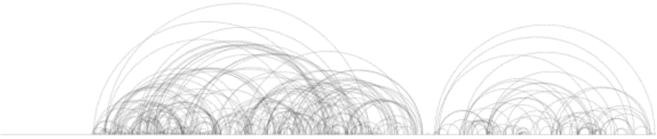

Figure 13. Java class file (bytecode)

The diagram clearly shows that the file has two main sections. This reflects the important piece of structural information that Java class files are divided into two main sections: the constant pool and the executable code. Moreover, the diagram shows that the initial section (the constant pool) takes up significantly more memory than the code itself. Thus the diagram provides both structural and quantitative information that would be difficult to discern from a standard "hex-dump" text view of the file.

When arc diagrams are applied to textual data, they can also produce useful results. For example, an arc diagram of a short HTML file resulted in the picture in Figure 14.

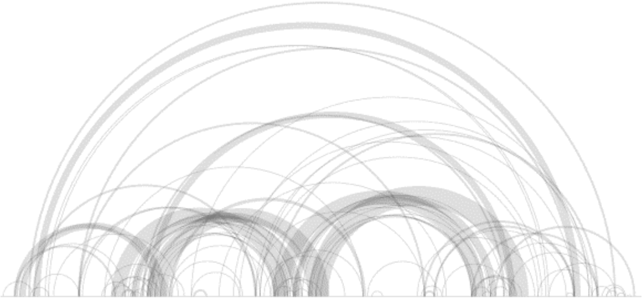

Figure 14. HTML page

The page was organized into three basic sections, a fact which was delineated clearly by wide arcs. (These correspond to the HTML code for images and tables.) At the same time, the finer-grained detail is also revealing. For instance, the diagram shows that the last section of text has many connections to previous parts of the page, with especially strong connections to the beginning; this indicates that the introduction and conclusion of the text contained similar phrases and themes.

Finally, it is natural to apply matching diagrams to DNA nucleotide sequences. One potential pitfall is that DNA is noisy data in the sense that exact repetition on a large scale is uncommon due to mutations. In some situations, however, this is not a problem. For example, there is significant interest in understanding patterns in upstream transcription regions (UTRs), i.e. the subsequences of DNA that precede regions that code for genes. The distribution of certain small (typically around 7 base pairs) subsequences called *motifs* in a gene's UTR is thought to play a key role in regulating that gene [C00].

Figure 15 shows an arc diagram for a UTR of length 500 for a particular yeast gene (identifier YGL123C in [S02]), filtered to show repeated patterns of 7 or more symbols. Although not as dramatic as the music diagrams, this picture does contain interesting information. For example, it shows that one special region of the UTR (from roughly 200 steps before the end to 100 steps before the end) contains at least one instance of most of the repeated patterns. This is potentially related to a recent finding [H00] that many regulatory motifs are more likely to appear in this same restricted region.

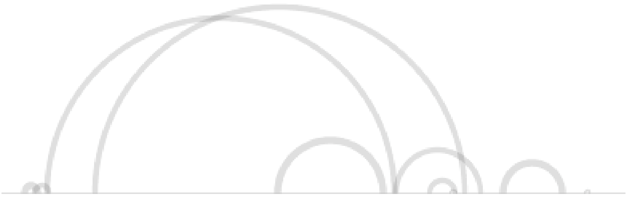

Figure 15. DNA sequence

6. Summary and directions for future work

Arc diagrams are a promising new method for visualizing sequences. They are well suited to displaying structure in sequences that contain complex patterns of repetition. We have shown examples of their potential use in domains ranging from text to DNA, although analysis of musical form is perhaps the most promising application.

Many areas remain for future exploration. One key direction of exploration is the best way to add interactivity to the diagrams. The visualizations described in this paper are static; an interactive version could be more powerful. One natural extension would be to add sliders to control the level of detail, allowing the user to specify how large a subsequence would need to be in order for an arc to be drawn. In addition, users could be allowed to drill down for details. For example, if the user pointed at a particular arc, the subsequence corresponding to that arc could be drawn on screen. If the underlying sequence were a musical composition, that particular passage could be played.

Another area for future investigation is the use of alternative pattern matching algorithms. Instead of drawing arcs between substrings that are identical, one could choose a more flexible criterion. For example, when diagramming a fugue the criterion for a match might include transpositions and inversions as well as identical repetition. In addition, by using "fuzzy" matching techniques, it might be possible to make the method more useful for noisy data, such as DNA sequences.

A final area to explore is the incorporation of additional variables into the visualization. One might use different hues to indicate substrings that match according to different criteria. Another technique for adding information would be to incorporate a notion of intensity (e.g., corresponding to volume in a musical composition) and draw arcs with a translucency factor corresponding to the intensity.

7. References

[C00] Cooper, Geoffrey. *The Cell: A Molecular Approach, 2nd ed*. Sinauer Assoc. 2000.

[CH92] Church, K.W., and Helfman, J.I. "Dotplot: A Program for Exploring Self-Similarity in Millions of Lines of Text and Code", *Proceedings of the 24th Symposium on the Interface*, Computing Science and Statistics V24, pp. 58-67, March, 1992.

[H00] Hughes, Jason *et al*., “Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae.” *Journal of Molecular Biology* (2000) 296: 1205-124.

[HR83] E. Hamori and J. Ruskin, "H-Curves, a novel method of representation of nucleotide sequences especially suited for long DNA sequences," *Journal of Biological Chemistry*, 258 (2):1381-1327, 1983.

[J90] H. J. Jeffrey. "Chaos game representation of gene structure." *Nucleic Acids Research*, 18(8):2163-2170, 1990.

[JS95] D. Jerding and J. Stasko. "The Information Mural: A technique for displaying and navigating large information spaces." *IEEE Visualization `95 Symposium on Information Visualization*, pp 43-50.

[S69] Schenker, Heinrich, *Five graphic music analyses*, New York: Dover, 1969

[S02] *Saccharomyces genome database*, http://genome-www.stanford.edu/Saccharomyces/ (2002)

[T02] "TimeSketch." *ECS Media*. www.ecsmedia.com 2002

[W01] M. Wattenberg, "The Shape of Song." http://www.turbulence.org/works/Song (2001)

[W93] D. Wu, J. Roberge, D. J. Cork, B. G. Nguyen and T. Grace, "Computer visualization of long genomic sequences." *IEEE Visualization '93*, pp. 308-315.

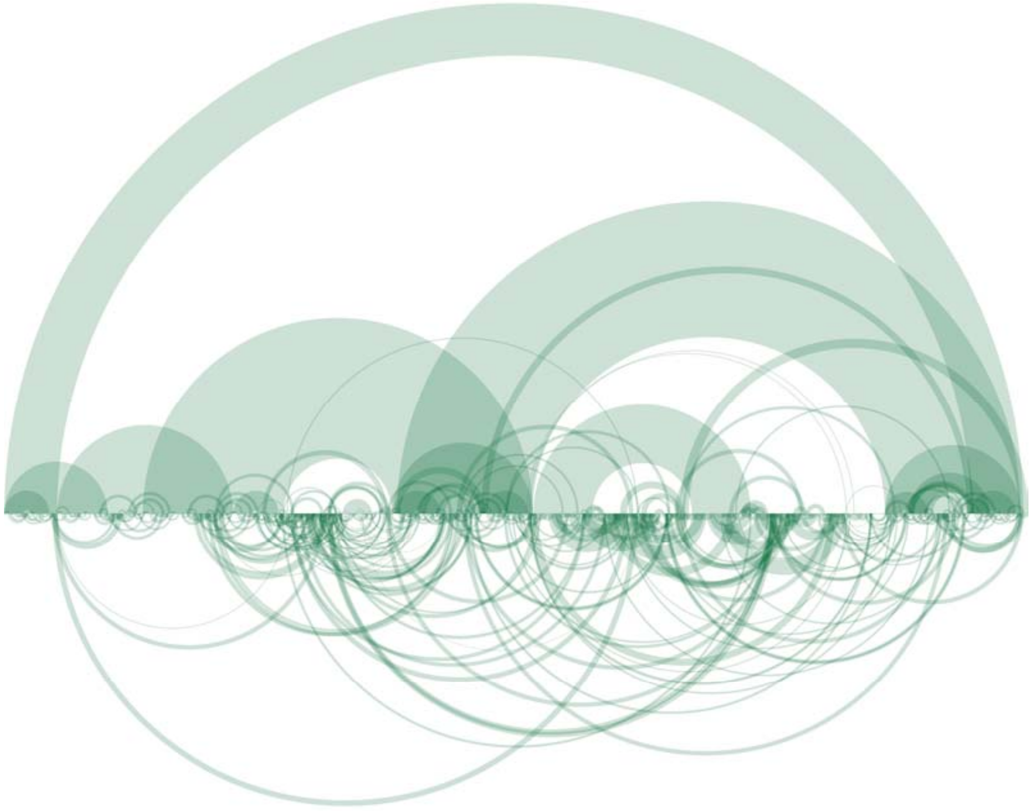

Figure 16. ***Für Elise*****, exact (top) and modulated (bottom) matches**