Pattern Matching for the Tracking of Melodic Development

Categories: Computer-Aided Musicology,
Music Data Structures and Representation,
Musical Databases and Data Mining

Paper Type: Discussion

Authors

Adam Tee[*+], David Cooper[*#], Nicholas J Bailey[†] and Des Mclernon[*+]
Email: Adam Tee {eenajt@electeng.leeds.ac.uk}

∗ Interdisciplinary Centre for Scientific Reasearch in Music
University of Leeds
LS2 9JT, UK

+ School of Electronic and Electrical Engineering
University of Leeds
LS2 9JT, UK

∗ Department of Music
University of Leeds
LS2 9JT, UK

† Centre for Music Technology
Department of Electronics and Electrical Engineering
The Rankine Building
The University of Glasgow
Glasgow, G12 8LT

Pattern Matching for the Tracking of Melodic Development

Adam Tee[*+], David Cooper[*#], Nicholas Bailey[†] , Des Mclernon[*+]

[*] Interdisciplinary Centre for Scientific Reasearch in Music, University of Leeds, LS2 9JT, UK

[+] School of Electronic and Electrical Engineering, University of Leeds, LS2 9JT, UK

[#] Department of Music, University of Leeds, LS2 9JT, UK

[†] Centre for Music Technology, Department of Electronics and Electrical Engineering, The Rankine Building, The University of Glasgow, Glasgow, G12 8LT

Abstract

In this paper we produce a synopsis of the currently used pattern matching techniques, discuss their features and propose additional musical features that should be considered when analysing melodies. Musical features considered include articulations, phrasing and dynamics.

1 Introduction

Over recent years extensive research has been carried out on the automatic detection of similar melodies in musical compositions. The research falls into two main application areas: *database queries* and *melodic analysis.* Database queries have so far usually involve a corpus of one particular genre of music, the most common being folk songs, because versions are often found with similar melodies. Melodic analysis has been applied to single works and looks at how a particular melody is varied and developed. Here we are interested in the techniques used to find melodic similarity with the application to melodic analysis. The most widely used technique in melodic similarity is pattern matching.

The representation of the musical score is important no matter which application is being used. The majority of existing approaches treat the score as an abstract sequence. By abstract sequence we mean that the score is treated as a simple sequence of notes with very little musical information is retained.

Pitch letter names and octave transposition identifiers are rarely used in the matching as they do not allow for musical techniques such as transposition. Instead, the contour of the melody is used. Various music perception experiments (Edworthy, 1985; Dowling, 1978) have shown that the contour of a melody is remembered better than the exact pitches used in the melody.

Selfridge-Field (Selfridge-Field, 1998) does not propose a single standard representation, because the requirements vary according to the application under development. The most widely-used representations of pitch contour are either a simple three letter description of the contour, or an intervallic distance representation. The first representation uses the characters **U**,**D**,**S**, (up, down, same), to represent the pitch difference between adjacent notes in the melody. The second method represents the interval in semitones between adjacent notes in the melody. As an alternative, modulo intervallic distance, which is designed to keep all intervals within the bounds of an octave, can be used. Figure 1 shows the two types of pitch contour for the main theme of the first movement of Mozart's 40th Symphony in G Minor K.550. Rhythmic contour is harder to specify as it depends upon the representation used for the duration of the notes, most applications representing durations in terms of the number of semiquavers that a note is equal to.

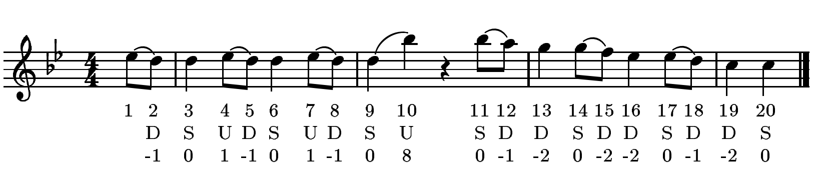


Figure 1: Pitch Contour Example

In the following sections, the most widely used pattern matching techniques are introduced with a discussion of results obtained using the first method. Then, the additional musical information that should be considered is introduced and discussed.

2 Pattern Matching Techniques

There are two main types of pattern matching,*exact matching* and *approximate matching*. Exact matching looks for exact replicas of an input pattern, whereas approximate matching allows for a certain number of differences between the pattern and the matching portion of the text. As approximate matching algorithms allow for exact matches to be found, algorithms which only allow exact matches are not normally used. Crawford et al. (Crawford et al., 1998) propose 13 different areas of analysis that could be used with various pattern matching algorithms for their solution.

2.1 Longest Common Subsequence

The most common approach used is to find the *longest common subsequence*, using the Dynamic Programming (DP) algorithm. This method was first used in text processing by Wagner and Fischer (Wagner and Fischer, 1974) and subsequently adapted by Sellers, for DNA sequences, Ukkonen (Ukkonen, 1985a; Ukkonen, 1985b) and others. For musical sequences, this approach was first used by Mongeau & Sankoff (Mongeau and Sankoff, 1990). It has subsequently been used by Smith et al (Smith et al., 1998), Uitdenbogerd et al (Uitdenbogerd and Zobel, 1999) and proposed as a possible solution by Crawford et al (Crawford et al., 1998).

The basic method is to calculate an $(m+1) \times (n+1)$ table, D, of edit distances between the pattern of length m, and text of length n. In this application the pattern is the melody being searched for and the text an entire composition or melodic line. the edit or Levenshtein distance is the number of changes required to change one string to another. The most common types of edit are:

- insertion - inserting a character into the pattern: $\emptyset \to b$

- deletion - removing a character from the text: $a \rightarrow \emptyset$
- change - changing a character in the pattern: $a \rightarrow b$

where $\emptyset$ represents an empty character.

The values of the table, D, are calculated using the following equations:

$$\begin{aligned}
D(0,0) &= 0 && && (1)\\
D(i,0) &= 0 && 1 \leq i \leq n && (2)\\
D(0,j) &= j && 1 \leq j \leq m && (3)\\
D(i,j) &= \min \begin{cases} D(i-1,j) + \mathbf{w}(a \rightarrow \emptyset) & 1 \leq i \leq n \\ D(i,j-1) + \mathbf{w}(\emptyset \rightarrow b) & 1 \leq j \leq m \\ D(i-1,j-1) + \mathbf{w}(a \rightarrow b) & \end{cases} && && (4)
\end{aligned}$$

where $\mathbf{w}()$ is a weighting given to the different type of edit operation. To find a match the table D is traversed column by column until a value $D(i,j)$ is less than a given threshold value. The threshold is the maximum edit distance that the pattern can be from the portion of text being matched.

Mongeau & Sankoff and Smith et al. add the following two edit operations: *fragmentation*, which is the replacement of one element in the string by many; and *consolidation*, which is the opposite of Fragmentation. These edit operations allow one to search for melodies inside a set of variations, where embellishments have replaced the original melody.

One of the disadvantages of this method is the running time of the algorithm when used with large patterns, entire compositions and databases. The reason for this is that the table D gets very large and usually requires $O(mn)$ time to calculate, where m is the length of the pattern and n is the length of the text.

Figure 2 represents an edit distance table with all weightings equal to one. Only insertion, deletion and change are used. The pattern used is the first three notes of the Mozart theme (Figure 1) and the text is the first half, up to the first B♭ of Figure 1.

		0	-1	0	1	-1	0	1	-1	0	8
	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	1	0	1	1	0	1
-1	2	1	0	1	1	1	1	1	1	1	1
0	3	2	1	0	1	2	1	2	2	1	2

Figure 2: Sample Edit Distance Table

2.2 Longest Common Substring

Another technique is to find the *longest common substring* between the pattern and the text. Uitdenbogerd et al. use this as one of the methods in their work. Cope (Cope, 1991; Cope, 1990) uses a similar method in his work, where small 2 or 3 bar phrases are used as patterns with like strings forming the basis of stylistic features. Uitdenbogerd et al. use the pitch contour representation in this method.

Uitdenbogerd et al. use two slightly different methods in their solution of this problem. Both are based on the concept of n-grams (Ukkonen, 1992), n-length substrings of the pattern and the text.

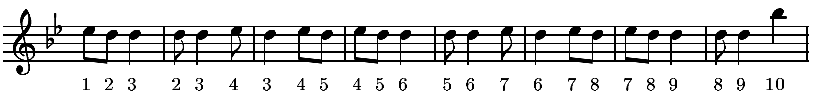


Figure 3: n-Grams Example

Figure 3 shows the set of n-grams, of length 3, for the first half of the opening theme of the first movement of Mozart's 40th Symphony K.550. In total there are 8 n-grams, for a pattern of 10 notes.

The first method simply counts the number of occurrences of the n-grams that are common to both the pattern and text. The n-gram with the greatest number of occurrences is then the most common substring. In Figure 3, the 3-gram 'E♭ D D' occurs the most frequently and is the most common substring of length 3.

The second method uses the *Ukkonen Distance Measure*,

$$\sum_{v \in \sum^n} |G(x)[v] - G(y)[v]| \tag{5}$$

where $\sum^n$is the set of possible n-grams, x and y are the strings being compared, $G(x)[v]$ is the number of occurrences of n-gram v in x, and $G(y)[v]$ is the number of occurrences of n-gram v in y.

2.3 Local Alignment

The final method is *local alignment*, used by Uitdenbogerd et al. and Mongeau & Sankoff. This method is analogous to the DP method in 2.1 except that an alignment function is used and subsequently maximised . The alignment function is of the following form:

$$D(i,0) = 0 \qquad \forall i \tag{6}$$

$$D(0,j) = 0 \qquad \forall j \tag{7}$$

$$D(i,j) = \max \begin{cases} D(i-1,j) + \mathbf{w}(a_i, \emptyset) & 1 \le i \le n \\ D(i-1,j-1) + \mathbf{w}(a_i, b_j) & 1 \le j \le m \\ D(i,j-1) + \mathbf{w}(\emptyset, b_j) & \\ 0 & \end{cases} \tag{8}$$

where the explanation of 2.1 applies.

Figure 4 shows the local alignment table for the pattern , 'E♭ D D' and the first half of the theme in Figure 1. The values used for the weightings are: -2, for insertions and deletions, -1, for a change, and 1 for a match.

		0	−1	0	1	−1	0	1	−1	0	8
	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1	0	0	1	0
−1	0	0	2	0	0	1	0	0	1	0	0
0	0	1	0	3	1	0	2	0	0	2	0

Figure 4: Sample Local Alignment Table

The maximum alignment is found by tracing a path through the table. As with the longest common subsequence method the table requires $O(mn)$ time to calculate. Mongeau & Sankoff traced from the maximum to a nil value to find the string with the best local alignment. They also added fragmentation and consolidation to the basic solution.

A problem with this method is that the maximum may occur only once in the table but when the actual diatonic pitches are looked at their may be more than one match, as is the case with Figure 4. To find approximate matches a threshold value should be used, with the trace starting at any value above the threshold.

3 Discussion

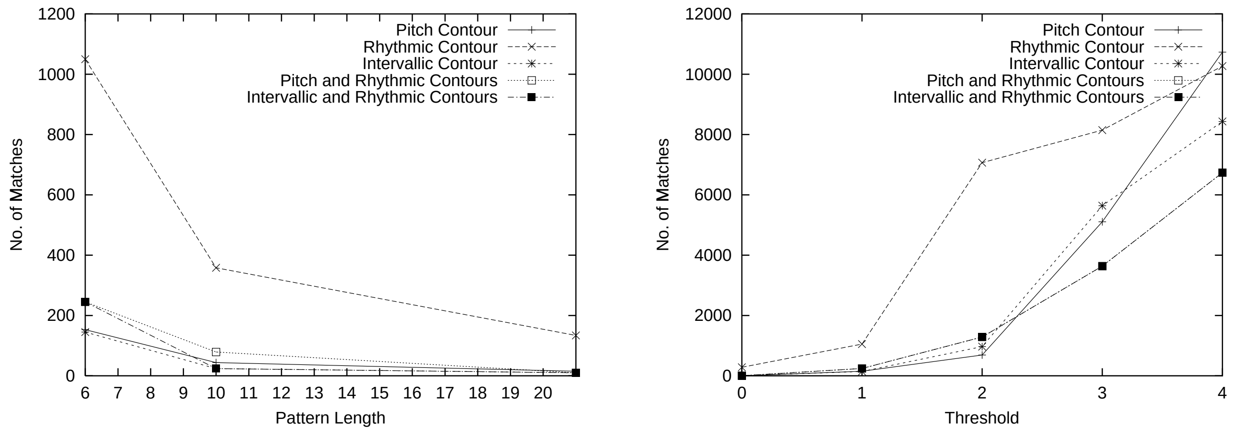
Of all the methods described above, the DP approach appears to be the most popular. Reasons for its popularity include the fact that it is fairly efficient, and that it also allows for variation in the threshold and weightings used. This enables the different edit operations to have greater significance in the matching process. In melodic analysis this may offer an insight into the elaborations that a composer makes in the development of a melody.

The work that has been carried out has looked at the pitch and rhythmic contours separately, and pitch and rhythm together, with duration being represented in terms of the number of crotchets occupied by the note.

Two algorithms have been used. The first is the dynamic programming approach and the second is the Knuth-Morris-Pratt algorithm (Knuth et al., 1977), which is an exact matching algorithm. The dynamic programming algorithm used was developed by Ukkonen (Ukkonen, 1985a; Ukkonen, 1985b; Jokinen et al., 1996). This method calculates a column of the table at a time, and only takes $O(kn)$ time to run,where k is the number of differences allowed and n is the length of the text. This method is more efficient than the traditional DP method especially when the score is large. However, fragmentation and consolidation edit operations used by Mongeau & Sankoff and Smith et al. require the entire edit distance table making it less efficient for large scores. The first movement of Mozart's 40th Symphony has been used as the test composition.

The amount of musical information in the matching process plays a part in its success. Uitdenbogerd et al. only use the two types of pitch contour, whereas Smith et al, and Mongeau & Sankoff use rhythm and pitch contour. The success of using pitch contour and rhythmic contour both separately and combined is shown Figure 5.

Figure 5 shows that the number of matches found decreases as the pattern length



(a) Pattern Length against No. of Matches (b) Threshold against No. of Matches

Figure 5: Number of Matches with Different Contours

increases. There are other variables which effect the number of matches found. These are the number notes in the pattern, shown in Figure 5(b) and the number of edit operations allowed, shown in Figure 5(a). The results are from the first movement of Mozart's 40th Symphony, using three different patterns derived from Figure 1. The first pattern uses the first 6 notes, the second the first 10 notes, and the third all the notes.

One of the problems of using pitch contour is the fact that rests are not accommodated. They play a part in the melodic/pitch contour and have an effect on the rhythmic contour. Rests are not considered in the pitch contour work of Uitdenbogerd et al, which may suggest why the results for pitch contour alone are not as good as intervallic contour. Mongeau & Sankoff use a dummy symbol in their work. In the work outlined here, rests are accommodated by using the letter '**R**' in the pitch contour, and a value of 0 for the intervallic contour.

3.1 The Importance of Musical Features

Consider now the broadly similar problem of speech recognition using the traditional Hidden Markov Model(HMM). If the HMM is trained using a joint audio-visual feature vector, as opposed to the traditional audio-only feature vector, then the recognition rate increases(Chen, 2001).

Likewise, as melodies are treated as abstract sequences, some important musical data is ignored that could aid melodic analysis. This includes metric accent, phrasing and metric structure. Other features including dynamics and articulations should also be considered. Although Mongeau & Sankoff acknowledged the work of Dillon & Hunter (Dillon and Hunter, 1982) they deemed that it was too specific for their purposes. Dillon & Hunter's work focused on folk songs but the fact that stressed pitches were encoded enabled variations to be found in which the meter was altered.

The above features are not as important as the pitch and rhythm contours but they are important in the development of a melody. Dynamics and articulations are used in Lerdahl & Jackendoff's grouping preference rules (Lerdahl and Jackendoff, 1983). They use changes in dynamic and articulation to help specify local grouping boundaries. This suggests that dynamics and articulations should be considered as variables in a melody.

Phrasing and articulation help to give a melody character. They also help to focus attention on the important notes of the melody, usually with the aid of metrical accent. If we consider the motifs shown in Figure 6, although they are slightly different they have the same basic contour shape. In Figure 6(a), which is the first occurrence of the motif in the passage, the third note is reinforced by the peak of the crescendo. Figure 6(b) the third note is accented to keep the peak of the motif at that point. In the third occurrence, Figure 6(c), there are two differences, the first being the bow stroke used for the first note and the second that the first two notes do not have the same articulations. These alterations change the emphasis of the motif, and now the first note of the motif has more significance than it did before. The dynamic at which the motif begins is also subject to change. All the differences in articulation and dynamic alter the character of this motif and should be considered when analysing and tracking the development of the motif. Using the dynamic and articulations of a melody will help to differentiate between occurrences of similar melodies enabling them to be categorised more easily. It is especially useful when exact occurrences of a melody are required, as the articulations add an extra layer of information, similar to the audio-visual speech recognition problem.

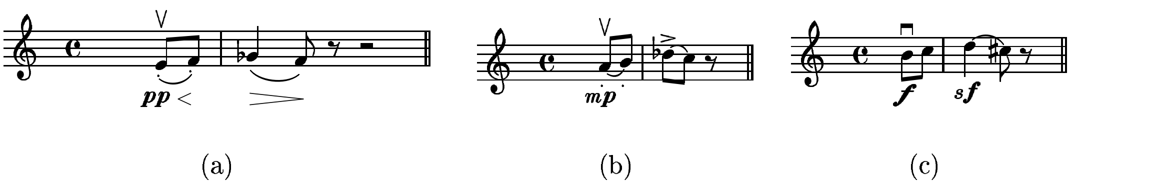


Figure 6: Three Occurrences a Motif from the First Movement of Tchaikovsky's *Pathètique* Symphony

We propose that the representations currently used should be extended to include the above features, and a hierarchy of musical features formed. This hierarchy will form the basis of the weighting calculation that will be used in the DP algorithm. The weightings will then be calculated using a similar procedure to Mongeau & Sankoff. The exact influence that the phrasing, articulations and dynamics have on a melody has still to be defined.

3.2 Melodic Lines

Another drawback is that only monophonic melody lines are considered. But the timbre of the instrument used to play the melody also affects how we 'hear' the melody. Timbre has not been considered as it is difficult to judge, with the current representation, which instrument is playing at any particular point. In the research undertaken each line is considered individually as monophonic sequences.

This leads to the problem of melodies distributed across multiple parts. Figure 7 shows the second theme of the first movement of Mozart's 40th Symphony, according to (Barlow and Morgenstern, 1948). However, in the score this theme appears across the first violin and the first oboe, shown in Figure 8. A more acute case is to be found in the opening of the fourth movement of Tchaikovsky's *Pathétique* Symphony, where notes of the theme alternate between the first and second violins. The accompaniment is written in the same way. This is addressed by Crawford et al. but no solution is proposed.

Uitdenbogerd et al. use melodic extraction techniques based on music perception research so only a single melody line is used. The main technique extracts the highest notes that appear in each part and construct the melodic sequence using those notes. However, this method may well have some errors in the abstracted melodic line that it generates, as it is unlikely to be what the composer wanted the listener to hear. The reason behind the selection of the highest notes is shown in Deustch(Deutsch, 1999).

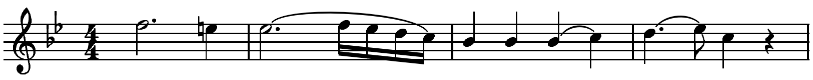


Figure 7: Second Theme of First Movement 40th Symphony by Mozart K.550

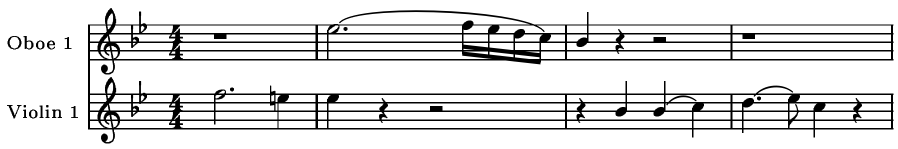


Figure 8: Second Theme of First Movement 40th Symphony by Mozart K.550, as it appears in the score.

A further problem with the results from the different techniques is that, although they obtain similar melodies they do not show precisely how they differ, in terms of the compositional techniques used. In melodic analysis this is extremely important. Crawford et al. call this *evolution detection*. A possible solution to this is to adapt the dynamic programming methods so that the alterations that a match has gone through are shown. This would only be in terms of edit operations and no compositional techniques would be listed. A set of techniques could be made and then the list of edits could be matched to the compositional technique. Ideally, this new iteration would use diatonic pitches and durations as opposed to pitch or intervallic contour. However, as the pitch and intervallic contours are a neutral representation, transposed matches would be found, so that using pitch names pitches would cause a large change in the motif. Intervallic contour would be used, substituting the pitch names once the evolution has been calculated.

The method of specifying the search pattern is an important issue. All of the approaches require a predefined search pattern for their work. In the case of Mongeau & Sankoff, variations on '*Ah, Vous Dirai-je Maman*', Mozart K.265 were used. Smith et al. and Uitdenbogerd et al. use a user-defined search pattern, as they are concerned with database queries. This is not very desirable in melodic analysis unless the analyst knows the important melodic material of the composition being analysed. In the ideal situation the melodical material that has significance would be obtained automatically and categorised into melodies, themes and motifs. In the current work the melodic patterns are defined by the user. Also, three widely used compositional techniques such as inversion, retrograde, and retrograde-inversion, are not catered for. In the current work these transformations are implemented as separate patterns.

4 Conclusions

We have shown that the most widely-used pattern matching techniques are efficient when the application is database searching and only melodic contour is required. However, for the purposes of tracking melodic development, additional musical information is required, as well as a method for representing the developments between different occurrences of a melody. The most effective matching algorithm for melody tracking is the DP algorithm, which can be adapted to accommodate the additional musical features.

In future work, the musical features outlined here will be used in the detection of similar melodies. With a finalised hierarchy of musical features, the cost of a change of one of the minor features, such as a dynamic, will be limited. A method of classifying melodies is also needed, this would enable the different forms of a melody to be grouped together. This would allow for easier analysis of the variation used in the development of the melody.

References

Barlow and Morgenstern (1948). *A Dictionary of Musical Themes.* Crown Publishers.

Chen, T. (2001). Audiovisual speech processing-lip reading and lip synchronisation. *IEEE Signal Processing Magazine*, 18(1):9–21.

Cope, D. (1990). Pattern matching as an engine for the computer simulation of musical style. In *Proceedings of the International Computer Music Conference*, pages 288–291.

Cope, D. (1991). *Computers and Musical Style.* Oxford University Press, 1st edition.

Crawford, T., Iliopoulos, C., and Raman, R. (1998). String-matching techniques for musical similarity and melodic recognition. *Computing in Musicology 11 : Melodic Similarity.*

Deutsch, D. (1999). *The Psychology of Music*, chapter 9 Grouping Mechanisms in Music. Academic Press, 2nd edition.

Dillon, M. and Hunter, M. (1982). Automated identification of melodic variants in folk music. *Computers and the Humanities*, 16:107–117.

Dowling, W. J. (1978). Scale and contour: Two components of a theory of memory for melodies. *Psychological Review*, 85(4):341–354.

Edworthy, J. (1985). Interval and contour processing. *Music Perception*, 2(3):375–388.

Jokinen, P., Tarhio, J., and Ukkonen, E. (1996). A comparison of approximate string matching algortihms. *Software - Practice and Experience*, 26(12):1439–1458.

Knuth, D. E., Morris Jr, J. H., and Pratt, V. R. (1977). Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350.

Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music.* MIT Press.

Mongeau, M. and Sankoff, D. (1990). Comparison of musical sequences. *Computers and the Humanities*, 24:161–175.

Selfridge-Field, E. (1998). Conceptual and representational issues in melodic comparison. *Computing in Musicology 11 : Melodic Similarity.*

Smith, L. A., McNab, R. J., and Witten, I. H. (1998). Sequence-based melodic comparison: A dynamic programming approach. *Computing in Musicology*, 11.

Uitdenbogerd, A. L. and Zobel, J. (1999). Melodic matching techniques for large music databases. In *Proceedings of the Association for Computer Machinery Multimedia*, pages 57–66.

Ukkonen, E. (1985a). Algorithms for approximate string matching. *Information and Control*, 64:100–118.

Ukkonen, E. (1985b). Finding approximate patterns in strings. *Journal of Algorithms*, 6:132–137.

Ukkonen, E. (1992). Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92:191–211.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the Association for Computer Machinery*, 21(1):168–173.