Peak Alignment using Restricted Edit Distances

Veli Mäkinen*
AG Genominformatik, Technische Fakultät
Universität Bielefeld, Germany.
veli@cebitec.uni-bielefeld.de

Abstract

A *peak* is a pair of real values (x, y), where x is the time when peak of height y is registered. In the *peak alignment problem*, we are given two sequences of peaks, and our task is to align the sequences allowing some basic edit operations on the peaks. We study an instance of the peak alignment problem that arises in the analysis of *Mass Spectrometry* data in Systems Biology. There the measurement technique guarantees that two peaks (x, y), (x', y') can only be considered the same if x is close enough to x', and y is close enough to y'. We review some methods to do alignment under such restrictions on matches.

1 Introduction

Mass spectrometry is a device that measures the masses and intensities of given sample particles. It can be used e.g. for measuring intensities of different proteins in a sample of cells. The result of a measurement can usually be interpreted as a *mass spectrum*: a function associating the intensity (y-coordinate value) to each possible mass (x-coordinate value). The purpose of peak alignment is to be able to compare two spectra taken on *different* samples. This immediately poses a problem, since the amount of substrate differs from sample to sample, and also the mass values can be non-linearly scaled. The measurement technique, however, allows to add so-called calibration peaks to the spectra. These can be used to normalize (calibrate) spectra so that the global variations can be taken into account as a preprocessing step. Hence, we can assume that between two spectra, there are only local variations that need to be traced. The local variations that are of interest are the peaks that have changed most or disappeared.

Let us now assume that after some preprocessing steps we have a reasonably accurate spectrum $A[1, m] = (x_1, y_1)(x_2, y_2) \cdots (x_m, y_m)$ of the measured particles, where the ith particle has mass x_i and intensity y_i. We can assume $x_{i-1} < x_i$. To simplify the notation, we use $A[i]$ to denote the ith particle, $A[i].x$ its mass and $A[i].y$ its intensity. We denote by $|A| = m$ the length of a sequence $A = A[1, m]$.

Let $A[1, m]$ and $B[1, n]$ be two mass spectra. To *align* A and B, one has to identify the correct set of basic *edit operations* to convert A into B. The specific properties of the problem must be taken into account when designing the right edit operations and their costs, in order to produce a reliable alignment. In this paper, we choose a "purists" approach; we minimize one quantity, and limit others. The purpose is that after the optimal alignment is computed, we can easily characterize what properties it possesses.

To apply our approach, we identify the following conditions and error sources that need to be taken into account in aligning A and B:

i) Small changes in peak heights: We can assume that $A[i]$ can be matched to $B[j]$ if $|A[i].y - B[j].y| \leq \delta$, for some $\delta > 0$ that is estimated knowing the maximum error the specific mass spectrometry technique produces to the values that should not have changed.

ii) Small changes in mass values: We can assume that $A[i]$ can be matched to $B[j]$ if $|A[i].x - B[j].x| \leq \alpha$, for some $\alpha > 0$ that is estimated as in case (i).

iii) Peak heights differ significantly: Sometimes $A[i]$ can correspond to $B[j]$ even if condition (i) does not hold. Tracing such changes is actually the main goal of peak alignment. Therefore, we should allow *substitutions* between values that differ more than by δ.

*Supported by the ``Deutsche Forschungsgemeinschaft'' (BO 1910/1-3) within the Computer Science Action Program and by the Academy of Finland.

iv) Extra and missing peaks: Sometimes there is no peak in B corresponding to $A[i]$, and vice versa. Such cases are either extreme cases of (iii), or errors caused by peak detection. Therefore, we should allow peak *insertions* and *deletions*.

The rest of the paper focuses on examining closer the different options to align A and B taking into account the observations listed above.

2 Related Work

Peak sequences can be seen as *trajectories*. Converting a trajectory into another is a classical problem known as *time warping* [13]. However, time warping methods are not suitable for data that contains extra/missing peaks, since deleting a peak incurs a cost that depends on its size. In our problem extra/missing peaks occur *on purpose*; they explain a biological phenomenon. The same classical book [13] contains, however, an approach very similar to our proposal; edit distances with restricted gaps. The basic recurrences for computing such restricted edit distances are already given there; we will review some of them in the sequel. As far as we know, only our recent previous work [12, 11] contains improved algorithms for these problems; we will also review some of these solutions for completeness since some of them only appear in a thesis work [11].

Peak sequences appear also in other applications than in mass spectrometry. Almost identical problem is the alignment of sequences from *one-dimensional gel electrophoresis*. There is a study on this problem that uses the time warping based approach [1]. Our approach is an alternative to this, and could as well be applied to the electrophoresis problem.

For mass spectra alignment, the earlier studies [2, 5, 3] are based on the probabilistic sequence analysis paradigm [7]. We remark that such modeling is rigid when considering pure mass sequences, as one can compare with the theoretical spectrum [3]. We are not aware of any approach that could model peak hights similarly.

3 Restricted Edit Distances

Recall the peak sequences $A[1,m]$ and $B[1,n]$ from the introduction. Let us define $A[i] \sim B[j]$ iff cases (i) and (ii) given in the introduction hold, i.e. $|A[i].y - B[j].y| \leq \delta$ and $|A[i].x - B[j].x| \leq \alpha$. We denote $A \sim B$ iff $|A| = |B|$ and $A[i] \sim B[i]$ for all $1 \leq i \leq |A|$.

Let X be a *subsequence* of A, i.e., any peak sequence that can be obtained by deleting zero or more peaks from A. Let Y be a subsequence of B. We say that a sequence Z is *approximately common* to X and Y iff $Z \sim X$ and $Z \sim Y$. Such Z is *approximately common subsequence* of A and B. We denote by $LACS(A, B, \sim)$ the Longest Approximately Common Subsequence of A and B.

Hence, $|LACS(A, B, \sim)|$ is a measure of similarity between A and B that takes into account cases (i), (ii) and (iv) listed in the introduction. Let us look at the dual of $|LACS(A, B, \sim)|$ that gives us the possibility to add case (iii) to the measure:

$$\widetilde{D}(A, B) = m + n - 2 * |LACS(A, B, \sim)|. \qquad (1)$$

It is easy to see that $\widetilde{D}(A, B)$ equals the minimum number of peak insertions and deletions needed to convert A into a sequence Z such that $Z \sim B$. To allow case (iii), we can simply add an additional operation to substitute any $A[i]$ with $B[j]$. Due the lack of space, we do not consider substitutions in the sequel (and also because consecutive insertion and deletion are enough to produce a substitution).

An equally justified way of taking into account case (i) is to minimize the sum of the differences $|A[i].y - B[j].y|$ between matched elements i, j in the alignment. Such approach is better if no good upperbound for δ is known. For this purpose, we fix a parameter κ, and try to find the best alignment between A and B allowing κ *free* edit operations. Best alignment now means the one minimizing the sum of the differences between matched elements. We denote by $D^{\kappa}(A, B)$ the distance, where we allow κ free insertions and deletions.

Distance $D^{\kappa}(A, B)$ can be seen as a means to find κ *most interesting peak changes*: Assume that the real data contains $k < \kappa$ significant changes in the peaks. If our estimate for κ is not too large, we are likely to find the correct k changes and $\kappa - k$ "false positives". That is because our alignment is based only on the (hopefully majority of) peaks that have changed a little, and the k *outliers* do not influence the alignment.

Notice that condition (ii) is not taken into account in the measure $D^{\kappa}(A, B)$. Therefore we also consider measure $\widetilde{D}^{\kappa}(A, B)$, where in addition to the κ free operations, we allow matches only with peaks $A[i] \sim B[j]$. This measure takes condition (i) into account twice (restricting absolute differences and minimizing their sum), that can be too restrictive. Our algorithms do not depend on how $\sim$-relation is defined, and for this distance one could as well redefine $\sim$ so that it only considers condition (ii).

4 Basic Recurrences

Standard dynamic programming recurrences follow easily for the distance measures defined in the previous section. Let us not repeat the most well-known formula here, but just mention that distance $\widetilde{D}(A, B)$ is easily computable in $O(mn)$ time [6]. Let us, instead, give recurrences that take into account the *sparse* nature of the problem. We define the *match set* $M = M(A, B, \sim)$ as $\{(i, j) \mid A[i] \sim B[j]\}$. Then we have the following recurrence for points $(i, j) \in M$:

$$\begin{aligned} d_{i,j} \quad &= \quad \min\{d_{i',j'} + i + j - i' - j' - 2 \\ &\qquad \mid (i', j') \in M, i' < i, j' < j\}, \end{aligned} \tag{2}$$

where we assume to have a boundary point $(0, 0) \in M$ with $d_{0,0} = 0$. There holds $\widetilde{D}(A, B) = d_{m+1,n+1}$.

Notice that trivial implementation of the above recurrence yields an $O(|M|^2)$ time algorithm, i.e. $O((mn)^2)$ in the worst case; much worse than by using the standard $O(mn)$ time algorithm. However, we will see in the next section how to implement this recurrence much faster.

Distance $D^\kappa(A, B)$ is not defined by the match set M, as it allows any peaks to match. We have the following basic recurrence for $D^\kappa(A, B)$:

$$\begin{aligned} d_{i,j,k} \quad &= \quad \min\{d_{i-1,j-1,k} + |A[i].y - B[j].y|, \\ &\qquad d_{i-1,j,k-1}, d_{i,j,k-1}\}, \end{aligned} \tag{3}$$

where the boundary conditions are $d_{0,k,k} = 0$ and $d_{k,0,k} = 0$ for $0 \le k \le \kappa$ (uninitialized cells are assumed to have value ∞). There holds $D^\kappa(A, B) = \min_{0 \le k \le \kappa} d_{m,n,k}$.

Again, trivial implementation of the above recurrence yields an $O(mn\kappa)$ time algorithm. One easily notices that only the $\kappa + 1$ middle diagonals need to computed, since elsewhere the κ insertions and deletions are not enough. This improves the running time to $O(\kappa^2 \min(m, n))$.

Distance $\widetilde{D}^\kappa(A, B)$ can be computed by slightly changing the above recurrence. The running time will remain the same. In the next section, we develop an algorithm whose running time depends on the size of the match set M. This algorithm is based on an alternative recurrence described next. For $\widetilde{D}^\kappa(A, B)$ we have the following recurrence for points $(i, j) \in M$:

$$\begin{aligned} d_{i,j,k} \quad &= \quad \min\{d_{i',j',k'} + |A[i].y - B[j].y| \\ &\qquad \mid (i', j') \in M, i' < i, j' < j, \\ &\qquad 0 \le k' = k - (i - i' - 1) - (j - j' - 1)\}, \end{aligned} \tag{4}$$

where we assume to have boundary point $(0, 0) \in M$ with $d_{0,0,0} = 0$. There holds $\widetilde{D}^\kappa(A, B) = \min_{0 \le k \le \kappa} d_{m+1,n+1,k}$, where we assume $A[m + 1] = B[n + 1]$.

5 Efficient Solutions using Range Minimum Queries

We will now review *sparse dynamic programming* solutions for the restricted edit distances. For $\widetilde{D}(A, B)$ such solutions have already existed some time. The essential technique is by Hunt and Szymanski [9]; they showed how to compute the length of the longest common subsequence in $O(|M| \log n)$ time. Almost identical algorithm can be applied in our case, and such was described by Eppstein et al.[8, Sect. 2] for the computation of RNA secondary structure. That algorithm uses $O(|M| \log \log(\min(mn/|M|, |M|))$ time, but is fairly complicated. Recently, a much simpler algorithm was developed with essentially the same running time [12]. Let us briefly describe it as we use the same technique later on to compute distance $\widetilde{D}^\kappa(A, B)$.

Let us assume that M is given in *reverse column order*, that is (i', j') precedes (i, j) iff $j' < j$ or ($j' = j$ and $i' > i$). Writing Eq. (2) in a form $d_{i,j} = i + j - 2 + \min\{d_{i',j'} - i' - j' \mid (i', j') \in M, i' < i, j' < j\}$ one notices that $d_{i,j}$ can be computed by making a *range minimum query* on some data structure $\mathcal{T}$ that stores points $(i', j') \in M$ associated with values $d_{i',j'} - i' - j'$, and adding $i + j - 2$ to this minimum *afterwards*. If values $d_{i,j}$ are computed in the reverse column order, and each value $d_{i,j} - i - j$ is added to $\mathcal{T}$ after its computation, one notices that condition $j' < j$ can be ignored; for values $d_{i',j'} - i' - j'$ that are stored in $\mathcal{T}$ when computing $d_{i,j}$, holds that if $i' < i$ then $j' < j$. That is, the condition $i' < i$ is enough. We have the following algorithm [12]:

```
Algorithm for $\widetilde{D}(A, B)$.
(1) $\mathcal{T}.Add(0, 0)$;
(2) for each $(i, j) \in M$ in reverse column order do
(3)         $d_{i,j} = i + j - 2 + \mathcal{T}.Minimum([-\infty, i))$;
(4)         $\mathcal{T}.Add(i, d_{i,j} - i - j)$;
(5) return $m + n + \mathcal{T}.Minimum([-\infty, m + 1))$;
```

Operation $\mathcal{T}.Add(i, v)$ adds key i associated with value v to the structure. Operation

$\mathcal{T}.Minimum([l, r))$ returns the minimum value associated to a key from range $[l, r)$. It is easy to support these operations if $\mathcal{T}$ is implemented as a binary search tree. As the keys are integers from range $[0, m]$, and the query range is semi-infinite, one can use more advanced structures [14] that support both operations in $O(\log\log m)$ time [12]. Hence, the overall running time is $O(|M| \log\log m)$, where $m \leq n$.

Let us now consider distance $\widetilde{D}^{\kappa}(A, B)$. Recall Eq. (5). When computing the value of a cell $d_{i,j,k}$ we need the minimum of values $|A[i].y - B[j].y| + d_{i',j',k'}$, where $(i', j'), (i, j) \in M$, $i' < i$, $j' < j$, and $k - (i - i' - 1) - (j - j' - 1) = k'$. We observe that the query area is actually on a two-dimensional layer of the three-dimensional point set. This is because we can write the condition $k - (i - i' - 1) - (j - j' - 1) = k'$ as $i' + j' - k' = i + j - k - 2$, and consider values $d_{i',j',k'}$ in a three-dimensional space at coordinate $(i', j', i' + j' - k')$. When computing value $d_{i,j,k}$ we can query the minimum, say d, from a two-dimensional range $[-\infty, i) \times [-\infty, j)$ at layer $i + j - k - 2$. Finally, $d_{i,j,k} = |A[i].y - B[j].y| + d$. This two-dimensional range query at layer $i + j - k - 2$ can be implemented as follows. First, we traverse M in reverse column order, so that the range $[-\infty, j)$ is handled automatically (similarly as explained before). Second, we use a separate data structure for each layer and store pointers to the correct layer in an array $L(-k, \ldots m + n)$. In a layer $i + j - k - 2$ we query the one-dimensional range $[-\infty, i)$ from data structure $L(i + j - k - 2)$. Since this range is semi-infinite and the values are in the range $[0, m]$, the query can be answered in $O(\log\log m)$ time. Altogether, the algorithm works in $O(\kappa |M| \log\log m)$ time.

Finally, we note that the construction time for $M = M(A, B, \sim)$ has not been included in the above time bounds. However, it is easy to construct M in time $O(n \log n + |M| \log\log m)$ [12, 11].

6 Implementation

We implemented a prototype software that visualizes two given spectra and aligns them using a selected algorithm. We experimented with real data and with simulated data: We generated random errors to real peak sequences, and tried to find the correct alignments between the originals and the noisy versions. The methods worked robustly, making only few mistakes each time, when using realistic error distributions and good (but not correct) estimates for the parameter values δ and κ.

References

[1] T. Aittokallio, P. Ojala, T. J. Nevalainen, O. Nevalainen. Automated detection of differentially expressed fragments in mRNA differential display. *Electrophoresis*, 22(10):1935–1945, 2001.

[2] V. Bafna and N. Edwards. SCOPE: A probabilistic model for scoring tandem mass spectra against a peptide database. *Bioinformatics*, 17:S13-S21, 2001.

[3] S. Böcker and H.-M. Kaltenbach. Mass Spectra Alignments and Their Significance. To appear in *Proc. CPM 2005*, Korea, June 19-22, 2005.

[4] H. N. Gabow, J. L. Bentley, and R. E. Tarjan. Scaling and related techniques for geometry problems. *Proc. STOC'84*, pp. 135–143, 1984.

[5] J. Colinge, A. Maselot, and J. Magnin. A systematic statistical analysis of ion trap tandem mass spectra in view of peptide scoring. In *Proc. WABI 2003*, Springer-Verlag LNCS 2812, pp. 25–38, 2003.

[6] M. Crochemore and W. Rytter. *Jewels of Stringology.* World Scientific, 2002.

[7] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Camb. Univ. Pr., 1997.

[8] D. Eppstein, Z. Galil, R. Giancarlo, and G. F. Italiano. Sparse dynamic programming I: linear cost functions. *Journal of the ACM*, 39(3):519–545, 1992.

[9] J. W. Hunt and T. G. Szymanski. A fast algorithm for computing longest common subsequences. *Commun. ACM*, 20(5):350–353, May 1977.

[10] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 6:707–710, 1966.

[11] V. Mäkinen. *Parameterized Approximate String Matching and Local-Similarity-Based Point-Pattern Matching.* PhD thesis, Report A-2003-6, Department of Computer Science, Univ. of Helsinki, 2003.

[12] V. Mäkinen, G. Navarro, and E. Ukkonen. Algorithms for Transposition Invariant String Matching. In *Proc. STACS 2003*, Springer-Verlag LNCS 2607, pages 191–202, 2003.

[13] D. Sankoff and J. B. Kruskal, editors. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley Publishing Company, 1983.

[14] P. van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Information Processing Letters*, 6(3):80–82, 1977.