Improved De Novo Peptide Sequencing using LC Retention Time Information

Yves Frank[†1], Tomas Hruz[2], Thomas Tschager[*3], and Valentin Venzin[†4]

1 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
2 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
3 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
4 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**

Abstract

Liquid chromatography combined with tandem mass spectrometry (LC-MS/MS) is an important tool in proteomics for identifying the peptides in a sample. Liquid chromatography temporally separates the peptides and tandem mass spectrometry analyzes the peptides, that elute one after another, by measuring their mass-to-charge ratios and the mass-to-charge ratios of their prefix and suffix fragments. De novo peptide sequencing is the problem of reconstructing the amino acid sequences of the analyzed peptide from this measurement data. While previous approaches solely consider the mass spectrum of the fragments for reconstructing a sequence, we propose to also exploit the information obtained from liquid chromatography. We study the problem of computing a sequence that is not only in accordance with the experimental mass spectrum, but also with the retention time of the separation by liquid chromatography. We consider three models for predicting the retention time of a peptide and develop algorithms for de novo sequencing for each model. An evaluation on experimental data from synthesized peptides for two of these models shows an improved performance compared to not using the chromatographic information.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity, J.3 Life and Medical Sciences: Biology and Genetics

Keywords and phrases Computational proteomics, Peptide identification, Mass spectrometry, De novo peptide sequencing, Retention time prediction

Digital Object Identifier 10.4230/LIPIcs.WABI.2017.26

1 Introduction

The amino acid sequences of peptides in a sample can be analyzed with the following tandem mass spectrometry experiment [7]. First, the peptides are separated temporally by liquid chromatography. Then, the mass spectrometer measures the mass-to-charge ratio of a peptide and fragments multiple copies of it at random positions. Finally, the mass spectrometer measures the mass-to-charge ratio of the resulting fragments. The peptide sequencing problem is to reconstruct the amino acid sequence of the peptide from the experimental data. This problem has been extensively studied [5, 18]. Nevertheless, when analyzing unknown peptides the otherwise very successful database search approach is not applicable and de novo sequencing, which is the reconstruction of the whole sequence from scratch, is necessary.

* Corresponding author: thomas.tschager@inf.ethz.ch.
† Preliminary work for this study was carried out during the bachelor theses of Y. Frank and V. Venzin.

Several algorithms for de novo sequencing [2, 1, 6, 10] consider the differences of the peptide's fragment masses to reconstruct the peptide's sequence. Various scoring functions have been proposed that try to exploit as much information as possible from the mass spectrum of the fragments to find a sequence that explains the observed spectrum as well as possible. However, the information obtained from the chromatographic separation in the first step of the experiment is not considered by these scoring functions.

In liquid chromatography, the peptides in a sample have to pass through a column. The time a peptide needs to traverse the column is called *retention time* and depends on the chemical properties of the peptide. This process results in the temporal separation of the peptides in a sample. Predicting the retention time of a peptide from its amino acid sequence is a challenging task [14, 11]. Several studies use retention time prediction models for peptide sequencing as a filtering step after a database search to increase the confidence of identification and to identify false positive identifications [12, 16].

However, to the best of our knowledge, the retention time information has not been considered by de novo peptide sequencing algorithms. This information can be useful, because it allows to reconstruct parts of a sequence that cannot be resolved by mass spectrometry (e.g. amino acids and fragments with equal masses). Moreover, it is available without additional experimental effort. However, simply filtering the solutions of a standard de novo sequencing algorithm by predicted retention time is not an option, as it requires to compute all possible solutions in the worst case to find an optimal solution. We formulate and study a de novo sequencing problem that integrates the retention time as an additional constraint and does not require filtering many candidates. We are interested in a sequence that both matches the experimental spectrum and the measured retention time. We consider three additive retention time prediction models and develop algorithms for each model.

In this study, we do not aim for a replacement for available de novo sequencing tools, but rather explore ways of exploiting the retention time information in de novo sequencing algorithms. We evaluate the performance of two algorithms on experimental measurements from synthesized peptides. In our evaluation, we consider a basic scoring function to clearly expose the impact of using retention time prediction models. We compare our algorithms to DeNovoΔ [4, 17], an algorithm that considers the same symmetric difference scoring model but no retention time information. This scoring model shows improved identification rates compared to the prevalent shared peak count scoring model [1]. For the third prediction model, we present some preliminary results.

Considering the retention time information comes at the cost of higher computational effort and requires additional parameters for retention time prediction (either estimated from suitable datasets or taken from the literature). Yet, we believe that it is useful to exploit retention time information for peptide identification and to further study the integration of retention time information in algorithms for de novo peptide sequencing.

2 Notation and Problem Definition

In this paper, we model amino acids by characters and peptides by strings. We consider an alphabet Σ of characters. A string $\mathtt{S} = \mathtt{a}_1 \dots \mathtt{a}_n$ is a sequence of characters. The empty string is denoted by $\mathtt{S}_\emptyset$. Every character $\mathtt{a} \in \Sigma$ has a mass $m(\mathtt{a}) \in \mathbb{R}^+$. The mass of a string $\mathtt{S} = \mathtt{a}_1 \dots \mathtt{a}_n$ is the sum of its character's masses $m(\mathtt{S}) = \sum_{i=1}^{n} m(\mathtt{a}_i)$. The empty string $\mathtt{S}_\emptyset$ has mass 0. A substring of $\mathtt{S}$ is denoted by $\mathtt{S}_{i,j} = \mathtt{a}_i \dots \mathtt{a}_j$ for $1 \leq i \leq j \leq n$. The prefix set $\mathrm{Pre}(\mathtt{S})$ contains all prefixes of $\mathtt{S}$ including the empty string, i.e. $\mathrm{Pre}(\mathtt{S}) = \mathbb{U}_{i=1}^{n} S_{1,i} \cup \{S_\emptyset\}$. The *theoretical spectrum* of $\mathtt{S}$ is the union of all its prefix and suffix masses $\mathrm{TS}(\mathtt{S}) =$

(a) Linear: $t_{\text{lin}}(\mathtt{S}) = t(\mathtt{A}) + t(\mathtt{I}) + t(\mathtt{A}) + t(\mathtt{G}) + t(\mathtt{A}) + t(\mathtt{K})$

(b) Position-dependent: $t_{\text{pos}}(\mathtt{S}) = t_{\text{pre}}(\mathtt{A}, 1) + t_{\text{pre}}(\mathtt{I}, 2) + t(\mathtt{A}) + t(\mathtt{G}) + t_{\text{suf}}(\mathtt{A}, 2) + t_{\text{suf}}(\mathtt{K}, 1)$

(c) Neighborhood-based: $t_{\text{nei}}(\mathtt{S}) = t(-, \mathtt{A}) + t(\mathtt{A}, \mathtt{I}) + t(\mathtt{I}, \mathtt{A}) + t(\mathtt{A}, \mathtt{G}) + t(\mathtt{G}, \mathtt{A}) + t(\mathtt{A}, \mathtt{K}) + t(\mathtt{K}, -)$

Figure 1 Retention time prediction for string $\mathtt{S} = \mathtt{AIAGAK}$. (a) In the linear model, the retention time of a string is the sum of its character's coefficients. (b) In the position-dependent model (with $\gamma = 2$), the position of the first and the last two characters is considered additionally. (c) The neighborhood-based model considers all pairs of consecutive characters in a string. The first and the last character have additional coefficients, as they only have one adjacent character.

$\{m(\mathtt{T}), m(\mathtt{S}) - m(\mathtt{T}) \mid \mathtt{T} \in \text{Pre}(\mathtt{S})\}$. Note that for every prefix $\mathtt{T} \in \text{Pre}(\mathtt{S})$ the string $\mathtt{S}$ has a complementary suffix of mass $m(\mathtt{S}) - m(\mathtt{T})$. We say a mass m is *explained* by $\mathtt{S}$ if $m \in \text{TS}(\mathtt{S})$.

We define three simple models for predicting the retention time of a string $\mathtt{S} = \mathtt{a}_1 \dots \mathtt{a}_n$ (Figure 1). The first model is a simple additive model with one coefficient for each character in Σ assuming the retention time of a string mainly depends on the composition of its characters [9]. The second model additionally considers the position of the characters at the beginning and the end of the string [9, 8]. The last model uses coefficients for pairs of consecutive characters to model the influence of a character's direct neighborhood [8, 15].

Linear model. Every character $\mathtt{a} \in \Sigma$ has a retention time coefficient $t(\mathtt{a}) \in \mathbb{Z}$. The retention time of a string $\mathtt{S}$ is the sum of the retention time coefficient of its characters,

$$t_{\text{lin}}(\mathtt{S}) = \sum_{i=1}^{n} t(\mathtt{a}_i). \tag{1}$$

Position-dependent model. We define distinct retention time coefficients for the first γ and the last γ positions of a string, where $1 \leq \gamma \leq \lfloor \frac{n}{2} \rfloor$. The retention time coefficient of the i-th character for $i \leq \gamma$ is denoted by $t_{\text{pre}}(\mathtt{a}_i, i) \in \mathbb{Z}$ and the retention time coefficient of the $(n - j + 1)$-th character for $j \leq \gamma$ by $t_{\text{suf}}(\mathtt{a}_{n-j+1}, j) \in \mathbb{Z}$. The retention time of a string $\mathtt{S}$ is the sum of the corresponding retention time coefficients

$$t_{\text{pos}}(\mathtt{S}) = \sum_{i=1}^{\gamma} t_{\text{pre}}(\mathtt{a}_i, i) + \sum_{j=\gamma+1}^{n-\gamma} t(\mathtt{a}_j) + \sum_{k=1}^{\gamma} t_{\text{suf}}(\mathtt{a}_{n-k+1}, k). \tag{2}$$

Neighborhood-based model. We define retention time coefficients $t(\mathtt{a}, \mathtt{b}) \in \mathbb{Z}$ for pairs of consecutive characters $\mathtt{a}, \mathtt{b} \in \Sigma$. The first and the last character $\mathtt{a}_1$ and $\mathtt{a}_n$ of a string $\mathtt{S}$ have additional coefficients $t(-, \mathtt{a}_1), t(\mathtt{a}_n, -) \in \mathbb{Z}$, as these characters have only one adjacent character in $\mathtt{S}$. The retention time of $\mathtt{S}$ is the sum of all these coefficients,

$$t_{\text{nei}}(\mathtt{S}) = t(-, \mathtt{a}_1) + \left(\sum_{i=1}^{n-1} t(\mathtt{a}_i, \mathtt{a}_{i+1}) \right) + t(\mathtt{a}_n, -). \tag{3}$$

The coefficients for all three models need to be estimated based on a training dataset (Sections 4.1 and Appendix Section B) or taken from the available literature.

We recall the de novo peptide sequencing problem with respect to the symmetric difference scoring model [17]: Given a mass M and a set of fragment masses X (measured by the mass spectrometer), find a string $\mathtt{S}$ of mass M that minimizes $|\text{TS}(\mathtt{S}) \triangle X|$. Equivalently to computing a string with mass M that minimizes $|\text{TS}(\mathtt{S}) \triangle X|$, we can compute a string that maximizes $|\text{TS}(\mathtt{S}) \cap X| - |\text{TS}(\mathtt{S}) \setminus X|$, as X is a fixed input and $\mathtt{S}$ can be chosen. Throughout this paper, we assume that $0, M \in X$.

We consider a variant of this problem that also considers the measured retention time and a retention time prediction function $t_* : \Sigma^* \to \mathbb{Z}$. A function $t_*()$ can return negative values, as a substring can have a negative effect on the retention time of a string.

▶ **Problem 1** (De Novo Sequencing Problem). *Let Σ be an alphabet of characters, with a mass $m(\mathtt{a}) \in \mathbb{R}^+$ for each $\mathtt{a} \in \Sigma$. Given a peptide mass $M \in \mathbb{R}^+$, a retention time $T \in \mathbb{N}$, a tolerance parameter $\varepsilon \geq 0$ and a set $X = \{x_i \in \mathbb{R}^+ \mid i = 1, \ldots, k\}$, find a string $\mathtt{S}$ of characters in Σ with $m(\mathtt{S}) = M$ and $|t(\mathtt{S}) - T| \leq \varepsilon$ that minimizes $|\mathrm{TS}(\mathtt{S}) \triangle X|$ among all strings with mass M and a retention time $t_*(\mathtt{S}) \in [T - \varepsilon, T + \varepsilon]$.*

2.1 Model Simplifications

The model used in this paper simplifies several aspects of experimental data. First, the peptide molecule contains an H_2O molecule in addition to the amino acid molecules. Therefore, the peptide mass has an offset of 18 Dalton compared to the sum of the amino acid masses. To simplify the description of the algorithms, we do not consider this offset (i.e. the mass M is the sum of only the amino acid masses) and the mass offsets of different ion types. However, we do consider both offsets in the implementation of our algorithms using techniques described in [17]. Moreover, the mass spectrometer measures masses-to-charge ratios. Charge state deconvolution [7] is required as a preparatory step to convert mass-to-charge ratios to masses if multiply charged fragments should be considered. Our model can consider fixed modifications by altering the amino acid masses and variable modifications by adding new characters to the alphabet. Finally, we consider integer values in the description of the algorithm and ignore the mass accuracy of the mass spectrometer. We discuss in the appendix how we account for the mass accuracy and also refer to [17].

3 Algorithms for De Novo Sequencing with Retention Time

We briefly describe the algorithm DeNovoΔ [17] for computing a string of mass M that minimizes $|\mathrm{TS}(\mathtt{S}) \triangle X|$ without considering retention times. We refer to [17] for a detailed description and a proof of correctness. Then, we describe algorithms based on DeNovoΔ for solving the de novo sequencing problem for each considered prediction model.

The search space of DeNovoΔ is modeled by a directed acyclic multigraph $G = (V, E)$ based on the given set X. A vertex in G represents a mass and a path in G represents a string. For every mass $m \in X$ there are two vertices m and $M-m$ in G, i.e. $V = \{m, M-m \mid m \in X\}$. An edge in G is always directed from the smaller to the larger mass. Two vertices v and w are connected by an edge if there exists a string with mass $w - v$. For each such string with mass $w - v$, we add an edge from v to w to the multigraph and label it with this string. That is, if v and w are connected by an edge with label $l(v, w)$, there is also an edge from v to w for every permutation of $l(v, w)$. In practice, we only consider edges with a maximal label length p. We denote the concatenation of the edge labels along a path P by $l(P)$.

Given a path P that starts at vertex 0, every traversed vertex represents the mass of a prefix of the string $l(P)$. If P additionally ends in vertex M, the path label both explains v and $M - v$ for every traversed vertex v. We find a string $\mathtt{S}$ of mass M that minimizes $|\mathrm{TS}(\mathtt{S}) \triangle X|$ by iteratively extending two paths both starting at vertex 0. One path represents a prefix and the other path a reversed suffix. We extend both paths until the sum of their labels' masses is equal to M and then concatenate the prefix and the reversed suffix to a string of mass M.

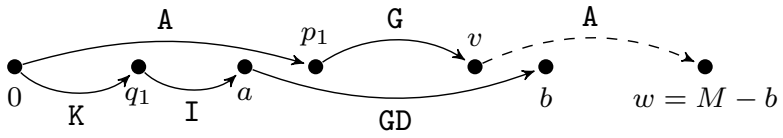

Figure 2 Multigraph G with two paths $P = (0, p_1, v)$ and $Q = (0, q_1, a, b)$. P and Q form a path pair, as there exists a sequence of balanced extensions leading to P and Q. A balanced extension of (P, Q) by (v, w) results in a path pair (P', Q), with $P' = (0, p_1, v, w)$ and $m(l(P')) + m(l(Q)) = M$. The path labels represent a prefix and a reversed suffix and can be combined to a string AGADGIK.

▶ **Definition 2** (Balanced extension)**.** Given two paths P and Q, a *balanced extension* extends the path that represents the string of smaller mass by a single edge, unless the resulting paths represent strings with a total mass larger than M. An arbitrary path is extended if both paths represent strings with equal masses.

▶ **Definition 3** (Path pair)**.** A *path pair* is a pair of paths $P = (0, \ldots, v)$ and $Q = (0, \ldots, a, b)$ in G that results from a sequence of balanced extensions starting from two paths $P_0 = (0)$ and $Q_0 = (0)$.

Figure 2 depicts an example for a path pair and a balanced extension. The set of masses that are explained by a path pair (P, Q) is the *partial theoretical spectrum*

$$\mathrm{PTS}(P, Q, M) = \{\ m(\mathtt{T}), M - m(\mathtt{T}) \mid \mathtt{T} \in \mathrm{Pre}(l(P)) \cup \mathrm{Pre}(l(Q))\ \}. \tag{4}$$

The *score* of the path pair is the number of masses explained by (P, Q) that are in X minus the number of explained masses that are not in X, i.e. $|\mathrm{PTS}(P, Q, M) \cap X| - |\mathrm{PTS}(P, Q, M) \setminus X|$. The set of masses explained by an edge (v, w) is denoted by

$$\mathrm{TSe}((v, w), M) = \{\ m(\mathtt{T}) + v,\ M - (m(\mathtt{T}) + v) \mid \mathtt{T} \in \mathrm{Pre}(l(v, w)),\ m(\mathtt{T}) \neq 0\ \}. \tag{5}$$

▶ **Lemma 4.** *For every path pair $P = (0, \ldots, v)$ and $Q = (0, \ldots, a, b)$ with $v \leq b$ and $v + b \leq M$ it holds that $a \leq v \leq b$. The balanced extension of (P, Q) by an edge (v, w) additionally explains all masses in $N((v, w), (a, b)) = \mathrm{TSe}((v, w), M) \setminus \mathrm{TSe}((a, b), M)$.*

Proof. Assume that there exists a path pair (P, Q) with $v \leq a$. This path pair results by definition from a sequence of balanced extensions. Consider the balanced extension in this sequence, where the last edge (a, b) of Q is added. In this step, either P ended in v or in some vertex $v' < v$. In both cases, a is the larger mass and Q represents the heavier string. Hence, the extension by (a, b) is not a balanced extension and (P, Q) is not a path pair.

Consider a balanced extension of (P, Q) by an edge (v, w). The edge (v, w) explains all masses in $\mathrm{TSe}((v, w), M)$. However, some of these masses might also be explained by (P, Q). We show that $\mathrm{TSe}((v, w), M) \setminus \mathrm{PTS}(P, Q, M) = N((v, w), (a, b))$, i.e. that all masses explained by (v, w) that are also explained by (P, Q), are explained by the last edge (a, b) of Q. We note that all masses in $\mathrm{TSe}((v, w), M)$ are larger than v and smaller than $M - v$. Moreover, all masses in $\mathrm{PTS}(P, Q, M)$ that are larger than v and smaller than $M - v$ are explained by the edge (a, b). Therefore, it follows that the balanced extension with (v, w) additionally explains all masses in $N((v, w), (a, b))$. ◀

Using Lemma 4, the algorithm DeNovoΔ [17] (Algorithm 1) computes a dynamic programming table DP. An entry $DP[v, (a, b)]$ contains the optimal score of a path pair ending at the vertex v, respectively at the edge (a, b). As a base case, we add a loop edge $(0, 0)$ to the

Algorithm 1 DeNovoΔ [17]

```
DP[v,(a,b)] = −∞ for all (a,b) ∈ E and all v ∈ V
DP[0,(0,0)] = 2
for (v ∈ V in ascending order):
    for ((a,b) ∈ E in lexicograph. asc. order with DP[v,(a,b)] ≠ −∞ ):
        for ((v,w) ∈ E with w + b ≤ M):
            if (w ≤ b):
                DP[w,(a,b)] = max(
                    DP[w,(a,b)], DP[v,(a,b)] + gain((v,w),(a,b))
                )
            else:
                DP[b,(v,w)] = max(
                    DP[b,(v,w)], DP[v,(a,b)] + gain((v,w),(a,b))
                )
```

graph and initialize $DP[0,(0,0)] = 2$. Given the optimal score $DP[v,(a,b)]$, the algorithm considers all possible balanced extensions of the corresponding path pair with outgoing edges of v. By Lemma 4, the additionally explained masses of such a balanced extension can be computed only given the last vertex v and the last edge (a,b) of the two paths. The score of the resulting new path pair can be computed by adding

$$\text{gain}((v,w),(a,b)) = |N((v,w),(a,b)) \cap X| - |N((v,w),(a,b)) \setminus X| \tag{6}$$

to the score $DP[v,(a,b)]$. The corresponding entry in the table is updated if the new score exceeds the value stored in this entry at this step of the algorithm. The optimal score for a string of mass M is equal to the maximum value of an entry $DP[M-b,(a,b)]$ among all edges (a,b) in G. The corresponding paths can be reconstructed starting from this entry. The combination of the resulting prefix and reversed suffix then leads to the desired string of mass M. The time complexity of DeNovoΔ is in $\mathcal{O}(|V| \cdot |E| \cdot d \cdot p)$, where d is the maximal out-degree of a vertex in G and p is the maximal length of an edge label [17].

3.1 Linear Prediction Model

In this section, we extend DeNovoΔ for the de novo sequencing problem with the linear retention time prediction model. First, we note that the retention time of a path pair $P = (0,\ldots,v)$ and $Q = (0,\ldots,a,b)$ with $a \leq v \leq b$ is the sum of the retention times of both substrings $t = t_{\text{lin}}(l(P)) + t_{\text{lin}}(l(Q))$. The retention time t' of a path pair obtained from (P,Q) by applying a balanced extension by some edge (v,w) can be computed as $t' = t + t_{\text{lin}}(l(v,w))$. That is, we only need t and the edge label $l(v,w)$ for computing t'.

However, it is not sufficient to only store the optimal score $DP[v,(a,b)]$ of any path pair ending in v, respectively (a,b), and its retention time to reconstruct a solution for our problem. There can be multiple path pairs ending in the same vertex and the same edge with different retention times. If we consider an optimal solution and its sequence of path pairs computed by the algorithm, a path pair $P = (0,\ldots,v)$ and $Q = (0,\ldots,a,b)$ in this sequence does not necessarily have an optimal score among all path pairs ending in v and (a,b). Nevertheless, its score is optimal among all path pairs with the same retention time that end in v and (a,b). Therefore, we need to store for each possible retention time t the optimal score of a path pair ending in vertex v and edge (a,b).

DeNovoΔLin (Algorithm 2) stores for each entry $DP[v,(a,b)]$ an array containing a score for every possible retention time t. $DP[v,(a,b)][t]$ is the optimal score for a path pair ending in v, respectively (a,b), with retention time t. For a given vertex v and an edge (a,b), the algorithm performs balanced extensions by all outgoing edges (v,w) of v. For every balanced extension and every feasible retention time t, the algorithm then computes the new retention time t' and the new score of the resulting path pair and updates the corresponding entry in the table. We can see by an inductive argument that the optimal scores in the table are computed correctly. As the base case, we note that $DP[0,(0,0)][0] = 2$ is correct, as an empty path pair explains the masses $\{0, M\} \subseteq X$ and has retention time 0. As soon as the entry $DP[v,(a,b)]$ is reached in line 7, all optimal scores for path pairs ending in vertex v and edge (a,b) have been computed. This holds by induction, as every possible balanced extension leading to a path pair ending in v and (a,b) has already been considered (given the optimal score of a preceding path pair). Moreover, the array in $DP[v,(a,b)]$ is not further modified as soon as the algorithm reaches the vertex v and the edge (a,b) in line 7. Therefore, the invariant holds that, if the algorithm considers a vertex v and an edge (a,b) in line 7, the corresponding entry $DP[v,(a,b)]$ contains the optimal score for each feasible retention time.

After computing all entries $DP[v,(a,b)]$, we can find the optimal score of a solution by iterating over all entries $DP[M-b,(a,b)][t]$ for $(a,b) \in E$ and all feasible retention times $t \in [T-\varepsilon, T+\varepsilon]$. We can reconstruct a corresponding string starting from this entry.

The running time of DeNovoΔ is in $\mathcal{O}(|V| \cdot |E| \cdot d \cdot p)$ [17], where d is the maximal out-degree of a vertex in G and p is the maximal length of an edge label. The additional overhead of DeNovoΔLin (highlighted lines in Algorithm 2) is to iterate over all feasible retention times t for each entry $DP[v,(a,b)]$ and compute the new retention time t'. The number of scores to be stored varies depending on the entry and the retention time coefficients. For a path pair ending in v, respectively (a,b), we have to consider all retention times in $[rt_{\min} \cdot (v+b), rt_{\max} \cdot (v+b)]$, where $rt_{\min}$ and $rt_{\max}$ are the minimum and the maximum retention time per mass unit. For example, we only store one optimal score in entry $DP[0,(0,0)]$, but up to $\lceil rt_{\max} \cdot M - rt_{\min} \cdot M \rceil$ scores in entries $DP[M-b,(a,b)]$ for $(a,b) \in E$. The time complexity of DeNovoΔLin is in $\mathcal{O}(|V| \cdot |E| \cdot |RT_M| \cdot d \cdot p)$, where $|RT_M|$ denotes the number of possible retention times for a string of mass M. In practice, most entries $DP[v,(a,b)]$ contain only few scores and it is advisable to use a memory-efficient data structure instead of an array to reduces the memory consumption of the algorithm.

3.2 Position-dependent Prediction Model

In the position-dependent prediction model, the retention time of a string $\mathtt{S}$ is not equal to the retention time of all permutations of $\mathtt{S}$. The retention time coefficient of a character in the first and the last γ positions of the string may be different from the coefficient of the same character at another position. To compute the retention time of a path pair $P = (0,\ldots,v)$ and $Q = (0,\ldots,a,b)$ with $a \le v \le b$, we first have to distinguish the prefix and the suffix path. We compute the retention time of (P,Q) by summing the retention times t_P and t_Q of the path labels. Assuming that P is the prefix path and Q the suffix path,

$$t_P = \sum_{\mathtt{a_i} \in\, l(P)} \begin{cases} t_{\text{pre}}(\mathtt{a_i}, i) & i \le \gamma \\ t(\mathtt{a_i}) & i > \gamma \end{cases} \quad \text{and} \quad t_Q = \sum_{\mathtt{a_j} \in\, l(Q)} \begin{cases} t_{\text{suf}}(\mathtt{a_j}, j) & j \le \gamma \\ t(\mathtt{a_j}) & j > \gamma. \end{cases} \tag{7}$$

If we want to update the retention time after a balanced extension of (P,Q) by an edge (v,w), we have to compute the retention time of the edge label $l(v,w)$. This retention time depends on whether the edge label contains some of the first or the last γ characters of a

Algorithm 2 DeNovoΔLin – Linear retention time prediction model

```
for ((a,b) ∈ E and v ∈ V)
    DP[v,(a,b)] = array with entries −∞ for each feasible ret. time
DP[0,(0,0)][0] = 2
for (v ∈ V in ascending order):
  for ((a,b) ∈ E in asc. lex. order with a ≤ v ≤ b):
    for ((v,w) ∈ E with w + b ≤ M):
      for (entry t in DP[v,(a,b)]):
        t' = t + t_lin(l(v,w))
        if (w ≤ b):
          DP[w,(a,b)][t'] = max(
              DP[w,(a,b)][t'], DP[v,(a,b)][t] + gain((v,w),(a,b))
          )
        else:
          DP[b,(v,w)][t'] = max(
              DP[b,(v,w)][t'], DP[v,(a,b)][t] + gain((v,w),(a,b))
          )
```

solution string $\mathtt{S}$ of mass M. However, there can be multiple such solution strings resulting from different further balanced extensions of this path pair. Independently of the solution string $\mathtt{S}$, we can decide whether $l(v,w)$ contains some of the first γ characters given the length k of $l(P)$. If $k \geq \gamma$, the edge label clearly does not contain any of the first γ characters of any solution resulting from extending (P,Q). Likewise, we know that $l(v,w)$ contains none of the γ last characters if $l(Q)$ has more than γ characters. However, if $l(Q)$ has less than γ characters, we cannot decide whether $l(v,w)$ contains some of the last γ characters without knowing the length of the solution string. Let us assume for now that $l(v,w)$ does not contain some of the last γ characters of the solution. The retention time of the new path pair resulting the balanced extension of (P,Q) by the edge (v,w) is

$$t' = t + \sum_{\mathtt{a_i} \in l(v,w)} \begin{cases} t_{\text{pre}}(\mathtt{a_i}, i) & i + k \leq \gamma \\ t(\mathtt{a_i}) & i + k > \gamma. \end{cases} \tag{8}$$

If P would be the suffix path, $t_{\text{pre}}(\mathtt{a_i}, i)$ would be replaced by $t_{\text{suf}}(\mathtt{a_i}, i)$ in the above equation.

It is important that the above assumption holds for every balanced extension leading to a solution string $\mathtt{S}$. Otherwise, the retention time of the new path pair is not computed correctly. We cannot check if our assumption holds for an individual balanced extension. However, given a solution string $\mathtt{S}$ and a path pair that represents a prefix and a suffix of $\mathtt{S}$, we can check if either the balanced extension leading to this path pair or a preceding balanced extension did not satisfy the assumption. If so, either the prefix or the suffix path label has at least $n - \gamma$ characters, where n is the length of $\mathtt{S}$. This does also hold for all subsequent path pairs, as we only add characters to path labels in a balanced extension.

When reconstructing a solution from the dynamic programming table, we have to additionally check, if one of the path labels has $n - \gamma$ or more characters, before we combine them to a solution string. If so, the assumption was not fulfilled at some step and we discard this solution, as its retention time was not computed correctly. Note that we cannot consider these strings, unless they can be constructed by another sequence of balanced extensions. It is very unlikely that the assumption is not fulfilled in practice, as we consider small values of γ. We never observed such a situation in our evaluation with $\gamma = 2$.

In our dynamic program, we have to store some additional information to compute a solution with respect to the position-dependent prediction model. First, we have to store

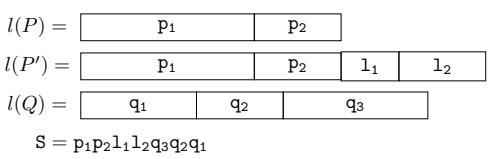

Figure 3 The retention time t of a path pair (P, Q) is the sum of the retention time coefficients up to the last characters $\mathtt{p}_2$ and $\mathtt{q}_3$. The path pair (P', Q) resulting from a balanced extension of (P, Q) by an edge with label $\mathtt{l}_1\mathtt{l}_2$ has retention time $t + t(\mathtt{p}_2, \mathtt{l}_1) + t(\mathtt{l}_1, \mathtt{l}_2)$. A path pair (P', Q) with $m(l(P')) + m(l(Q')) = M$ can be combined to a solution string $\mathtt{S}$ by concatenating $l(P')$ and the reversed string of $l(Q')$. The retention time of $\mathtt{S}$ is $t_{\text{nei}}(P', Q') + t(\mathtt{l}_2, \mathtt{q}_3)$.

whether P is a prefix or a suffix path. Second, we have to store the length of both path labels, unless they are larger than γ. DeNovoΔPos (Algorithm A.1 in the appendix) stores the optimal scores of path pairs ending in v and (a, b) in an array with an entry for every retention time t, the length α and β of the path labels and a Boolean flag *bit* indicating if the path ending in v is the prefix or the suffix path. Given the optimal score of a path pair, the algorithm performs every possible balanced extension with an outgoing edge of v, computes the new score and retention time, and updates the corresponding entries. We reconstruct a solution starting from a path pair ending in some vertex $M - b$ and some edge (a, b) and the algorithm additionally verifies that both the prefix and the suffix path label have more than γ characters. DeNovoΔPos considers at most $\gamma^2 \cdot |RT_M|$ optimal scores for each table entry $DP[v, (a, b)]$, where $|RT_M|$ is the number of possible retention times for a string of mass M. Therefore, the running time is in $\mathcal{O}\left(|V| \cdot |E| \cdot |RT_M| \cdot \gamma^2 \cdot d \cdot p\right)$, where d is the maximal out-degree of a vertex in G and p is the maximal length of an edge label.

3.3 Neighborhood-based Prediction Model

The neighborhood-based model predicts the retention time of a string $\mathtt{S}$ by considering all pairs of consecutive characters. We define the retention time of a prefix of $\mathtt{S}$ as the sum of the retention time coefficients of the pairs of consecutive characters and the additional coefficient of the first character. Note that we consider only one coefficient for the last character in the prefix. The other coefficient depends on the next character in $\mathtt{S}$ that is not part of the prefix. We define the retention time of a suffix analogously and compute the retention time of (P, Q) by summing the retention times of the path labels (Figure 3): We denote the two substrings by $l(P) = \mathtt{p}_1, \ldots, \mathtt{p}_n$ and $l(Q) = \mathtt{q}_1, \ldots, \mathtt{q}_m$. The retention time of (P, Q) is

$$t_{\text{nei}}(P, Q) = t(-, \mathtt{p}_1) + \left(\sum_{i=1}^{n-1} t(\mathtt{p}_i, \mathtt{p}_{i+1})\right) + \left(\sum_{i=m}^{2} t(\mathtt{q}_i, \mathtt{q}_{i-1})\right) + t(\mathtt{q}_1, -). \tag{9}$$

We can update the retention time after a balanced extensions of (P, Q) as follows. Consider a balanced extension of the prefix path P by an edge (v, w) with $l(v, w) = \mathtt{l}_1 \ldots \mathtt{l}_k$. Let $\mathtt{p}_n$ be the last character of $l(P)$. The retention time t' of the new path pair resulting from the balanced extension is $t' = t_{\text{nei}}(P, Q) + t(\mathtt{p}_n, \mathtt{l}_1) + \sum_{i=1}^{k-1} t(\mathtt{l}_i, \mathtt{l}_{i+1})$. The retention time of a solution $\mathtt{S}$ is not the sum of the retention times of a prefix of $\mathtt{S}$ and its complementary suffix. We have to additionally consider the coefficient of the last character of the prefix and the first character of the suffix, which are consecutive in $\mathtt{S}$. If we combine the path labels of a path pair (P', Q) to a string $\mathtt{S}$ (Figure 3), the retention time of $\mathtt{S}$ is $t_{\text{nei}}(P, Q) + t(\mathtt{p}_n, \mathtt{q}_m)$, where $\mathtt{p}_n$ and $\mathtt{q}_m$ are the last characters of P and Q.

DeNovoΔNei (Algorithm A.2 in the appendix) extends the algorithm DeNovoΔ and computes a solution with respect to the neighborhood-based prediction model as follows. Instead of storing the optimal score $DP[v,(a,b)]$ of a path pair ending in vertex v and edge (a,b), we distinguish prefix and suffix path and store an optimal score for each retention time t, last character $\mathtt{p}$ of the path ending in v. The algorithm considers at most $|\Sigma| \cdot |RT_M|$ optimal scores for each pair of a vertex v and an edge (a,b), where $|RT_M|$ is the number of possible retention times for a string of mass M and $|\Sigma|$ is the size of the considered alphabet. The running time of DeNovoΔNei is in $\mathcal{O}(|V| \cdot |E| \cdot |RT_M| \cdot |\Sigma| \cdot d \cdot p)$. We refer to the appendix for a more detailed description of the algorithm.

4 Experimental Evaluation and Discussion

In this section, we study the performance of our algorithms for de novo peptide sequencing with retention time prediction. We first describe the considered dataset and a method for estimating the parameters of the three models. Then, we compare the identification rates of the proposed algorithms to the identification rate of DeNovoΔ [17].

4.1 Dataset and Parameter Estimation

We use the SWATH-MS Gold Standard (SGS) dataset (`peptideatlas.org`, identifier PASS00289, [13]). Specifically, we consider the 944 spectra of synthesized peptides from DDA-experiments that have also been considered in [17]. The database search tool Comet [3] identified a sequence for each of these spectra using the very restricted database containing only the 422 synthesized peptides (see [17] for a detailed explanation). We randomly split the dataset into a training set with 80% of the spectra (755 spectra) and a test set with the remaining 20% of the spectra (189 spectra). We use the training set to estimate the retention time coefficients by linear regression and choose the tolerance parameter ε for each model using the test set. We choose the tolerance parameter ε based on the minimum and maximum prediction error (Appendix B) and set $\varepsilon = 1000$ (in seconds) for the linear prediction model and $\varepsilon = 750$ for the position-dependent model. The neighborhood-based prediction model requires many retention time coefficients for each character. Due to the small training dataset, the estimate of some coefficients is based on few observations and some cannot be estimated and are set to 0. A much larger training dataset would be necessary to train this model. We analyze all spectra for the linear and position-dependent model, but limit our evaluation of the neighborhood-based prediction model to some exemplary spectra. It is also be possible to use retention time coefficients reported in the literature (e.g. [9] and references therein), if training data is not available.

4.2 Comparison of DeNovoΔLin and DeNovoΔPos

We analyzed the 944 considered spectra with DeNovoΔLin and DeNovoΔPos. Both algorithms compute all solutions with a score of at least 90% of the optimal score and a predicted retention time within the tolerance range. Figure 4 shows a comparison of the identification rates of DeNovoΔ [17], DeNovoΔLin, and DeNovoΔPos. Without considering the retention time, DeNovoΔ reported the annotated sequence as best-scoring sequence for 586 spectra (62.1%). Considering the linear retention time prediction model, DeNovoΔLin computed the annotated sequence with an optimal score for 610 spectra (64.6%). DeNovoΔPos considers the position-dependent prediction model and achieved the highest identification rate. The annotated sequence was reported as best-scoring sequence for 629 spectra (66.6%). A filtering

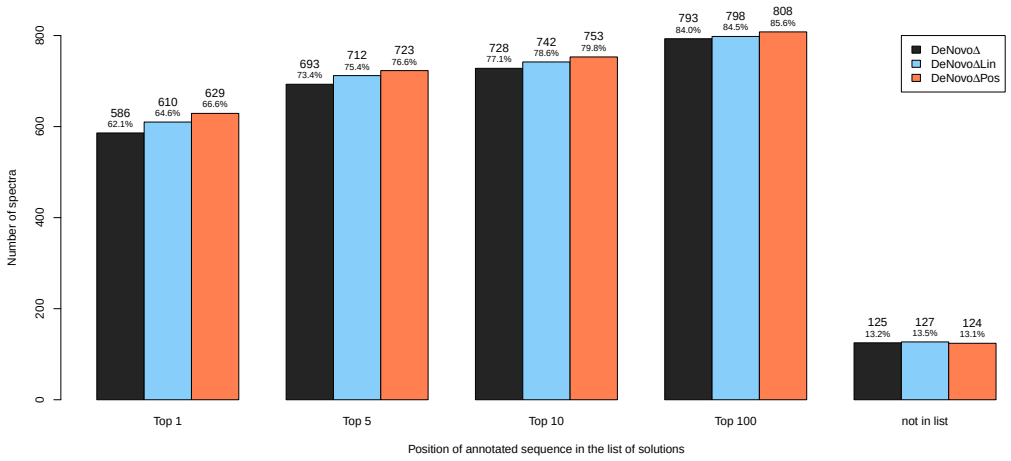

Figure 4 Position of annotated sequence in the list of reported sequences (sorted by score). DeNovoΔ reported the annotated sequence among the top 5 sequences in 73.4% of the spectra, DeNovoΔLin in 75.4% and DeNovoΔPos in 76.6% of the spectra.

approach that considers the top 100 sequences reported by DeNovoΔ, would not be as successful as the proposed algorithms. While the annotated sequence was reported by DeNovoΔ for 793 spectra among the top 100 sequences, DeNovoΔLin reported it in 798 cases and DeNovoΔPos in 808 cases. Even an optimal filtering approach by retention time would miss the sequences that have not been reported by DeNovoΔ. For few spectra, DeNovoΔLin and DeNovoΔPos did not report the annotated sequence, where DeNovoΔ did report it, as the predicted retention time of the annotated sequence was not in the chosen tolerance range.

4.3 Discussion

We develop algorithms for three additive retention time prediction models. However, we did not study the predictive robustness of our models and efficient methods for parameter estimation in detail. In the experimental evaluation, we especially studied the effect of considering the retention time information. We compare the performance of our algorithms to the algorithm DeNovoΔ [17] that uses the same scoring model, but no retention time information. An accurate retention time prediction model is crucial for exploiting the retention time information successfully, as the identification rates of our algorithms depend on the choice of the tolerance parameter ε. Increasing ε diminishes the effect of considering the retention time, while decreasing ε might exclude the correct sequence from the search space. This is especially an issue if the prediction model is not accurate, as for the neighborhood-based retention time model with our small training dataset. To get a glimpse on the performance of DeNovoΔNei, we set $\varepsilon = 500$ (in seconds) and analyzed the spectra from the test set, where the correct sequence was not excluded due to the predictive error. In three cases, the annotated sequence was reported by DeNovoΔNei, but by no other considered algorithm. The position of the annotated sequence improved compared to the position reported by DeNovoΔPos for 12 spectra.

The running time of our prototypical implementations is in some cases not yet practical. DeNovoΔLin needs less than 3 seconds per spectra for half of the considered spectra, but several hours in exceptional cases. In general, DeNovoΔPos is more time-consuming. Half of the spectra were analyzed within about 2 minutes. However, we note that we did not optimize our implementations for speed and memory usage.

5 Conclusion

In this paper, we propose the first algorithms for exploiting the retention time information in de novo peptide sequencing. We study three retention time prediction models and develop algorithms for computing a sequence that matches the experimental mass spectrum as well as possible and is in accordance with the observed retention time. The experimental evaluation of our algorithms shows that identification rates can definitively be improved by exploiting this additional information. Yet, the proposed algorithms score sequences with a very simplistic scoring function that only counts explained and measured masses, but does not consider any other available information. For real-world applications, a more evolved scoring function using all available information needs to be integrated. While [17] introduces a new scoring model, we explore ways of exploiting the retention time information. The proposed algorithms open room for developing new scoring functions that consider both the retention time information and the symmetric difference scoring model.

Acknowledgments. We would like to thank Peter Widmayer, Christian Panse, Witold Wolski, and Ludovic Gillet for helpful discussions. Moreover, we thank the reviewers for their constructive criticism.

References

1 Ting Chen, Ming-Yang Kao, Matthew Tepel, John Rush, and George M. Church. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 8(3):325–337, 2001. doi:10.1089/10665270152530872.

2 Vlado Dančík, Theresa A. Addona, Karl R. Clauser, James E. Vath, and Pavel A. Pevzner. De novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 6(3-4):327–342, 1999. doi:10.1089/106652799318300.

3 Jimmy K. Eng, Tahmina A. Jahan, and Michael R. Hoopmann. Comet: an open-source MS/MS sequence database search tool. *Proteomics*, 13(1):22–24, 2013. doi:10.1002/pmic.201200439.

4 Ludovic Gillet, Simon Rösch, Thomas Tschager, and Peter Widmayer. A better scoring model for de novo peptide sequencing: The symmetric difference between explained and measured masses. In *16th International Workshop on Algorithms in Bioinformatics, WABI 2016*, volume 9838, pages 185–196, 2016. (extended version: [17]). doi:10.1007/978-3-319-43681-4.

5 Christopher Hughes, Bin Ma, and Gilles A. Lajoie. De novo sequencing methods in proteomics. *Proteome Bioinformatics*, 604:105–121, 2010. doi:10.1007/978-1-60761-444-9_8.

6 Kyowon Jeong, Sangtae Kim, and Pavel A. Pevzner. UniNovo: a universal tool for de novo peptide sequencing. *Bioinformatics (Oxford, England)*, 29(16):1953–1962, 2013. doi:10.1093/bioinformatics/btt338.

7 Michael Kinter and Nicholas E. Sherman. *Protein Sequencing and Identification Using Tandem Mass Spectrometry*. Wiley-Interscience, New York, 2000. doi:10.1002/0471721980.

8 Oleg V. Krokhin. Sequence-specific retention calculator. Algorithm for peptide retention prediction in ion-pair RP-HPLC: Application to 300- and 100-A pore size C18 sorbents. *Analytical chemistry*, 78(22):7785–95, 2006. doi:10.1021/ac060777w.

9 Oleg. V Krokhin, Robertson Craig, Vic Spicer, Werner Ens, Kenneth G. Standing, Ronald C. Beavis, and John A. Wilkins. An improved model for prediction of retention times of tryptic peptides in ion pair reversed-phase HPLC: its application to protein peptide mapping by off-line HPLC-MALDI MS. *Molecular & cellular proteomics : MCP*, 3(9):908–19, 2004. doi:10.1074/mcp.M400031-MCP200.

10 Bin Ma. Novor: Real-time peptide de novo sequencing software. *Journal of The American Society for Mass Spectrometry*, 26(11):1885–1894, 2015. doi:10.1007/s13361-015-1204-0.

11 Luminita Moruz and Lukas Käll. Peptide retention time prediction. *Mass spectrometry reviews*, 2016. doi:10.1002/mas.21488.

12 Magnus Palmblad, Margareta Ramström, Karin E. Markides, Per Håkansson, and Jonas Bergquist. Prediction of chromatographic retention and protein identification in liquid chromatography/mass spectrometry. *Analytical Chemistry*, 74(22):5826–5830, 2002. doi:10.1021/ac0256890.

13 Hannes L Röst, George Rosenberger, Pedro Navarro, Ludovic Gillet, Saša M. Miladinović, Olga T. Schubert, Witold Wolski, Ben C Collins, Johan Malmström, Lars Malmström, and Ruedi Aebersold. OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data. *Nature biotechnology*, 32(3):219–223, 2014. doi:10.1038/nbt.2841.

14 Kosaku Shinoda, Masahiro Sugimoto, Masaru Tomita, and Yasushi Ishihama. Informatics for peptide retention properties in proteomic LC-MS. *Proteomics*, 8(4):787–98, 2008. doi:10.1002/pmic.200700692.

15 Vic Spicer, Marine Grigoryan, Alexander Gotfrid, Kenneth G. Standing, and Oleg V. Krokhin. Predicting retention time shifts associated with variation of the gradient slope in peptide RP-HPLC. *Analytical chemistry*, 82(23):9678–85, 2010. doi:10.1021/ac102228a.

16 Eric F. Strittmatter, Lars J. Kangas, Konstantinos Petritis, Heather M. Mottaz, Gordon A. Anderson, Yufeng Shen, Jon M. Jacobs, David G. Camp, and Richard D. Smith. Application of peptide LC retention time information in a discriminant function for peptide identification by tandem mass spectrometry. *Journal of Proteome Research*, 3(4):760–769, 2004. doi:10.1021/pr049965y.

17 Thomas Tschager, Simon Rösch, Ludovic Gillet, and Peter Widmayer. A better scoring model for de novo peptide sequencing: The symmetric difference between explained and measured masses. *Algorithms for Molecular Biology*, 12(1), 2017. (extended version of [4]). doi:10.1186/s13015-017-0104-1.

18 Susan K. Van Riper, Ebbing P. de Jong, John V. Carlis, and Timothy J Griffin. Mass spectrometry-based proteomics: Basic principles and emerging technologies and directions. *Advances in experimental medicine and biology*, 990:1–35, 2013. doi:10.1007/978-94-007-5896-4_1.

A Pseudocode of DeNovoΔPos and DeNovoΔNei

DeNovoΔPos (Algorithm A.1) computes the optimal score for a path pair with retention time t, a prefix path with label length α ending in vertex v and a suffix path with label length β ending in edge (a, b). The algorithm distinguishes prefix and suffix path, as the retention time of a string is different to the retention time of its reversed string. We store the length of the path labels only up to length γ, as the exact length is only important as long as the path labels have less than γ characters. If the algorithm reaches an entry $DP[v, (a, b)]$ in line 7, all optimal scores for path pairs ending in vertex v and edge (a, b) have been computed correctly, as all balanced extensions leading to such path pairs have already been considered. The algorithm then considers all balanced extensions by outgoing edges of v and computes the score and the retention time of the resulting path pairs. After computing all entries of the table, the solution can be reconstructed starting from an entry in some array $DP[M - b, (a, b)]$ for $(a, b) \in E$ with optimal score and a feasible retention time. The algorithm additionally has to check if the prefix and the suffix path label of the reconstructed solution have more than γ characters.

DeNovoΔNei (Algorithm A.2) computes and stores the optimal score for a path pair that ends in a given vertex v and a given edge (a, b) with a retention time t, where the path ending in v is a prefix (suffix) and `p` is the last character of the corresponding path label. As

Algorithm A.1 DeNovoΔPos – Position-dependent retention time prediction model

```
for ((a,b) ∈ E and v ∈ V):
    DP[v,(a,b)] = (|RT_M| × γ × γ × 2)-array initialized with −∞
DP[0,(0,0)][0,0,0,0] = 2
for (v ∈ V in ascending order):
    for ((a,b) ∈ E asc. lex. order with a ≤ v ≤ b):
        for (entry (t,α,β,bit) in DP[v,(a,b)]):
            for ((v,w) ∈ E with w + b ≤ M):
                t' = retention time of resulting path pair
                if (bit == 1):
                    α' = max(γ, α + |l(v,w)|); β' = β
                else:
                    α' = α; β' = max(γ, β + |l(v,w)|)

                if (w ≤ b):
                    DP[w,(a,b)][t',α',β',bit] = max(
                        DP[w,(a,b)][t',α',β',bit],
                        DP[v,(a,b)][t,α,β,bit] + gain((v,w),(a,b))
                    )
                else:
                    DP[b,(v,w)][t',α',β',¬bit] = max(
                        DP[b,(v,w)][t',α',β',¬bit],
                        DP[v,(a,b)][t,α,β,bit] + gain((v,w),(a,b))
                    )
```

a base case, the algorithm computes the optimal score for a path pair ending in vertex 0 and the loop edge $(0,0)$ as $DP[0,(0,0)][0,-,0]$. Note that there exists only one such path pair with retention time $t = 0$ and we define the last character as $-$. The algorithm considers the vertices and edges of G in ascending order. Whenever the algorithm reaches a vertex v and an edge (a,b), it has already computed the optimal score of path pairs ending in this vertex and this edge for any combination of retention time t, last character p. Given these scores, the algorithm considers all possible balanced extensions by outgoing edges of v and computes the score and the retention time of the resulting path pair. The algorithm updates the corresponding entries in the table and continues with the next pair of endpoints of a path pair. Finally, the optimal score can be computed by iterating over all entries $DP[M-b,(a,b)]$ and considering the feasible retention time interval and all possible last characters of the path ending in $M-b$. In contrast to DeNovoΔ, which has a running time in $\mathcal{O}(|V| \cdot |E| \cdot d \cdot p)$, where d is the maximal out-degree of a vertex in G and p is the maximal length of an edge label, DeNovoΔNei has to consider for each pair v and (a,b) all possible retention times and all possible last characters. That is, the algorithm considers for each pair v and (a,b) at most $|RT_M| \cdot |\Sigma|$ optimal scores and performs for each optimal score all possible balanced extensions. Therefore, the running DeNovoΔNei is in $\mathcal{O}(|V| \cdot |E| \cdot |RT_M| \cdot |\Sigma| \cdot d \cdot p)$, where $|RT_M|$ is the number of feasible retention times for a string of mass M and $|\Sigma|$ is the size of the alphabet.

B Parameter Estimation

In this work, we are mainly interested in the algorithmic problem of using retention time information for de novo sequencing and do not focus on efficient procedures for estimating

Algorithm A.2 DeNovoΔNei – Neighborhood-based retention time prediction model

```
for ((a,b) ∈ E and v ∈ V):
    DP[v,(a,b)] = (|RT_M| × |Σ| × 2)-array initialized with −∞
DP[0,(0,0)][0,-,0] = 2
for (v ∈ V in ascending order):
    for ((a,b) ∈ E in ascending order of a and b):
        for (entry (t,p,bit,score) in DP[v,(a,b)]):
            for ((v,w) ∈ E with w + b ≤ M):
                t' = retention time of resulting path pair

                if (w ≤ b):
                    p' = last character of l(v,w)
                    DP[w,(a,b)][t',p',bit] = max(
                        DP[w,(a,b)][t',p',bit],
                        DP[v,(a,b)][t,p,bit] + gain((v,w),(a,b))
                    )
                else:
                    p' = last character of l(a,b)
                    DP[b,(v,w)][t',p',¬bit] = max(
                        DP[b,(v,w)][t',p',¬bit],
                        DP[v,(a,b)][t,p,bit] + gain((v,w),(a,b))
                    )
```

the coefficients of our models. We use linear regression for estimating the coefficients for our three retention time models. Even with these simple models and estimation procedures, our method shows improved identification rates and an increased performance might be achieved by considering a larger dataset.

We consider 944 spectra from the SGS dataset and partition the dataset randomly into a training set containing 80% of the spectra for estimating the retention time coefficients and a test set containing the remaining 20% of the spectra for selecting the tolerance parameter ε. The retention time coefficients are estimated by linear regression. We choose the coefficients such that the sum of the squared loss $\sum_{\mathtt{S_i},T_i}(T_i - t(\mathtt{S_i}))^2$ is minimized, where T_i is the measured retention time, and $t(\mathtt{S_i})$ the predicted retention time of the annotated sequence $\mathtt{S_i}$. For example, for estimating the coefficients of the linear model, we first compute the occurrence vector for each sequence in the dataset. The occurrence vector of a sequence is a vector of length $|\Sigma|$ that indicates how often a character occurs in the sequence; e.g., the occurrence vector of the string `AGA` has value 2 at entry `A`, value 1 at entry `G` and value 0 at all other entries. Then, the retention time of a sequence `S` is the product of its occurrence vector $occ(\mathtt{S})$ and the vector of the retention time coefficients t. Standard software tools for statistical methods can be used to compute t, such that $\sum_i (T_i - t \cdot occ(\mathtt{S}))^2$ is minimized.

In order to choose the tolerance parameter ε, we analyzed the difference between the measured and the predicted retention time of the sequences in the test set. Figure B.1 shows the differences between the predicted and the measured retention times for all three models on the test dataset. Especially for the neighborhood-based prediction model, we have to predict many retention time coefficients for each character. Several coefficients are estimated based on few observations and others cannot be estimated at all. Therefore, we cannot extensively evaluate the identification rates of our algorithm with the neighborhood-based prediction model, as a much larger training dataset for estimating all parameters would be necessary. Our comparison of the identification rates regarding this prediction model is

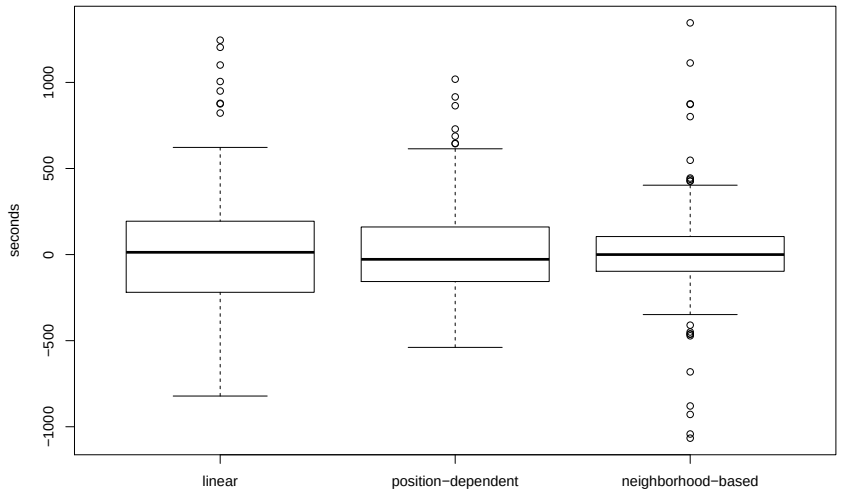

Figure B.1 Retention time prediction models – Difference between predicted and measured retention time of all sequences in the test set with respect to the three prediction model.

limited to some examples, where the prediction model works well.

We chose the tolerance parameter ε independently for each prediction model as half the difference between the maximum error $e_{\max}$ and the minimum error $e_{\min}$, i.e. $\varepsilon = (e_{\max} - e_{\min})/2$. Concretely, we set $\varepsilon = 1000$ for the linear prediction model and $\varepsilon = 750$ for the position-dependent model. The neighborhood-based prediction model has a very large predictive error for several sequences due to the small training dataset. For our limited evaluation, we ignore the 5 largest and the 5 smallest retention time errors when picking the tolerance parameter and use $\varepsilon = 500$.

C Experimental Evaluation – Supplementary Figures

In our experiments, we only considered spectra from peptides with an assumed (precursor) charge state 2 (as reported by Comet). Moreover, we assume that all measured fragment masses are singly charged, i.e. the mass-to-charge ratio is equal to the mass of a fragment. While it is possible to also consider spectra with higher charge states, the analysis of such spectra requires additional data preprocessing to convert the measured mass-to-charge ratios of the fragments to the corresponding masses (charge state deconvolution). Figure C.2 shows a distribution of the number of identified spectra with respect to the length of the corresponding peptide sequence. The position-dependent prediction model improves the identification rates on peptides with less than 15 amino acids, while the linear prediction model is favorable for longer amino acid sequences.

In the description of our algorithms, we only consider integer values and ignore the measurement accuracy. For our evaluation, we consider two masses two be equal if they differ by at most 0.02 Da. Moreover, as described in [17], we use a simple merging algorithm to reduce the size of the graph. We observed a great variation of spectrum graph sizes in our experiments. The spectrum graphs contained roughly 8400 edges on average, whereas the largest observed graph contained 23000 edges. Spectra measured on low resolution lead to denser spectrum graph, i.e. to a larger number of edges, but a lower number of vertices. However, we did not study the performance and runtime of our algorithms on this type of spectra.

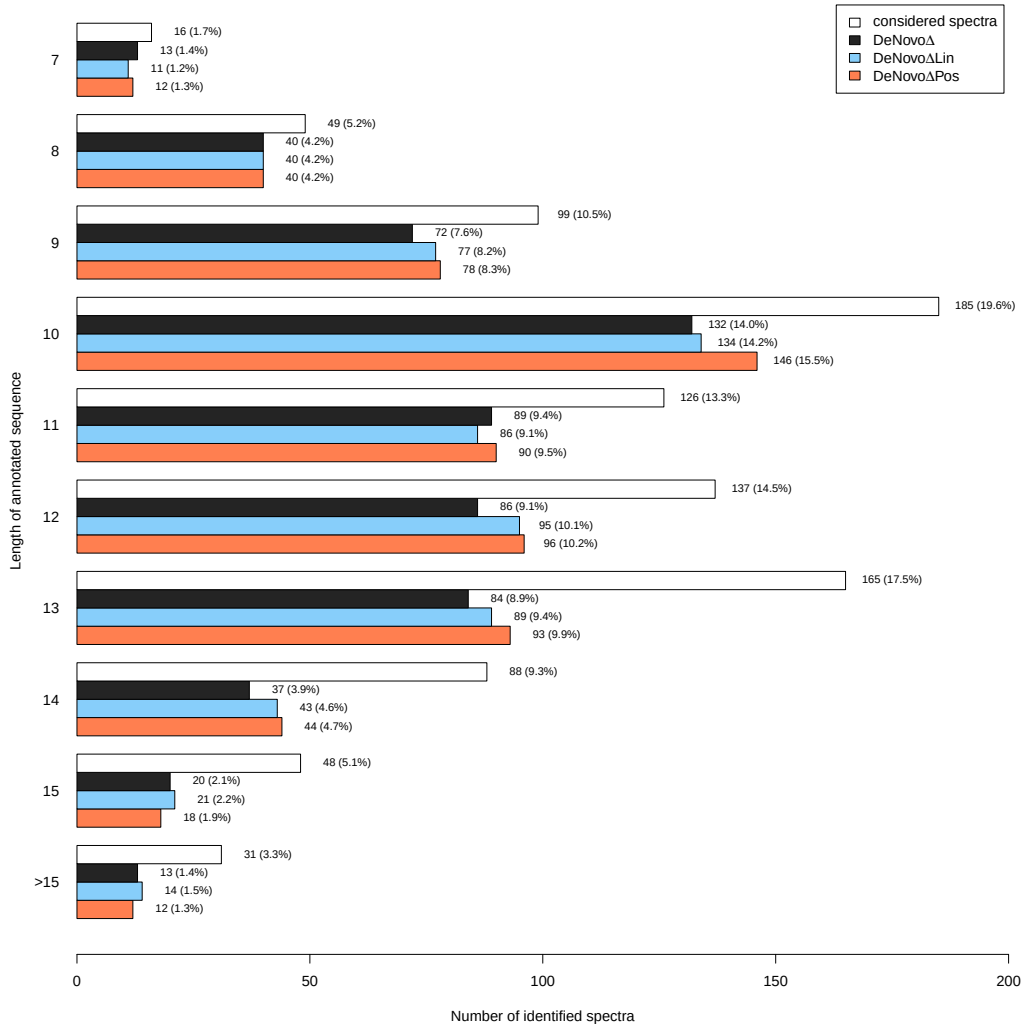

Figure C.2 Identified spectra with respect to the length of the annotated sequence.