Bioinformatics, 32, 2016, i322–i331
doi: 10.1093/bioinformatics/btw269
ISMB 2016

OXFORD

Faster and more accurate graphical model identification of tandem mass spectra using trellises

Shengjie Wang[1], John T. Halloran[2], Jeff A. Bilmes[1,2,*] and William S. Noble[1,3,*]

[1]Department of Computer Science and Engineering, [2]Department of Electrical Engineering and [3]Department of Genome Sciences, University of Washington, Seattle, WA 98195, USA

*To whom correspondence should be addressed.

Abstract

Tandem mass spectrometry (MS/MS) is the dominant high throughput technology for identifying and quantifying proteins in complex biological samples. Analysis of the tens of thousands of fragmentation spectra produced by an MS/MS experiment begins by assigning to each observed spectrum the peptide that is hypothesized to be responsible for generating the spectrum. This assignment is typically done by searching each spectrum against a database of peptides. To our knowledge, all existing MS/MS search engines compute scores individually between a given observed spectrum and each possible candidate peptide from the database. In this work, we use a *trellis*, a data structure capable of jointly representing a large set of candidate peptides, to avoid redundantly recomputing common sub-computations among different candidates. We show how trellises may be used to significantly speed up existing scoring algorithms, and we theoretically quantify the expected speedup afforded by trellises. Furthermore, we demonstrate that compact trellis representations of whole sets of peptides enables efficient discriminative learning of a dynamic Bayesian network for spectrum identification, leading to greatly improved spectrum identification accuracy.
Contact: bilmes@uw.edu or william-noble@uw.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

A critical problem in medicine and biology is accurately identifying the proteins in a complex mixture, such as a drop of blood. Solutions to this problem have many important applications, including the early detection of diseases and congenital defects (Walters *et al.*, 1996). The most widely used high throughput technology to identify proteins in complex mixtures is *tandem mass spectrometry* (MS/MS), whose output is a collection of tens of thousands of fragmentation spectra, each of which ideally corresponds to a single generating peptide. The core problem in the interpretation of MS/MS data involves identifying the peptide responsible for generating each observed spectrum, which we call the *spectrum identification problem*.

The most accurate methods to solve the spectrum identification problem employ a database of peptides (reviewed in Nesvizhskii, 2010). Given an observed spectrum, peptides in the database are scored, and the top scoring peptide is assigned to the spectrum. The pair consisting of an observed spectrum and a peptide sequence is referred to as a *peptide-spectrum match* (PSM).

In this work, we show how *trellises* may be used to make this database search significantly more efficient and accurate. A trellis is a data structure capable of representing an exponential size collection of strings in polynomial space. Trellises have been used to speed up inference in hidden Markov models (Huang and Soong, 1991; Jelinek, 1997; Young *et al.*, 1997), dynamic Bayesian networks (DBNs) and dynamic graphical models (DGMs; Ji *et al.*, 2006). In the context of MS/MS, we use a trellis to compactly represent the collection of candidate peptides associated with an observed fragmentation spectrum, i.e. peptides whose masses are close to the observed peptide mass associated with the spectrum. Using the trellis allows for the sharing of computation across candidate peptides. We describe how to apply trellises to any scoring function expressible as a DGM. This includes linear scoring functions such as the SEQUEST XCorr (Eng *et al.*, 1994), the score functions employed by X!Tandem (Craig and Beavis, 2004), Morpheus (Wenger and Coon, 2013), MS-GF+ (Kim and Pevzner, 2014) and OMSSA (Geer *et al.*, 2004), as well non-linear methods such as our recently proposed DBN for Rapid Identification of Peptides (DRIP; Halloran *et al.*, 2014).

Table 1. Notation used in this article

Symbol	Function
x	observed spectrum of length T_x, x_t is a (m/z, intensity) pair
t	index along m/z axis for DGM expansion, $t = 0, \ldots, T-1$
T	DGM frame unrolling amount and number of peaks in observed spectrum
$\mathscr{M}, \mathscr{M}^x$	precursor mass, precursor mass of spectrum x
$\mathscr{C}, \mathscr{C}^x$	precursor charge, precursor charge of spectrum x
B	number of bins of m/z axis quantization (i.e. typically 2000 for low-resolution data)
$\widehat{x}$	binned and processed observed spectrum of length B
$\widehat{x}'$	difference observed spectrum (used in XCorr).
y	a peptide
M_y	length of y
M	number of theoretical peaks in some peptide
$\widehat{y}$	binned sparse theoretical vector of length B (dot product with binned observed spectrum: $\langle \widehat{x}', \widehat{y} \rangle$)
$\acute{y}$	length M_y sequence of increasing incremental m/z values of fragment ions of y (used for trellis)
$\acute{Y}$	set of strings $\acute{y} \in \acute{Y}$ to be compressed into a trellis.
$\mathring{y}$	vector of peaks indices of y of length T
$\mathring{y}_{t_1:t_2}$	subsequence of $\mathring{y}$ from position t_1 to t_2
a	vector of peaks types of y of length T
n_i, n_s, n_t	trellis nodes
l	trellis links
Δ	trellis DGM transition variable
N	trellis DGM node variable
L	trellis DGM link variable
$\bar{z}_t$	observed child variable for use in a DGM (i.e. $\bar{z}_t = 1$ always)

Note: x and its variations denote the observed spectrum for different methods/contexts. y and its variations denote the theoretical spectrum for different methods/contexts.

For common MS/MS application settings, we prove and quantify the extent to which, in expectation, determining the top scoring data instance in a trellis is substantially more efficient than scoring each data instance individually. We then demonstrate empirically that trellises provide a significant reduction in the computational costs of the XCorr and DRIP scoring functions, ranging from 12- to 16-fold speedups in the low-resolution and high-resolution datasets examined here.

Next, we show that trellises may be used to discriminatively train DBNs for MS/MS, leading to significantly improved peptide identification accuracy. In particular, we modify the DRIP model, which was originally trained via maximum likelihood estimation, to instead employ discriminative training via maximum mutual information (MMI) estimation (Povey, 2003).

Maximum likelihood estimation maximizes the log-likelihood given a set of high-confidence PSMs, whereas MMI estimation maximizes the conditional log-likelihood between the high-confidence PSMs and a large 'background' collection of alternative PSMs. Thus, MMI estimation encourages learned parameters which both explain the high-confidence labels well and discriminate against the background labels. However, MMI estimation is costly because it requires computing denominator quantities over whole sets of peptides before being able to take a single gradient step in the parameter space; this is infeasible if considering each peptide in the set individually. We demonstrate that using trellises renders the discriminative training procedure feasible. Furthermore, we present empirical evidence that this discriminative approach significantly improves identification accuracy relative to the generatively trained DRIP model (Section 7.2), resulting in 11.2% and 6.2% more correct identifications at a stringent false discovery rate (FDR) of 0.5% for two datasets. The general trellis-based discriminative training procedure used herein is applicable to any DGM for any class of problem, but we are unaware of any previous work that does this.

The article is organized as follows. In Section 2, we formally define the spectrum identification problem and introduce the two database search scoring functions, XCorr and DRIP. We then define trellises in Section 3, which we utilize to compress the theoretical spectra of candidate peptides to be scored during database search. We show how graphical models may be used to efficiently traverse elements in a trellis (Section 3.3), enabling easy combination with any scoring function represented as a graphical model. Thus, in Section 4, we show how the vastly different scoring functions, XCorr and DRIP, may be represented as graphical models. The combination of trellises with these graphical model scoring functions (Section 5) allows for significantly faster database search (Section 7.1). The highly compressed trellises allow feasible discriminative training for DRIP (Section 6), providing significantly more accurate peptide identification accuracy (Section 7.2).

2 Database search in tandem mass spectrometry

We describe the spectrum identification problem as follows. Given an observed spectrum x (check Table 1 for notations in the paper) with precursor m/z $\mathscr{M}^x$ and precursor charge $\mathscr{C}^x$, and given a database $\mathcal{D}$, we wish to find the peptide $y \in \mathcal{D}$ responsible for generating x. Using the precursor m/z and charge, we constrain the set of peptides to be scored by setting a mass tolerance threshold, w, such that we score the set of *candidate peptides* $\mathbf{D}(\mathscr{M}^x, \mathscr{C}^x, \mathcal{D}, w) = \{y : y \in \mathcal{D}, |m(y)/\mathscr{C}^x - \mathscr{M}^x| \leq w\}$, where $m(y)$ denotes the mass of peptide y. Denoting an arbitrary scoring function as $f(y,x)$, the spectrum identification problem, for a given x, involves finding:

$$y^* \in \underset{y \in \mathbf{D}(\mathscr{M}^x, \mathscr{C}^x, \mathcal{D}, w)}{\operatorname{argmax}} f(y, x). \tag{1}$$

We study two very different database search algorithms: SEQUEST (Eng *et al.*, 1994) and DRIP (Halloran *et al.*, 2014). SEQUEST begins by quantizing and preprocessing the observed spectrum x into a vector $\widehat{x}$. Given a candidate peptide y, a theoretical spectrum $\widehat{y}$ (typically, a sparse vector) is constructed from y with length equal to that of $\widehat{x}$. This yields the XCorr score function:

$$\text{XCorr}(x, y) = \widehat{y}^T \left(\widehat{x} - \frac{1}{151} \sum_{\tau=-75}^{75} \widehat{x}_\tau \right) = \widehat{y}^T \widehat{x}', \tag{2}$$

where $\widehat{x}_\tau$ denotes the vector shifted by τ m/z units. XCorr is thus a foreground minus a background inner product, and is hence linear.

Our recently proposed DBN-based scoring function, DRIP (Halloran *et al.*, 2014), constructs a potentially non-linear alignment between the theoretical and observed spectra. A peptide's theoretical spectrum is given, and hidden variables are used to represent two MS/MS alignment phenomena: *insertions*, which are observed peaks that do not match a theoretical peak, and *deletions*, which are theoretical peaks that do not match an observed peak. The most probable alignment, i.e. sequence of insertions and deletions, is calculated via the max-product (or the Viterbi) algorithm (Bilmes, 2010), and peptides are scored using the log-likelihood of the most probable alignment. A key advantage of DRIP over most other scoring functions is that the alignment costs are automatically deduced using a machine learning method such as maximum likelihood estimation or (in the present paper) conditional likelihood.

For these two scoring functions, we define their corresponding DGMs in Section 4. We note that the DGM used to model XCorr may be used to model any linear MS/MS scoring function, including functions employed by many commonly used search algorithms (e.g. X!Tandem, the base scores for MS-GF+and OMSSA, and Morpheus). By combining these DGMs with trellises, we efficiently model the scoring of all candidate peptides in Equation (1) within a single data structure (Section 3), affording efficient discriminative training for MS/MS (Section 6) and significantly faster database search (Section 7.1).

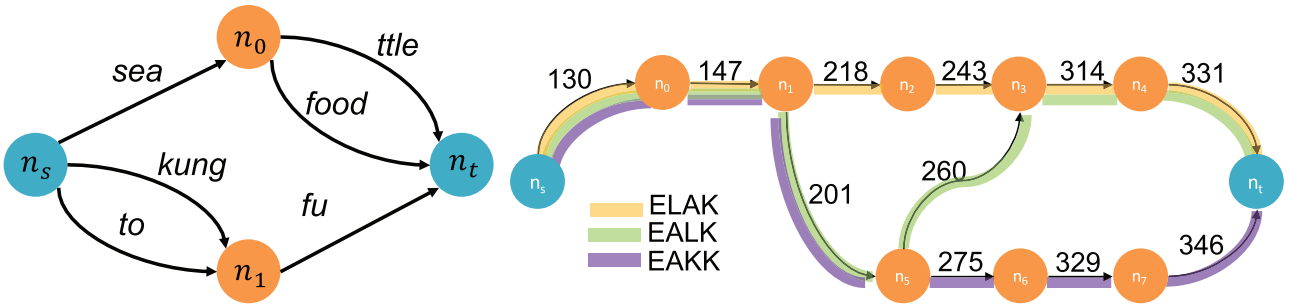
3 Trellises

Trellises are powerful structures capable of representing an exponentially large set of sequential data hypotheses compactly and efficiently, often using only linear space. For example, natural language dictionaries can be stored in trellises for more efficient querying; in speech recognition, trellises constructed out of the top phoneme (or word, or sentence) hypotheses (e.g. *N*-best lists) can be used to rescore and select the best hypothesis much more efficiently than a simple linear max computation over all *N* scores (Dyer *et al.*, 2008; Huang and Soong, 1991; Jelinek, 1997; Young *et al.*, 1997). In this section, we first formally define a trellis, then show how to efficiently construct a trellis given a large (exponential) set of strings, and lastly show how to apply the constructed trellises to MS/MS scoring.

A trellis over an alphabet Σ is a directed graph $G = (¥, \mathbb{L}, n_s, n_t)$, where ¥ is the node set, $\mathbb{L}$ is the link set, and $n_s, n_t \in ¥$ denote the source and target nodes, respectively. To avoid confusion, we use the terms 'vertex' and 'edge' for graphical model graphs (Section 3.3), and we use 'node' and 'link' for trellis graphs. Every link $l \in \mathbb{L}$ is a tuple $(n_1, n_2, \alpha(l))$, where n_1, n_2 are the from-node, and to-node respectively, and $\alpha(l) \in \Sigma$ is the alphabet element encoded in l. Each path in the trellis from n_s to n_t represents a sequence. Thus, letting $P(G, n_s, n_t)$ denote the set of paths from n_s to n_t, every $p = l_1, l_2, \ldots, l_{|p|}$, $p \in P(G, n_s, n_t)$ represents a sequence of characters, or a string, over alphabet Σ: $\alpha(l_1), \alpha(l_2), \ldots, \alpha(l_{|p|})$. Also, let $P(G, l_1, l_2)$, where $l_1, l_2 \in \mathbb{L}$, denote the set of paths starting with l_1 and ending with l_2.

For a set of strings $Ý = \{ý_1, ý_2, \ldots, ý_m\}$, the trellis representation of the strings $G(Ý)$ can be extremely compact because the elements of $Ý$ might be highly redundant. For example, if $ý_i$ and $ý_j$ share some substring in common, then we can merge the common parts into a sequence of common links. Figure 1A shows a trellis over four character strings, with the shared substrings collapsed into common trellis links to reduce redundancy.

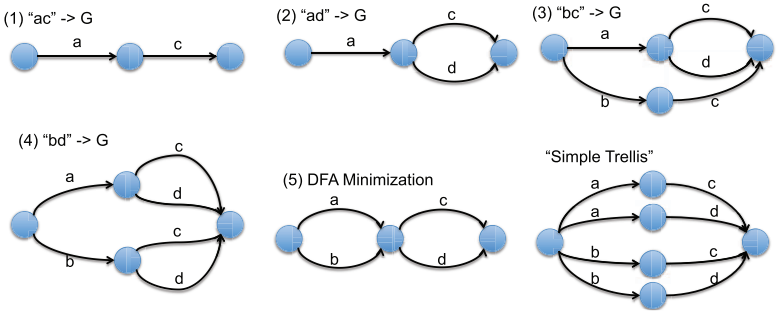
The common links of trellises not only reduce memory requirements when representing a set of strings but can also speed up computations over the encoded string set, sometimes quite significantly.



(A) (B)

$G = \{\{n_s, n_t\}, \emptyset, n_s, n_t\}$ // initialize an empty trellis ;
Assign an ordering in Σ, and sort $Ý$ lexicographically;
for $ý \in Ý$ **do**
  Split $ý$ into prefix $ý[1 : i-1]$ and suffix $ý[i : end]$, where $ý[1 : i-1]$ is the longest prefix that is consistent within G;
  Traverse G with prefix $ý[1 : i-1]$ stopping at node n';
  Build a chain structure C out of the suffix $ý[i : end]$, where each link on the chain corresponds to one character;
  Merge C into G by merging the start node of C with n' and the end node of C with n_t;
end
Run DFA minimization on G; Return G;

(C)



(D)

Fig. 1. (A) A trellis encoding 'seattle', 'seafood', 'kungfu' and 'tofu'. $n_s, n_0, n_1,$ and n_t are trellis nodes, and every arrow corresponds to a trellis link, e.g. $(n_s, n_0,$'sea'). **(B)** An example of a simple trellis for MS/MS scoring functions, consisting of the theoretical peaks (discretized b/y-ions) for three peptides: 'ELAK', 'EALK' and 'EAKK'. Every edge corresponds to the m/z value of a fragment ion rounded to the nearest integer. Three colored paths from source node n_s to sink node n_t correspond respectively to three peptides. The observed spectrum is charge +2 with low-resolution fragment ions. **(C)** Trellis construction algorithm that takes as input a set of strings $Ý$ and the corresponding alphabet Σ and returns a trellis representation of $Ý$. **(D)** Sample trellis construction for strings: 'ac', 'ad', 'bc' and 'bd'.

For this article, we focus on the task of DGM inference with the Viterbi algorithm. Trellises allow us to reduce the state space of the Viterbi algorithm and to apply smart pruning strategies more effectively, achieving orders of magnitude reductions in computation time (Section 7.1).

3.1 Trellis construction

Constructing the optimal trellis from the input set of strings $\acute{Y}$ over alphabet Σ is a difficult problem. The objective of the 'optimal' trellis is task dependent. For example, for natural language dictionary queries to be computationally optimal, the trellis should have a minimal number of nodes. For data compression, trellises are often stored as a node list and a link list, where each entry of the link list records the starting/ending node and possibly some additional features. The optimal trellis should, thus, be minimal in overall size, so both the nodes and links matter. Moreover, some tasks do not require the trellis to be an 'exact' representation of the input strings. For a set of strings $\acute{Y}$, let $G(\acute{Y})$ be a trellis representation, and for a trellis G, let $\mathbb{T}(G)$ be the set of strings represented by the trellis. We define an *exact trellis* to be one where precisely $\mathbb{T}(G(\acute{Y})) = \acute{Y}$. For our task of speeding up DGM training/inference for MS/MS database search, our objective is to construct a trellis $G(\acute{Y}) = (¥, \mathbb{L}, n_s, n_t)$ that is exact but where $|\mathbb{L}|$ is minimized.

Constructing the optimal trellis is a hard problem, as we can think of a trellis as a non-deterministic finite automaton (NFA; Hopcroft *et al.*, 2001), and it has been proven (Schnitger and Gramlich, 2005) that NFA minimization in terms of the number of states/transitions (trellis nodes/links) is NP-hard to approximate within a constant factor. We have hence developed a heuristic algorithm (Fig. 1C) that is similar to the determinize–minimize procedure of Watson (1993) but that is specialized to sets of MS/MS theoretical spectra. The resulting trellis $G(\acute{Y})$ is the minimum state deterministic finite automaton (DFA; Hopcroft *et al.*, 2001) of the given language of peptides $\acute{Y}$.

An example run of this algorithm for the strings 'ac', 'ad', 'bc' and 'bd' is depicted in steps (1)–(5) of Figure 1D. The for loop, which merges prefixes of input strings, constructs a DFA out of $\acute{Y}$. Minimization on the constructed DFA can be thought of as a process that merges nodes which share the same suffixes. Both merging prefixes and suffixes reduce the number of links in the trellis, making the algorithm a powerful heuristic in practice. The complexity of the algorithm is bounded by the DFA minimization step. With Hopcroft's algorithm (Hopcroft, 1971), the running time is $\mathcal{O}(|\Sigma||\acute{Y}|l_{\max}\log(|\acute{Y}|l_{\max}))$, where $l_{\max} = \max_{\acute{y}\in\acute{Y}}|\acute{y}|$.

3.2 Trellises for MS/MS scoring functions

To speed up the scoring of a set of candidate peptides, we construct a trellis, G_p, consisting of the set of theoretical peaks from the candidate peptides to be scored (i.e. peptides whose masses lie within the specified precursor mass tolerance window). The alphabet Σ_p for G_p contains all possible peak m/z values (in Thomsons) discretized according to the resolution of the dataset (e.g. for low-resolution fragment ion spectra, the values are rounded to the nearest integers). Figure 1B gives an example of a trellis constructed over the theoretical spectra of three peptides.

For each observed spectrum, a trellis is thus constructed containing the theoretical peaks of all peptide candidates within the corresponding mass window. Each trellis then becomes a 'database' to search for the best peptide candidate match. All trellises (one for each 1 Th mass bin) are pre-computed and stored during a database indexing step prior to search. The complexity of construction is bounded by the DFA minimization step as stated above ($\mathcal{O}(|\Sigma||\acute{Y}|l_{\max}\log(|\acute{Y}|l_{\max}))$), and for trellises of theoretical peptide peaks, $|\Sigma|$ is the number of distinct peak m/z values and is a constant based on the resolution of the data, $|\acute{Y}|$ is the number of theoretical peptides in the querying mass window, and $l_{\max}$ is the number of theoretical peaks of the longest peptide candidate in the mass window.

We next show how MS/MS trellis traversal can be expressed and implemented with DGMs (Section 3.3). This enables the combined use of trellises with both linear and non-linear MS/MS scoring functions expressed by DGMs (Section 4), thereby allowing significantly faster database search (Section 5.1) and efficient generative and discriminative training for improved identification accuracy (Section 7.2).

3.3 Traversing trellises using DGMs

A graphical model compactly represents the factorization properties of a family of probability distributions defined over a set of random variables. In a graphical model's graph, vertices represent random variables and edges denote allowable direct interaction between variables. A Bayesian network is one type of graphical model that uses directed acyclic graphs. DGMs are defined over temporal sequences where each element in the sequence (called a *frame*) is represented by a repeated set of vertices and edges. DGMs provide a great deal of modeling power and flexibility, offering a calculus with which to construct widely varying and potentially very complex models to reason about the underlying data while providing strategies to maintain tractable inference. DBNs are DGMs where the graphs are directed and acyclic.

As in Ji *et al.* (2006), we can use DGMs to traverse over a trellis. At frame t, we use three vertices to access the trellis: a trellis-node vertex V_t, a trellis-link vertex L_t, and a transition vertex Δ_t. Intuitively, V_t corresponds to a node in the trellis, L_t corresponds to a link in the trellis and Δ_t controls the traversal of the trellis. V_{t-1}, V_t, and Δ_t determine the set of possible values for L_t, with each value of L_t corresponding to one character in the encoded strings.

Our trellises can be represented as a DGM structure (Fig. 2). Values $V_t = n_i$ and $\Delta_{t+1} = d$ $(d \geq 0)$ determine the allowable set of trellis nodes $V_{t+1} \in \{n_j \in N | \exists p \in P(G, n_i, n_j), |p| = d\}$ (V_{t+1} has 0 probability for values not in the allowable set, and has the same probability for all values in the allowable set). Also, values $V_t = n_i$, $V_{t+1} = n_j$, and $\Delta_{t+1} = d$ together determine the allowable set of links $L_{t+1} \in \{l \in \mathbb{L} | \exists p \in P(G, n_i, n_j), |p| = d, p[d-1] = l\}$. Thus, L_{t+1} is a random variable corresponding to all links that go into n_j and can be reached from n_i with a path of length d. If $d = 0$ (i.e. a

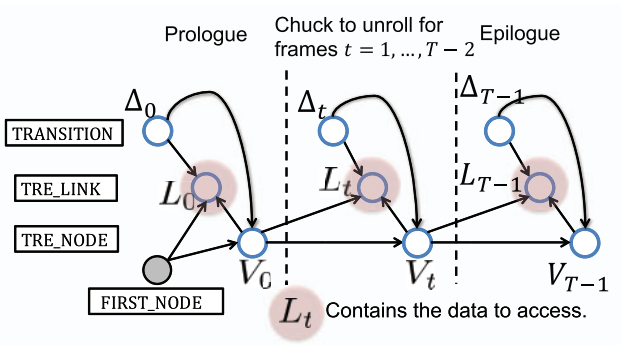


Fig. 2. DBN for traversing a trellis. L_t corresponds to the set of links being traversed, which contains the data to access. The value of L_t is decided based on the previous node V_{t-1}, the current node V_t, and the transition Δ_t.

zero length path connecting two nodes), then the algorithm stays at the current node and the link stays put as well, i.e. $L_{t+1} = L_t$. In the simplest case, Δ_t is binary (so $\Delta_t \in \{0, 1\}$), so that if $\Delta_{t+1} = 0$, the algorithm stays at the current node and link, and if $\Delta_{t+1} = 1$, L_{t+1} may be the outgoing link incident to n_i, and V_{t+1} the set of corresponding destination trellis-nodes for those outgoing links. Taking Figure 1A as an example, suppose $V_t = \{n_0, n_1\}$ and $\Delta_{t+1} = 1$, then $V_{t+1} = \{n_t\}$ and $L_{t+1} = \{\cdot$ttle', 'food', 'fu'$\}$. The 'FIRST_NODE' vertex is observed to be the node value n_s and is used only for initializing the time-dependent structure.

The complexity for constructing the conditional probability table (CPT) to store $\Pr(L_t|V_t, V_{t+1}, \Delta_{t+1})$, which is required for traversing the trellis as described above, is $|\{(l_1, l_2) : l_1, l_2 \in \mathbb{L}, \exists p \in P(G, l_1, l_2), |p| \leq d_{\max}\}|$, where $d_{\max}$ is the largest value Δ_t may take (i.e. the maximum number of deletions). The CPT is likely quite sparse because many paths do not exist. The value of $d_{\max}$ can vary based on the underlying DGM. If only link-by-link traversal is desired, then $d_{\max} = 1$. Setting $d_{\max} = \infty$ encodes all subsequences of the data instance in the trellis. The CPT can be constructed online to save memory. Our implementation in the Graphical Model Toolkit (GMTK) (Bilmes and Zweig, 2002) efficiently supports sparse trellis CPTs.

A trellis DGM representation is applicable to any DGM that accesses data in a sequential manner. Rather than accessing data in the traditional way, the trellis representation, moreover, offers two major benefits over accessing sets of sequences in the traditional way, namely: (i) various pruning and approximate inference strategies can be applied locally that speed up the underlying DGM significantly, since pruning causes many trellis paths to be removed simultaneously; (ii) compressed trellis representation makes practical certain expensive learning methods that requires access to the entire set of data, such as discriminative training. Below, we show how we take advantage of both of these benefits in our trellis/DGM-based peptide-spectrum score functions.

4 Graphical model MS/MS scoring functions

4.1 Linear scoring via graphical models

Many MS/MS scoring functions, including XCorr, X!Tandem, the base scores for MS-GF+ and OMSSA, and Morpheus (Wenger and Coon, 2013), are *linear* in the theoretical and observed spectra (i.e. they constitute a dot product between two preprocessed vectors corresponding to the theoretical and the observed spectrum). Using *virtual* evidence (Bilmes, 2004; Pearl, 1988), where the conditional distributions of observed child variables may be unnormalized non-negative scores, we may easily represent any MS/MS linear scoring function in a graphical model as the log-likelihood of a mixture-like model. This can then be combined with the aforementioned trellis constructs. In this section, we first describe linear score functions and then, in Section 4.2, show how a non-linear score function is also compatible with trellises.

Let $\bar{z}_t$ be an observed child variable that is always observed to be a fixed and known value (e.g. unity). Such a child is known as 'virtual evidence' since it may be used to impart a soft version of evidence into a model as follows: given a distribution $\Pr(y_t)$ over a random variable y_t, the construct $\Pr(\bar{z}_t|y_t)\Pr(y_t)$ is a function of only y_t but is a generalization of evidence for y_t. For example, if $\Pr(\bar{z}_t|y_t) = \delta(y_t = \bar{y}_t)$, where δ is a Kronecker delta, then this would be the same as y_t being observed at value $\bar{y}_t$. If, on the other hand, $\Pr(\bar{z}_t|y_t)$ is a non-negative vector (indexed by y_t) of real values, this imparts virtual evidence for different values of y_t. Another construct we utilize (Bilmes and Zweig, 2002) is that of 'switching parents' and 'switching weights'. Let a_t be what is known as a switching parent, and consider a conditional distribution $\Pr(\bar{z}_t|y_t, a_t)$. When a_t is a switching parent, the current value of a_t determines the subset of other parents of $\bar{z}_t$ that are active. For example, $a_t = 3$ might say that $\bar{z}_t$ is no longer dependent on y_t. The construct of switching weights, moreover, allows a_t to determine an exponential weight of the distribution. That is, $\Pr(\bar{z}_t|y_t, a_t) \propto (\Pr(\bar{z}_t|y_t))^{w_{a_t}}$, where $\Pr(\bar{z}_t|y_t)$ is some locally normalized y_t-dependent distribution on $\bar{z}_t$, and w_{a_t} is some non-negative a_t-dependent constant weight. More details of these constructs are given in Bilmes (2004).

Virtual evidence, along with switching weights, allows us to express any linear (i.e. dot-product) MS/MS score within a DGM in a way that is ideally suited to DGM-expressed trellises. Given a peptide y, recall that its binned theoretical spectrum, $\hat{y}$, is a length-B sparse vector that corresponds to the positions, along the binned m/z axis, where there are peaks in a theoretically derived spectrum. Let $\mathring{y}$ be an increasing-order sorted vector of indices from 0 to $B-1$, that is $\mathring{y} = (0, 1, \ldots, B-1)$. Also, recall that $\hat{x}'$ is the length-B processed observed spectrum that is utilized in a dot-product scoring function $\langle \hat{x}', \hat{y} \rangle$ (for example, Equation (2) in the case of XCorr).

Most MS/MS linear scoring functions use different weights depending on the type of theoretical peak. For instance, in XCorr, b- and y-ions are each assigned weight 50, and the neutral losses of ammonia, water and carbon monoxide are each assigned weight 10. These weights are the unique values in the vector $\hat{y}$ and are then multiplied by the corresponding processed observed spectrum $\hat{x}'$, as expressed by the dot-product $\langle \hat{x}', \hat{y} \rangle$. Let $a = (a_0, a_1, \ldots, a_{B-1})$ be a length-B vector of peak type indices, so there are B peak types indices, one for each peak. Each a_t is integer valued and takes on values $a_t \in \{0, 1, \ldots, k\}$ for k peak types. Since $\hat{y}$ is sparse, some of the positions along the binned m/z are empty, and we encode the empty condition of position t as $a_t = 0$. Moreover, let $w = (w_0, w_1, \ldots, w_k)$ be a fixed vector of peak type non-negative weights, one weight for each of the k possible peak types, and where $w_0 = 0$. That is, the value of $w(a_t)$ depends on the ion type of the theoretical t-th theoretical peak (i.e. whether it is a b-/y-ion or a neutral loss), or if the peak is non-present. Therefore, $(\hat{y}(\mathring{y}_0), \hat{y}(\mathring{y}_1), \ldots, \hat{y}(\mathring{y}_{B-1})) = (w(a_0), w(a_1), \ldots, w(a_{B-1}))$. Now for $t = 1, \ldots, B$, define a virtual evidence factor so that $\log\Pr(\bar{z}_t|\,\mathring{y}_t) = \hat{x}'(\mathring{y}_t)$. Then with virtual evidence, and switching weights, we produce a probability model as follows:

$$\Pr(\bar{z}_{0:B-1}, \mathring{y}_{0:B-1}, a_{0:B-1}) = \prod_{t=0}^{B-1} p(u_t)p(a_t)p(\bar{z}_t|\mathring{y}_t, a_t) \quad (3)$$

$$\propto \prod_{t=0}^{B-1} p(\mathring{y}_t)p(a_t)(\Pr(\bar{z}_t|\mathring{y}_t))^{w(a_t)} \quad (4)$$

When it is the case that the current theoretical peptide is observed, then both $y_{0:B-1}$ and $a_{0:B-1}$ are also observed, and if at these observed values we set $p(\mathring{y}_t) = p(a_t) = 1$, we, therefore, get that $\log\Pr(\bar{z}_{0:B-1}, \mathring{y}_{0:B-1}, a_{0:B-1}) = \langle \hat{x}', \hat{y} \rangle + \text{const}$. All of the above can be expressed with a graphical model of length $T = B$ (Fig. 3A). If a sequencer over a trellis determines the vector variables $\mathring{y}_{0:T-1}$ and $a_{0:T-1}$ (which are no longer observed), then the specifics of the dot-product be directly controlled and used via a trellis, and all of the aforementioned benefits of trellises become applicable (see Fig. 4B and details in Section 5).

4.2 Non-linear scoring via graphical models

A linear model is not the only one that can be used with trellises with a DGM. DRIP is a model that (potentially non-linearly) aligns

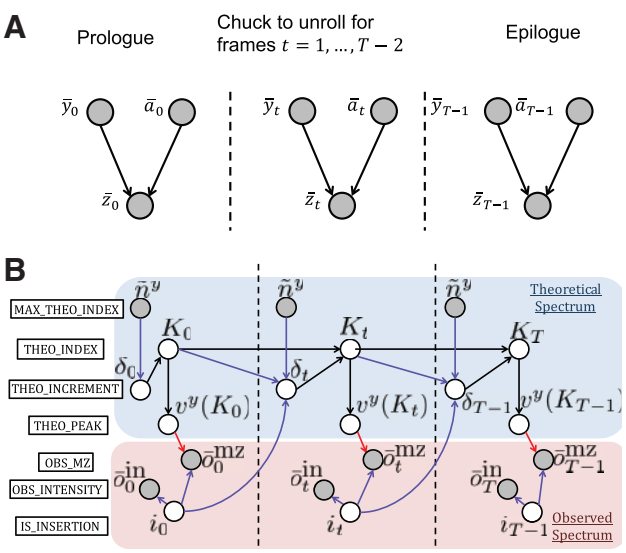


Fig. 3. (A) The graphical model representation of linear MS/MS scoring functions. **(B)** DRIP template. Shaded vertices are observed variables, while unshaded vertices are hidden variables. Black edges correspond to deterministic functions of parent variables, red edges correspond to conditional Gaussian distributions and blue edges represent switching parents.

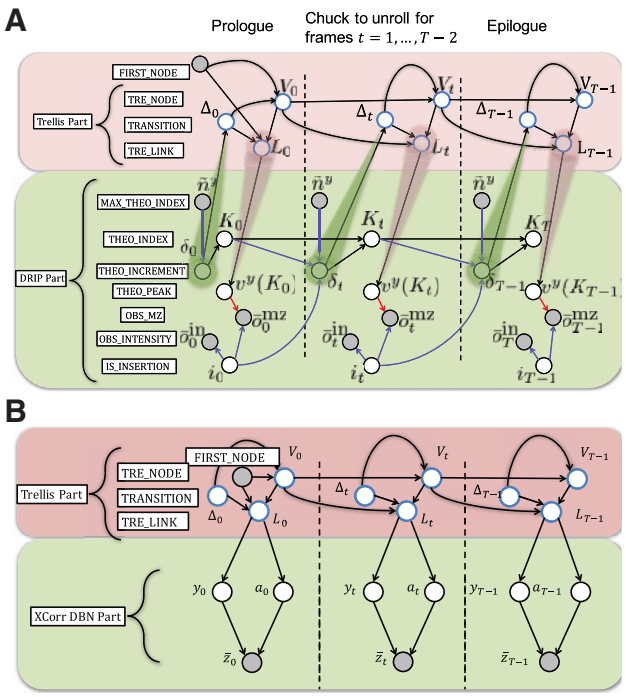


Fig. 4. (A) The DRIP trellis model. The trellis DBN (Trellis Part) is attached to the DRIP DBN (DRIP Part) by taking the input from δ_t from DRIP (green cone), which controls the traversal of theoretical peaks, and outputting L_t for DRIP to score (pink cone), which is the m/z values of theoretical peaks. The DRIP DBN structure remains unchanged (the part with green background is unchanged from Fig. 3B). **(B)** The graphical model representation of linear MS/MS scoring functions incorporated with trellis structure. Trellis Part (pink background) is attached to the linear MS/MS function graphical model (green background), which remains unchanged.

an unquantized observed spectrum with a peptide's theoretical spectrum. In DRIP, the theoretical spectrum is represented by hidden variables, as are constructs corresponding to insertions (spurious observed peaks) and deletions (missing theoretical peaks). An instantiation of the random variables in DRIP thus correspond to an alignment between the theoretical and observed spectra, where an alignment corresponds to the sequence of theoretical peaks used to score observed peaks as well as the sequences of insertions and deletions. During exact probabilistic inference, all possible alignments between the theoretical and observed spectra are considered, and the most probable alignment is used to score a peptide. We next describe particulars of how DRIP aligns spectra, and in Section 5 we show how it naturally combines with trellises.

The DRIP DGM template is displayed in Figure 3B, where the middle frame (the chunk) is dynamically expanded to fit the length of the current observed spectrum; that is, like in Figure 3A, DRIP considers the observed spectrum as the temporal sequence being modeled. Let $\tilde{n}^x$ be the number of frames and t be an arbitrary frame number. Each frame of the model corresponds to a single observed peak, with observed variables $\bar{o}_t^{\text{mz}}$ and $\bar{o}_t^{\text{in}}$ corresponding to the t-th m/z value and intensity, respectively. Parent to these variables, the Bernoulli random variable i_t denotes whether an observed peak is an insertion (in which case, we score these observations using a constant penalty) or not (in which case, we score these observations using a Gaussian centered along the m/z access—a Gaussian centered near the current frame's theoretical peak will of course score much higher than a Gaussian centered farther away).

To score a sequence of observed peaks using a set of theoretical peaks (which corresponds to a particular theoretical spectrum), a sequencer, expressed using a set of hidden random variables, probabilistically traverses through the spectrum from left to right, and this corresponds to the blue shaded portion of Figure 3B. Let v^y be a vector containing a theoretical peptide's fragment ions, sorted in increasing order. We index into elements of this vector via the random variable K_t, which denotes the element index of v^y in a particular frame. The non-negative, discrete random variable δ_t indicates the number of theoretical peaks we skip when advancing in the model by one frame; hence, $\Pr(K_{t+1} = i | K_t = j) = \mathbf{1}_{\{i=j+\delta_t\}}$. Hence, the number of deletions that occur after frame t is $\delta_t - 1$. The observed variable $\tilde{n}^y$, δ_t's parent, ensures that δ_t does not increment past the number of remaining theoretical peaks in the current theoretical peptide.

We note that despite the use of probability in DRIP to traverse through a theoretical peptide's spectrum to achieve an alignment with an observed spectrum, only one theoretical peptide is considered at a time to control the sequencer. In the next section, we describe trellises which mitigate this limitation.

5 Connecting trellises with graphical model MS/MS scoring functions

Figure 4A shows the DGM for DRIP that uses trellises. The vertex δ_t, which controls the number of deletions, behaves similarly to the transition random variable Δ_t in the trellis representation in DGMs, and we feed the value of δ_t into Δ_t (green cone in Fig. 4A). As δ_t ranges from 0 to the maximum length of the candidate peptides, all subsequences of a peptide are encoded in the trellis.

The value of L_t, which contains the set of m/z values of b/y-ions stored as trellis links given values of V_t and Δ_t, is fed into the 'THEO_PEAK' vertex $v^y(K_t)$ (pink cone in Fig. 4A) for peptide scoring (Section 4). In general, the trellis variant of DRIP scores all the candidate peptides all together unlike standard DRIP, which scores candidate peptides separately and one by one. Thus, the trellis acts like a database for querying theoretical peaks, and it does not affect the mechanism of the underlying DGM but it does allow joint decoding and reuse of common computational patterns.

Similar to DRIP, candidate theoretical peaks for XCorr can be represented as a trellis and accessed via the node, transition, link and index variables, as in the 'Trellis Part' of Figure 4B (the transition is always 1 for XCorr). However, unlike DRIP, XCorr requires that a theoretical peak be weighted differently depending on the corresponding ion type, i.e. whether it is a b/y-ion or neutral loss. To traverse two sets of sequences simultaneously within the same trellis—the sequence of theoretical peaks as well as the sequence of theoretical peak ion types—an extra bit is appended to each theoretical peak in the trellis. This bit denotes the value of the ion type variable, which acts as a switching parent and changes the weight of the theoretical peak (Section 4). This mechanism may be easily extended to include any variable number of ion types.

5.1 Speeding up graphical model inference with trellises

Utilizing trellises within DGMs means the state space for accessing all the data instances is much smaller compared with accessing each instance separately. Intuitively, consider a 'simple trellis' that contains $|\acute{Y}|$ disjoint paths from n_s to n_t, where each path corresponds to one data instance (e.g. the 'simple trellis' in Fig. 1D). The state space for the simple trellis is no different than accessing each data instance separately. By constructing the compact trellis, redundant structures in the simple trellis get merged into shared links so that the state space is greatly reduced [step (5) in Fig. 1D]. Depending on the data, the state space reduction can be quite significant. In general, as datasets get larger, we expect more shared structures since there is more tendency for redundancy. Hence, the size of a trellis will grow sublinearly with the size of the input data. For the task of peptide identification in mass spectrometry, there are often thousands of candidate peptides within a certain mass window for one spectrum having the potential of being compressed. Moreover, trellises can be even more effective when post-translational modifications or sequence variants are considered (which otherwise greatly increase the number of separate peptide candidates).

Along with trellises, approximate inference algorithms are effective at significantly reducing the state space of DGMs, decreasing runtime but without keeping the most probable sequence from being inferred. One such method, ideally suited for trellis inference, is *k-beam* pruning (histogram pruning in Ney *et al.* (1994), a heuristic where only the k most probable hypotheses (or states) at each time frame are retained and all other hypotheses are pruned away (i.e. no longer modeled). While k-beam pruning may be used in DGMs without trellises, utilizing k-beam pruning with trellises is significantly more effective since we are pruning hypotheses from the joint representation shared by all sequences. For example, the hypotheses of the original DRIP model only consists of a single peptide's alignments between an observed spectrum so that, when using beam pruning, poor alignments are pruned away early on. With trellises, beam pruning induces a competition among all peptide hypotheses, where peptide candidates which align poorly with the observed spectrum are pruned away early, so we end up scoring only a subset of the candidates. Note that, while we use k-beam pruning in this work, other beam pruning methods or forms of approximate inference may also be sped up using trellises, because trellises only alter the representation of the underlying sequential hypothesis space.

5.1.1 Optimal pruning bounds for Viterbi decoding in trellises

Here we prove under generally applicable assumptions that, in expectation, a small beam width may be used with impunity when jointly modeling multiple sequences (in our case, theoretical spectra) in a single trellis, compared to performing approximate inference with beam pruning independently on each sequence. This pruning results in a substantial computational reduction while ensuring that we have not pruned the most probable hypothesis.

Define a *hypothesis* $h = h_1, \ldots, h_t = h_{1:t}$ to be a sequence of instantiated random variables for frames $1 \ldots, t$ in a DGM. For frame t', let $h^*_{t'}$ be the most probable hypothesis we are trying to infer, m be the number sequences represented in the trellis, $n_{t'}$ be the number of hypotheses with higher probability than $h^*_{t'}$, and $r_{t'}$ be the total number hypotheses in the trellis without pruning at frame t'.

Theorem 1: Suppose m and $n_{t'}$ are large, $r_{t'} \gg n_{t'}$, and the $n_{t'}$ hypotheses with higher probability than $h^*_{t'}$ are uniformly distributed from the m sequences in the trellis. In expectation, performing inference on each sequence independently requires $\Omega(m\sqrt{n_{t'}/m})$ more beam width to ensure h^* is not pruned compared to inference in a trellis.

The proof of Theorem 1 is provided in Section 1 of the Supplementary Appendix. Though a uniform distribution is assumed over the hypotheses, such a distribution is biased in practice for many applications, and the trellis may be even more efficient, because we may more aggressively prune using the k-beam strategy while still preserving the top hypothesis. This theoretically quantifies the expected speedup using trellises to score MS/MS candidate peptides as opposed to scoring candidate peptides individually (in Section 7.1, we demonstrate this speedup empirically).

6 Training DRIP with trellises

In Halloran *et al.* (2014), generatively training the DRIP model's Gaussian parameters was shown to significantly increase performance. Here we extend this framework to discriminative training, using trellises to make such training tractable. For the overall training procedure, assume that we have a collection, C, of N i.i.d. pairs (x^i, y^i), where x^i is an observed spectrum and y^i the corresponding peptide we have strong evidence to believe generated x^i. Let θ be the set of parameters for which we would like to learn (in our case, DRIP's Gaussian parameters). For generative training, we then wish to find $\theta^* = \text{argmax}_\theta \sum_{i=1}^{N} \Pr(x^i|y^i, \theta)$, i.e. we wish to maximize DRIP's likelihood with respect to the parameters to be learned, achieved via the expectation–maximization algorithm (Dempster *et al.*, 1977).

A much more difficult training paradigm is that of discriminative training, where we not only wish to maximize the likelihood of a set of parameters, but would also like to simultaneously minimize a parameterized distribution defined over a set of alternative hypotheses. In our case, this alternative set consists of all candidate peptides within the precursor mass tolerance not equal to y^i, i.e. all incorrect explanations of x^i. More formally, our discriminative training criterion is that of MMI estimation (Povey, 2003). Defining the set of candidate peptides for x^i within precursor mass tolerance w as $\mathcal{C}^i = \mathbf{D}(\mathcal{M}^x, \mathcal{C}^x, \mathcal{D}, w)$ and the set of all training spectra and high-confidence PSMs as $\mathcal{X}$ and $\mathcal{Y}$, respectively, the MMI objective function we maximize with respect to θ is

$$\tilde{I}_\theta(\mathcal{X}; \mathcal{Y}) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{\Pr(x^i|y^i, \theta)}{\sum_{x \in \mathcal{C}^i} \Pr(x^i, x|\theta)}$$

$$= \frac{1}{N}\sum_{i=1}^{N} (\log \Pr(x^i|y^i, \theta) - \log \sum_{y \in \mathcal{C}^i} \Pr(x^i, y|\theta)), \qquad (5)$$

where we call $M_n(x^i, y^i, \theta) = \log \Pr(x^i | y^i, \theta)$ the *numerator model* and $M_d(y^i, \theta) = \log \sum_{y \in \mathcal{C}^i} \Pr(x^i, y | \theta)$ the *denominator model*. Note that the numerator model is our objective function for generative training.

Intuitively, Equation (5) is maximized by learning parameters which increase the numerator model (what is done in generative training) and/or decrease the denominator model. Thus, while generative training only learns parameters based on the high-confidence PSMs in the numerator, discriminative training learns parameters which also discourage the ensemble of PSMs in the denominator model (which are incorrect matches). We solve maximize Equation (5) with respect to θ using stochastic gradient ascent.

In stochastic gradient ascent, we calculate the gradient of the objective function with regards to a single training instance, $\nabla_\theta \tilde{I}_\theta(x^i; y^i) = \nabla_\theta M_n(x^i, y^i, \theta) - \nabla_\theta M_d(x^i, \theta)$, where the gradients of M_n and M_d are vectors typically referred to as *Fisher scores*. We update the parameters θ using the previous parameters plus a damped version of the objective function's gradient, iterating this process until convergence. In practice, we begin the algorithm by initializing θ_0 to a good initial value, i.e. the output of generative training, and the learning rate η_j is updated with $\eta_{j+1} = (\sqrt{j})^{-1}$. Intuitively, the gradients move in the direction maximizing the difference between the numerator and denominator models, encouraging improvement for the numerator while discriminating against the incorrect labels in the denominator.

Discriminative training is computationally expensive. The denominator model requires calculating the gradients of all candidate peptides $\mathcal{C}^i$, which can be infeasible for many tasks. A further challenge in DRIP's case is that it is difficult to constrain the model to consider valid peptides only, because the distance between subsequent theoretical peaks can take on any value. Trellises address both of these challenges. The denominator model of the discriminative training for DRIP is exactly the same as the DRIP trellis model (Section 5). Furthermore, the trellis of all possible labels can be very compact; together with different strategies to speed up graphical models with trellises discussed in the previous session, discriminative training with trellises is highly efficient. In practice, we use a trellis constructed from a decoy set, which contains permutations of the target peptides as the denominator. As the denominator set grows larger, trellises make the computational cost grow sub-linearly so that discriminative training with trellises is efficient. Our experimental results show that discriminative training positively influences performance (Section 7.2).

7 Results

A significant challenge in evaluating the quality of a spectrum identification algorithm is the absence of a 'ground truth' dataset where the generating peptide is known for each observed spectrum. We, therefore, follow the standard approach of using *decoy peptides* (which in our case correspond to randomly shuffled versions of each real *target* peptide) to estimate the number of incorrect identifications in a given set of identified spectra. In this work, targets and decoys are scored separately and used to estimate the number of identified matches at a particular FDR, i.e. the fraction of spectra improperly identified at a given significance threshold. We estimate FDR using the target-only variant of target-decoy competition, T-TDC (Keich *et al.*, 2015). Because the FDR is not monotonic with respect to the underlying score, we instead use the *q-value*, defined to be the minimum FDR threshold at which a given score is deemed to be significant. Since datasets containing more than 10% incorrect identifications are generally not practically useful, we only plot *q*-values in the range $[0, 0.1]$.

7.1 Faster graphical model identification of MS/MS spectra using trellises

In Section 5.1.1, the expected performance of Viterbi decoding with beam pruning using trellises was proven to be significantly more efficient than considering each sequence independently. We first verified that DRIP's performance, in terms of the number of spectra identified at a fixed FDR threshold, is equivalent with and without use of the trellis (Supplementary Appendix Fig. 1). We then investigate the improved inference speed using trellises. To do so, we randomly select and search 200 spectra each from three datasets (described in Supplementary Appendix Section 2): yeast and worm, both acquired using low-resolution precursor scans (± 3 Th tolerance) and low-resolution fragment ions; and *Plasmodium*, acquired using a high-resolution precursor scan (± 50 ppm tolerance) and high-resolution fragment ions. Further details regarding these datasets and search settings may be found in Section 2 of Supplementary Appendix.

For all methods, we use the same graphical model inference engine, GMTK (Bilmes and Zweig, 2002). Experiments were carried out on a 3.40 GHz CPU with 16G memory. For each experiment, the lowest CPU time out of three runs is recorded, and we report the relative CPU time of methods using trellises to those without. For the XCorr mixture model (Section 4), a fixed *k*-beam for all frames was used. Per spectrum, the trellis-inferred top PSM scores were exactly the same as computing each XCorr individually and determining the top PSM.

We test DRIP trellis with two beam pruning strategies (some discussion of beam pruning with trellises can be found in Section 5.1), and we compare the results against DRIP using the beam pruning settings of Halloran *et al.* (2014). The trellis$_{\text{base}}$ pruning uses *k*-beams that are dynamic across time frames, with wider beams for the early part and narrower beams later on. The trellis$_{\text{speed}}$ pruning also uses a dynamic *k*-beam but with a narrower beam, followed by another pruning strategy that removes all hypotheses whose score falls below some fraction of the currently top scoring hypothesis while building up the inference structures. Timing tests show that DRIP runs 7–15 times faster using trellises versus without trellises (Fig. 5A).

The absolute timing numbers of all the searches implemented using the graphical model engine are relatively high. XCorr (without a trellis) takes $\sim$2 s per spectrum to search the low-resolution datasets (yeast and worm) and $\sim$0.5 s per spectrum to search the high-resolution *Plasmodium* dataset. Without a trellis, the more complex DRIP model takes $\sim$10 s per spectrum searching the low-resolution datasets and $\sim$2 s searching the high-resolution dataset. Constant factor improvements for these models may be accomplished by optimized C++ implementations. For instance, a highly optimized implementation of XCorr (McIlwain *et al.*, 2014) takes 0.024 s per spectrum searching the high-resolution *Plasmodium* dataset (minimum time over three runs) with the same compute environment and search settings used in our timing tests. Thus, we expect that an optimized trellis XCorr implementation would search the same spectra (under the same settings and environment) in roughly 0.0024 s per spectrum.

We note that trellises can speed up many scoring algorithms simply by changing the preprocessing of data; the trellis approach is agnostic to the underlying scoring algorithm and, therefore, compatible with any method that makes the underlying scoring

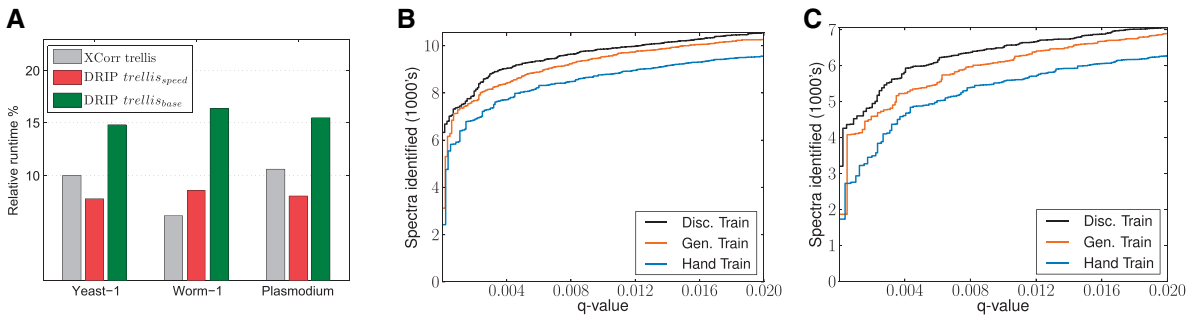


Fig. 5. (A) Percent running time of XCorr trellis, DRIP $trellis_{base}$, and DRIP $trellis_{speed}$ relative to the running time of the original model without a trellis. **(B)** Discriminative training improves performance for the worm dataset. **(C)** Similar to (B), but for the yeast dataset.

algorithm more efficient. We also note that our use of a prototyping language (GMTK) to express the above methods, while slower than highly optimized and specialized C implementation, allows the exploration of vastly different models (linear XCorr and non-linear DRIP) without needing to re-implement each model from scratch.

7.2 More accurate graphical model identification of MS/MS spectra using trellises

As detailed in Section 6, we use a set of high confidence, charge +2 PSMs and their corresponding peptide database to discriminatively train the DRIP model, using trellises in the denominator model. We compare the discriminatively trained DRIP model to generatively trained and hand-tuned DRIP models, as well as the methods described in Section 2. The discriminatively trained DRIP model identifies more spectra at a 1% FDR threshold relative to the generatively trained and hand-tuned models (Fig. 5). Note that the discriminatively trained model employs the $trellis_{base}$ pruning strategy; thus, the model yields an increase in accuracy as well as an approximately 7-fold speedup. Comparisons against other search engines (MS-GF+, XCorr, XCorr *P*-value and X!Tandem) may be found in Supplementary Appendix Section 3. We further note that, while MS/MS scoring methods that afford efficient parameter learning are scarce, any DBN may be discriminatively trained using trellises in the same fashion as DRIP. Because several widely used MS/MS algorithms may be expressed as DBNs (shown in Section 5), they may also adapt this overall training procedure to their benefit.

8 Discussion

We have proven that, for many practical settings, the expected runtime when using a trellis with beam pruning for Viterbi decoding is significantly faster than considering sequences independently. We have also empirically shown that trellises may be used to significantly improve the speed and accuracy of peptide identification in tandem mass spectrometry. Using the DRIP model and a DBN implementation of XCorr, we have shown how to apply trellises to dramatically speed up inference (6- to 15-fold), both for low-resolution and high-resolution precursor mass spectra. We further note that the algorithmic speedup afforded by using trellises may, in future work, be combined with previous work on improving XCorr runtime, which focused on constant-factor improvements (Diament and Noble, 2011). We also note that trellises constructed from peptides with variable modifications can be potentially more efficient as variable modifications produce a lot redundancy among peptides.

Our MS/MS trellises support traversing all subsequences of a data instance, allowing 'jumps' over whole subsequences, a novel feature in contrast to traditional trellises (such as those used for speech recognition), which only allow data instances to be sequentially traversed. As in DRIP, where jumps correspond to deletions, this feature may be used to model noise or missing data. Furthermore, the jump feature enriches the hypothesis space representation, allowing more sophisticated models to be expressed and evaluated efficiently. With this feature and the ability to compactly represent entire sets of peptides, we have extended DRIP's learning framework to discriminative training, significantly improving its performance relative to previous training strategies.

In future work, we plan to further explore using trellises for improved MS/MS identification. Using trellises to efficiently evaluate and score peptides beyond those in the given database, we will investigate ways to take thresholds with respect to DBN-based scoring functions to compute *P*-values, similar to the *P*-value calculations done via dynamic programming by MS-GF+ and XCorr *P*-value. By improving score calibration, we expect this approach to greatly improve DRIP's performance.

Funding

This work was supported by National Institutes of Health awards R01GM096306 and P41GM103533.

References

Bilmes,J. (2004) On virtual evidence and soft evidence in Bayesian networks. *Technical report UWEETR-2004-0016*. Electrical Engineering, University of Washington.

Bilmes,J. (2010) Dynamic graphical models. *IEEE Signal Proc. Mag.*, **27**, 29–42.

Bilmes,J. and Zweig,G. (2002) The Graphical Models Toolkit: an open source software system for speech and time-series processing. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Orlando, US.

Craig,R. and Beavis,R.C. (2004) Tandem: matching proteins with tandem mass spectra. *Bioinformatics*, **20**, 1466–1467.

Dempster,A.P. *et al.* (1977) Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. B*, **39**, 1–22.

Diament,B. and Noble,W.S. (2011) Faster sequest searching for peptide identification from tandem mass spectra. *J. Proteome Res.*, **10**, 3871–3879.

Dyer,C. *et al.* (2008) Generalizing word lattice translation. *Technical report*, DTIC Document, LAMP-TR-149.

Eng,J.K. *et al.* (1994) An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *J. Am. Soc. Mass Spectrom.*, **5**, 976–989.

Huang E.-F. and Soong, F.K. (1991) A tree-trellis based fast search for finding the n-best sentence hypotheses in continuous speech recognition. In: *International Conference on Acoustics, Speech, and Signal Processing*, Toronto, Canada, IEEE, Vol. 1, pp. 705–708.

Geer,L.Y. *et al.* (2004) Open mass spectrometry search algorithm. *J. Proteome Res.*, **3**, 958–964.

Halloran,J.T. *et al.* (2014) Learning peptide-spectrum alignment models for tandem mass spectrometry. In: *Uncertainty in Artificial Intelligence (UAI)*, Quebic City, Quebec Canada. AUAI.

Hopcroft,H. (1971) An n log n algorithm for minimizing states in a finite automaton. *Technical report*, DTIC Document, STAN-CS-71-190.

Hopcroft,J.E. *et al.* (2001) Introduction to automata theory, languages, and computation. *ACM SIGACT News*, **32**, 60–65.

Jelinek,F. (1997) *Statistical Methods for Speech Recognition*. Cambridge, MA, US, MIT Press.

Ji,G. *et al.* (2006) Graphical model representations of word lattices. In: IEEE/ACL Workshop on Spoken Language Technology (SLT), Palm Beach, Aruba.

Keich,U. *et al.* (2015) Improved false discovery rate estimation procedure for shotgun proteomics. *J. Proteome Res.*, **14**, 3148–3161.

Kim,S. and Pevzner,P.A. (2014) Ms-gf+ makes progress towards a universal database search tool for proteomics. *Nat. Commun.*, **5**, 5277.

McIlwain,S. *et al.* (2014) Crux: rapid open source protein tandem mass spectrometry analysis. *J. Proteome Res.*, **13**, 4488–4491.

Nesvizhskii,A.I. (2010) A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics. *J. Proteom.*, **73**, 2092–2123.

Ney,H. *et al.* (1994) Improvments in beam search. In: *Proceedings the International Conference on Spoken Language Processing*, Yokohama, Japan.

Pearl,J. (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, USA.

Povey,D. (2003) *Discriminative training for large vocabulary speech recognition*. PhD Thesis, Cambridge, UK, University of Cambridge.

Schnitger,G. and Gramlich,G.(2005) Minimizing NFA's and regular expressions. In: 22nd International Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, Berlin, Springer, Vol. 3404, pp. 399–411.

Walters,M.C. *et al.* (1996) Bone marrow transplantation for sickle cell disease. *New Engl. J. Med.*, **335**, 369–376.

Watson,B.W. (1993) A taxonomy of finite automata minimization algorithms. *Comput. Sci. Note*, **44**.

Wenger,C.D. and Coon,J.J. (2013) A proteomics search algorithm specifically designed for high-resolution tandem mass spectra. *J. Proteome Res.*, **12**, 1377–1386.

Young,S. *et al.* (1997) *The HTK Book, 2.1 edn*. Cambridge, UK, Entropic Labs and Cambridge University.