Sequence Database Compression for Peptide Identification from Tandem Mass Spectra

Nathan Edwards* and Ross Lippert**

Informatics Research, Advanced Research and Technology, Applied Biosystems
45 W. Gude Drive, Rockville MD, 20850
{EdwardNJ,LipperRA}@AppliedBiosystems.com

Abstract. The identification of peptides from tandem mass spectra is an important part of many high-throughput proteomics pipelines. In the high-throughput setting, the spectra are typically identified using software that matches tandem mass spectra with putative peptides from amino-acid sequence databases. The effectiveness of these search engines depends heavily on the completeness of the amino-acid sequence database used, but suitably complete amino-acid sequence databases are large, and the sequence database search engines typically have search times that are proportional to the size of the sequence database.

We demonstrate that the peptide content of an amino-acid sequence database can be represented by a reformulated amino-acid sequence database containing fewer amino-acid symbols than the original. In some cases, where the original amino-acid sequence database contains many redundant peptides, we have been able to reduce the size of the amino-acid sequence to almost half of its original size. We develop a lower bound for achievable compression and demonstrate empirically that regardless of the peptide redundancy of the original amino-acid sequence database, we can compress the sequence to within 15-25% of this lower bound. We believe this may provide a principled way to combine amino-acid sequence data from many sources without unduly bloating the resulting sequence database with redundant peptide sequences.

1 Introduction

The identification of peptides from tandem mass spectra is an important part of many high-throughput proteomics pipelines. In the high-throughput setting, the spectra are typically identified using software that matches tandem mass spectra with putative peptides from amino-acid sequence databases. The effectiveness of these search engines depends heavily on the completeness of the amino-acid sequence database used, but suitably complete amino-acid sequence databases are large, and the search engines typically have search times that are proportional

* Corresponding Author.
** Current address: Department of Mathematics, Massachusetts Institute of Technology. Email: lippert@math.mit.edu

I. Jonassen and J. Kim (Eds.): WABI 2004, LNBI 3240, pp. 230–241, 2004.

to the size of the sequence database. See [1, 2] for an extensive discussion of the running time cost of these search engines.

An inherent weakness of sequence database search engines such as SEQUEST [3], Mascot [4], and SCOPE [5], is that they struggle to identify the tandem mass spectra of peptides whose sequence is missing from the sequence database. The obvious solution, then, is to construct amino-acid sequence databases containing more of the peptide sequences that we might need in order to identify tandem mass spectra. For example, the sequences of all protein isoforms listed as variant annotations in Swiss-Prot [6, 7] records can be enumerated using the program `varsplic` [8, 9]. The resulting amino-acid sequence database is more than 1.5 times the size of Swiss-Prot, but at most, it contains about 2% additional peptides candidates. Since sequence database search engines typically have search times proportional to the size of the input sequence database, this increase in search time is a significant cost. Similarly, when we form the union of a number of sequence databases in order to create a comprehensive search database, merely ensuring non-redundancy at the protein sequence level leaves significant peptide level redundancy. In this paper, we strive to rewrite amino-acid sequence databases in such a way that most of the peptide redundancy is eliminated without losing any peptide content.

Current mass spectrometers are capable of reliably acquiring tandem mass spectra from ions with mass of up to about 3000 Daltons. However, the tandem mass spectra from peptides with more than 20 amino-acids or charge state 4 or more are rarely successfully interpreted by current sequence database search engines. Further, most peptide identification workflows digest proteins with trypsin, which cuts at either lysine or arginine (unless followed by proline). These amino-acids occur with sufficient frequency that peptides longer than 25 amino-acids are rare. Bearing all this in mind, we can conservatively upper-bound the length of peptides that sequence database search engines need to consider at about 30 amino-acids. We adopt the terminology of DNA sequence analysis and refer to a sequence of k amino-acids as a k-mer. The peptide content of an amino-acid sequence database is therefore represented by the set of 30-mers it contains.

We will refer to a number of commonly used amino-acid sequence databases to demonstrate our approach. We will refer to the Swiss-Prot section of the UniProt Knowledgebase [6, 7] as Swiss-Prot. The union of the Swiss-Prot, TrEMBL, and TrEMBL-New sections of the UniProt Knowledgebase will be referred to as UniProt. Many Swiss-Prot and UniProt entries contain protein isoform annotations, but only a single sequence is provided per entry. To construct a sequence database containing all of these isoforms, we use `varsplic` [8, 9] with the command line options

```
-which full -uniqids -fasta -varsplic -variant -conflict
```

to enumerate all original sequences plus all splice forms, variants and conflicts. These sequence databases will be referred to as Swiss-Prot-VS and UniProt-VS respectively. MSDB [10] (Mass Spectrometry protein sequence DataBase)

Table 1. Sequence statistics of some sequence databases used for peptide identification via tandem mass spectrometry.

Sequence Database	Sequence Length	Distinct 30-mers	Overhead
IPI-HUMAN	20358846	12115520	68%
IPI	54145883	29769766	81%
Swiss-Prot	56454588	44374286	27%
Swiss-Prot-VS	89541275	45307827	97%
UniProt	472581860	274510105	72%
UniProt-VS	506796094	275391669	84%
MSDB	481919777	276523755	74%
NRP	495502241	283160529	75%
NCBI-nr	619132252	378721915	63%
UnionNR	674700840	385369671	75%
Union	2157353500	385369671	460%

is a composite non-identical protein sequence database built from a number of primary source databases. MSDB is often used in conjunction with the Mascot sequence database search engine for protein identification. NCBI-nr [11] is a composite, non-redundant protein database from NCBI constructed from various sources. NRP [12] is a composite, non-redundant protein sequence database from the Advanced Biomedical Computing Center at NCI in Frederick, MD. The international protein index [13] (IPI) sequence databases, from EBI, are human, mouse and rat sections from Swiss-Prot, TrEMBL, RefSeq and Ensembl. We denote the human protein index by IPI-HUMAN, and the union of the three pieces as IPI. Finally, we form the concatenation of each of these comprehensive sequence databases to form a new comprehensive sequence database, Union, comprising UniProt-VS, MSDB, NCBI-nr, NRP, and IPI. For completeness, we also do the standard exact protein sequence redundancy elimination for Union, since its constituent sequence databases contain common protein entries. The protein level non-redundant version of Union is called UnionNR. Table 1 shows the size, or sequence length, of these sequence databases, as well as the number of distinct 30-mers they contain.

Suppose that we could rewrite each of these amino-acid sequence databases as a new amino-acid sequence database that is:

Complete
Every 30-mer from the original sequence database is present,
Correct
Only 30-mers from the original sequence database are present, and
Compact
No 30-mer is present more than once.

Suppose further that such a complete, correct and compact sequence database consists of a single sequence. Since the sequence is complete, a sliding 30-mer

window on this sequence will generate all the 30-mers of our original sequence database. Therefore, this sequence has at least as many amino-acids as our original sequence database has distinct 30-mers. Hence the number of distinct 30-mers of our original sequence database is a lower bound on the size of any complete sequence database. The column **Overhead** of Table 1 shows the amount of additional sequence contained in various sequence databases over and above this lower bound. We will demonstrate how to construct complete, correct, compact sequence databases that come as close as possible to this lower bound.

We note that the well known shortest (common) superstring (SCS) problem [14, 15] cannot be applied as it does not guarantee its output to be correct or compact. Further, since the input sequences must be represented intact, the SCS cannot take advantage of redundancy in the interior of the protein sequences.

2 Sequence Databases and SBH-Graphs

In order to compress the amino-acid sequences of our input sequence database, we must first build a representation of its 30-mers.

Sequencing-by-Hybridization, proposed by Bains and Smith [16], Lysov *et. al.* [17], and Drmanac *et. al.* [18], is a technique by which DNA is interrogated by hybridization to determine the presence or absence of all possible length k DNA sequences. The information from these experiments can be represented in a graph, which we call the SBH-graph, first proposed by Pevzner [19]. The graph contains a directed edge for every observed k-mer probe, from a node representing the first $(k-1)$-mer of the probe to a node representing the last $(k-1)$-mer of the probe. Determining the original DNA sequence is then a matter of finding a path through the SBH-graph that uses every edge, representing an observed k-mer, at least once. See Figure 1 for a small SBH-graph example. As shown in Figure 1, each node has its $(k-1)$-mer sequence associated with it, while each edge holds the nucleotide that is appended to the sequence at the tail of the edge to form the k-mer it represents. We will use a SBH-graph to represent all the k-mers of our input sequence database.

We point out the connection here to de Bruijn graphs [20]. The SBH-graph is really just a subgraph of a de Bruijn digraph. A de Bruijn graph has edges for all k-mers from the alphabet, and hence nodes for all $(k-1)$-mers from the alphabet. A de Bruijn sequence is a circular sequence from the alphabet that represents all k-mers of the alphabet exactly once. A de Bruijn sequence can be enumerated by constructing an Eulerian tour on the corresponding de Bruijn graph, which happens to be an Eulerian graph. We call our de Bruijn subgraphs SBH-graphs to emphasize that we are representing some subset of all the k-mers, those k-mers found in the input sequence database. Despite this distinction, the de Bruijn sequence represents the motivation for the results to follow.

Given G, the SBH-graph representation of the k-mers of the sequence database S, we can identify the sequences of S with paths of G. In fact, any sequence s containing only k-mers of S can be represented by a path on G. Given s, we first locate the node representing the first $(k-1)$-mer of s. The edge

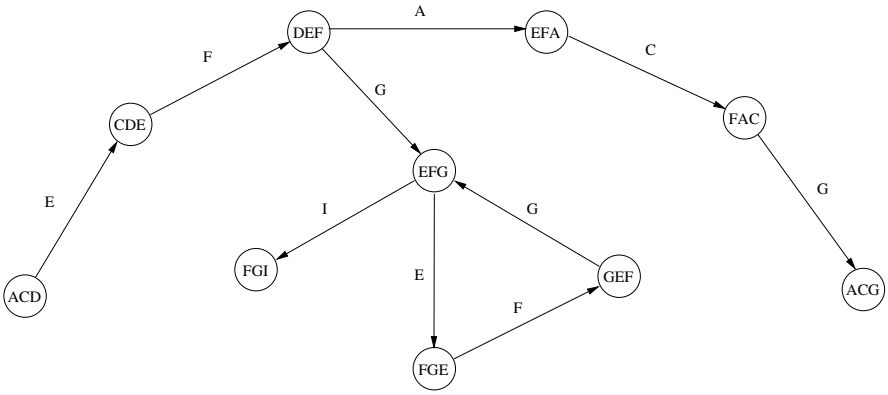

Fig. 1. SBH-graph for 4-mers from the sequences: ACDEFGI, ACDEFACG, DEFGEFGI.

representing the first k-mer of s must be present since s contains only k-mers from S. We follow this edge, and look for an edge representing the next k-mer of s. The series of edges traced by this procedure corresponds to a path on G. Conversely, we construct the sequence represented by some path by outputting the $(k-1)$-mer of the initial node of the path and then the sequence on each edge of the path.

With this understanding of the equivalence of sequences on the k-mers of S and paths on the SBH-graph, we must now determine what the complete, correct, compact constraints imply for paths on our SBH-graph. Since each edge of the SBH-graph represents a distinct k-mer from our input sequence database, we obtain a complete, correct, compact sequence database by finding a path set that uses each edge exactly once and by generating the sequences represented by these paths.

Next, we must quantify the size of our new amino-acid sequence database. Let $S' = \{s'_1, \ldots, s'_l\}$ be our complete, correct, compact sequence database, and let N_k be the number of distinct k-mers in our original sequence database. Since S' is complete, correct, and compact, we know that all k-mers are observed, no extra k-mers are observed, and that each k-mer window generates a distinct k-mer. Therefore

$$N_k = \sum_{i=1}^{l} \text{windows}(s'_i) = \sum_{i=1}^{l} \text{length}(s'_i) - (k-1)$$

$$\implies \qquad l(k-1) + N_k = \sum_{i=1}^{l} \text{length}(s'_i)$$

Consequently, the size of S' is

$$|S'| = \text{length}(s'_1) + 1 + \cdots + 1 + \text{length}(s'_l) = N_k + lk - 1$$

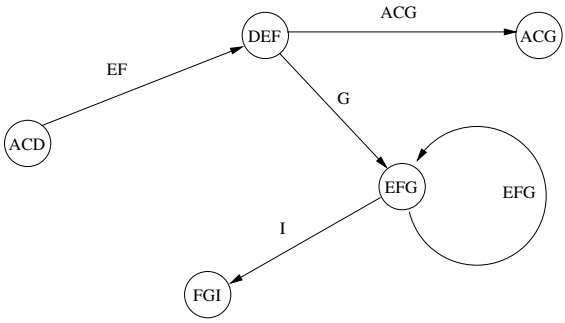

Fig. 2. Compressed SBH-graph for 4-mers from the sequences: ACDEFGI, ACDEFACG, DEFGEFGI.

The extra character we output with each sequence is an end-of-sequence marker, necessary to ensure we don't consider k-mers that straddle the end of one sequence and the start of the next. We can, of course, suppress the last end-of-sequence marker.

The important thing to notice here is that the only parameter we are free to change in order to reduce the size of S' is the number of sequences it contains. Therefore, constructing a minimum size complete, correct, compact sequence database is equivalent to minimizing the number of paths in a SBH-graph path set such that each edge is used exactly once.

The best possible scenario, then, is that our SBH-graph admits an Eulerian path. In this case, one path is sufficient. Note that this lemma effectively restates our lower bound on the minimum size of any complete sequence database.

Lemma 1. *The set of k-mers of S can be represented by a complete, correct, and compact sequence database of size $(k-1)+N_k$ if and only if the SBH-graph of S admits an Eulerian path.*

In this work, we will use a *compressed* SBH-graph to represent the k-mers of our sequence database. The compressed SBH-graph (CSBH-graph) represents the same information as the SBH-graph, except that paths through trivial nodes, those with in and out degree one, are turned into a single edge. Where an edge of the CSBH-graph replaces a path through trivial nodes, we associate the concatenation of the character on the edges of the path with the new edge. Figure 2 shows the CSBH-graph corresponding to the SBH-graph of Figure 1. It should be clear that all of the preceding discussion, including Lemma 1, holds just as well for CSBH-graphs as for SBH-graphs.

Lemma 2. *The set of k-mers of S can be represented by a complete, correct, and compact sequence database of size $(k-1)+N_k$ if and only if the CSBH-graph of S admits an Eulerian path.*

We use the CSBH-graph instead of the SBH-graph because it is much smaller, with many fewer nodes and edges. However, building the CSBH-graph for a

Table 2. SBH-graph and CSBH-graph sizes for some sequence databases used for peptide identification via tandem mass spectrometry.

Sequence Database	SBH-graph Nodes	SBH-graph Edges	CSBH-graph Nodes	CSBH-graph Edges
IPI-HUMAN	12119290	12115520	115246	111476
IPI	29645471	29769766	545356	669651
Swiss-Prot	44352317	44374286	550060	572029
Swiss-Prot-VS	45259553	45307827	609679	657953
UniProt	274510105	274510105	4075920	4445958
UniProt-VS	274995795	275391669	4132200	4528074
MSDB	276094660	276523755	4313501	4742596
NRP	282706577	283160529	4443077	4897029
NCBI-nr	384256196	378721915	5534281	5978394
Union(NR)	384866007	385369671	5758507	6262171

given amino-acid sequence database is non-trivial, particularly for $k = 30$ and an alphabet of size 20. Clearly we could first build the SBH-graph and remove trivial nodes, but the initial cost to build the SBH-graph quickly becomes prohibitive. We avoid these issues by constructing the CSBH-graph directly, using a suffix-tree data-structure. Table 2 shows the sizes of the SBH and CSBH-graphs for each of the sequence databases of Table 1.

3 Optimal Complete, Correct, Compact (C^3) Enumeration

For a directed graph $G = (V, E)$, we define $\delta_i(v)$ and $\delta_o(v)$ to be the in and out degree of node $v \in V$. Further, we define $b(v) = \delta_i(v) - \delta_o(v)$. $b(v)$ is called the degree deficit if $b(v) < 0$ and the degree surplus if $b(v) > 0$. A node v with $b(v) = 0$ is called balanced, otherwise it is called unbalanced. A graph or connected component is called balanced if all of its nodes are balanced and unbalanced otherwise. We define V_+ to be the set of surplus degree nodes and n_+ to be number of these nodes. Similarly, we define V_- to be the set of deficit degree nodes and n_- to be the number of these nodes. The total degree surplus is defined to be $B_+ = \sum_{v \in V_+} b(v)$ while the total degree deficit is defined as $B_- = \sum_{v \in V_-} b(v)$. The degree surplus of a connected component $C \subseteq V$ is defined to be $B_+(C) = \sum_{v \in V_+ \cap C} b(v)$, with the $B_-(C)$ defined analogously.

In practice, CSBH-graphs built from amino-acid sequence databases with $k = 30$ fail to be Eulerian on two counts. First, very few nodes of the graph are balanced, and second, the graphs usually have more than one connected component. Table 3 shows the extent to which the CSBH-graphs of 30-mers built for our test set of amino-acid sequence databases fail to be Eulerian.

We must, of course, have at least one path in our path set per component. For each balanced component, we require exactly one path, an Eulerian tour, as

Table 3. CSBH-graph statistics of some sequence databases used for peptide identification via tandem mass spectrometry.

Sequence Database	Degree Surplus Nodes	Total Surplus Degree	Degree Deficit Nodes	Total Deficit Degree	Components (Balanced)
IPI-HUMAN	57275	57971	56975	57971	23076 (1)
IPI	267896	273052	267329	273052	35728 (2)
Swiss-Prot	270279	276262	270228	276262	93611 (0)
Swiss-Prot-VS	299410	307551	299154	307551	93624 (0)
UniProt	1992448	2086977	1988855	2086977	626503 (5)
UniProt-VS	2019828	2116632	2015947	2116632	626470 (5)
MSDB	2112761	2213341	2101795	2213341	629636 (6)
NRP	2175883	2281329	2164551	2281329	643496 (6)
NCBI-nr	2712544	2826497	2701270	2826497	850325 (7)
Union(NR)	2822070	2943180	2810160	2943180	863078 (8)

per Lemma 2. We denote the number of balanced components by m. What then, for unbalanced components? Lemma 3 prescribes a lower bound on the number of paths required.

Lemma 3. *If C is an unbalanced component of a CSBH-graph, then we require at least $B_+(C)$ paths, in order to use each edge exactly once.*

Proof. Given a path set, suppose we consider each path, in turn, and delete its edges from C. The deletion of the edges of a path from s to t increases $b(s)$ by one, decreases $b(t)$ by 1, and leaves the remaining $b(v), v \neq s, t$ unchanged. Since the path set uses every edge exactly once, when the process is complete we must have $b(v) = 0$ for all nodes $v \in V$. Therefore, there must have been $-b(v)$ paths starting at nodes $v \in V_- \cap C$ and $b(v)$ paths ending at nodes $v \in V_+ \cap C$. Therefore, we require at least $\max\{B_+(C), -B_-(C)\}$ paths in the path set, each of which starts at a node of $V_- \cap C$ and ends at a node of $V_+ \cap C$. By induction on the edges of C, it is straightforward to show that $B_+(C) = -B_-(C)$. □

What we have shown then is that our path set must contain at least $B_+ + m$ paths. All that remains is to demonstrate how this bound can be achieved.

Lemma 4. *Given an unbalanced component C of a CSBH-graph G, there exists a path set of size $B_+(C)$ that uses each edge exactly once.*

Proof. We add $B_+(C) - 1$ artificial *restart* edges from nodes of $V_+ \cap C$ to nodes of $V_- \cap C$. The edges are added in such a way that $v \in V_+ \cap C$ has at most $b(v)$ outgoing restart edges, while $v \in V_- \cap C$ has at most $-b(v)$ incoming restart edges. The resulting CSBH-graph must contain exactly two unbalanced nodes, s with $b(s) = -1$ and t with $b(t) = 1$. Therefore, we can construct an Eulerian path between s and t that uses every edge exactly once. This Eulerian path can then broken into $B_+(C)$ paths at the restart edges and the lemma is shown. □

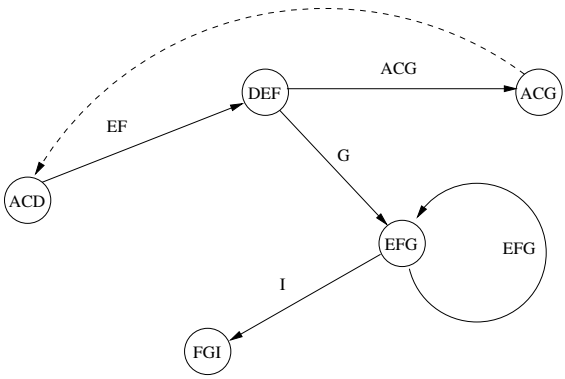

Fig. 3. Compressed SBH-graph with artificial edge (dashed) for 4-mers from the sequences: ACDEFGI, ACDEFACG, DEFGEFGI.

Thus, we have demonstrated how to construct a complete, correct, compact sequence database of minimum size.

Theorem 1. *Let G be the CSBH-graph of an input sequence S for some mer size k with m balanced components and total degree surplus B_+. Then the optimal complete, compact, correct sequence database with respect to S and k has size*

$$N_k + k(m + B_+) - 1.$$

Figure 3 shows the CSBH-graph of Figure 2 with its required $B_+ - 1 = 2 - 1 = 1$ artificial restart edge(s) inserted. The Eulerian path of this graph leads to the minimum length 4-mer enumeration: DEFACG, ACDEFGEFGI; which uses 17 characters, instead of the original 25 characters, to represent 10 distinct 4-mers. The lower bound implied by Lemma 2 is 13 characters.

4 Computational Experiments

Table 4 shows the performance of the C^3 30-mer enumeration strategy as a compression technique for the amino-acid sequence databases of Table 1.

We notice that despite the widely varying degree of overhead of the original sequence databases, the overhead of the compressed version is typically in the 15-25% range. For those sequence databases, such as Swiss-Prot, which have little overhead, the compression is not particularly impressive, but for those sequence databases containing lots of peptide level redundancy, such as Swiss-Prot-VS, UniProt-VS, or Union, the compression is significant. Even for sequence databases with moderate overhead, the compression is substantial. This compression technique has reduced each of the sequence databases to approximately the same level of peptide redundancy.

Most encouraging is the performance of the compression on the Swiss-Prot-VS, UniProt-VS and Union sequence databases. These sequence databases are

Table 4. Size of optimal complete, correct, compact 30-mer enumerations for sequence databases used for peptide identification via tandem mass spectrometry.

Sequence Database	C^3 30-mer Enumeration	Overhead	Compression	Compression Bound
IPI-HUMAN	13854679	14.35%	68.05%	59.51%
IPI	37961385	27.52%	70.11%	54.98%
Swiss-Prot	52662145	18.68%	93.28%	78.60%
Swiss-Prot-VS	54534356	20.36%	60.90%	50.60%
UniProt	337119564	22.81%	71.34%	58.09%
UniProt-VS	338890778	23.06%	66.87%	54.34%
MSDB	342924164	24.01%	71.16%	57.38%
NRP	351600578	24.17%	70.96%	57.15%
NCBI-nr	463517034	22.39%	74.87%	61.17%
UnionNR	473665310	22.91%	70.20%	57.12%
Union	473665310	22.91%	21.96%	17.86%

constructed specifically to create richer peptide candidate lists, but they necessarily contain many redundant peptide candidates. As discussed in the introduction, the Swiss-Prot-VS sequence database contains about 2% additional distinct 30-mers, despite being more than 1.5 times the size of Swiss-Prot. After compression, Swiss-Prot-VS is smaller than the original Swiss-Prot sequence database. With no modification of existing search engines, we can search all the isoforms of Swiss-Prot in less time than it would take to search the original Swiss-Prot sequence. The Union sequence database is compressed to about three quarters of the size of one of its constituents, NCBI-nr. Again, without any modification to our existing search engines, we can search all isoforms of UniProt, all of MSDB, all of NRP, all of IPI, and all of NCBI-nr in less time than it takes to search the original NCBI-nr sequence.

5 Conclusion

We have shown that the peptide content of an amino-acid sequence database can be represented in a reformulated amino-acid sequence database containing fewer amino-acid symbols than the original. We have demonstrated how to construct an enumeration of all k-mers of a sequence database that is complete, correct and compact, and shown that this representation is as small as possible. We believe that this technique will be most useful for sequence databases that contain lots of peptide level redundancy, such as those constructed to enumerate protein isoforms or merge sequence databases from different sources. In some cases, this technique makes it possible to search a richer set of peptide candidates using sequence databases that are no bigger than their less expressive counterparts.

There remains much work to be done. We have not addressed a number of issues that must be resolved before we can use such a sequence database with Mascot or SEQUEST. In the process of our k-mer enumeration, our amino-acid sequences lose all explicit connection to their original protein sequence and

annotation. We have effectively decoupled the problem of associating peptide sequences with spectra and the problem of determining which proteins the peptides represent. Fortunately, searching for exact peptide sequences in large sequence databases can be done very quickly using well established string matching techniques, particularly since we only need to do this for 10-20 peptides per spectrum. The key piece of infrastructure required to accomplish this is a tool that can take a Mascot or SEQUEST output file and a sequence database as input, and output a new Mascot or SEQUEST output file with the protein information inserted appropriately.

For some use cases, our requirement that we represent all 30-mers is too strong. In particular, for identification of tandem mass spectra, we often search only for those peptides that have trypsin digest motifs at each end. A complete, correct, compact tryptic peptide enumeration would be a natural specialization of this work. However, in order to implement this, we would need either to be able to build a CSBH-graph like data-structure for variable length mers, or to consider tryptic fragment "characters" as input to the problem.

For those use cases, such as peptide identification, in which 30-mers are merely an upper bound on the length of relevant sequences, we note that there is no particular advantage in insisting that all 30-mers are distinct, since this guarantees nothing about the number of occurrences of shorter sequences. In fact, we believe that it should be possible to compress our input sequences databases still further by relaxing the compactness constraint, permitting k-mers to appear more than once in the output. Instead of adding restart edges, we may, perhaps, reuse edges to get from the end of one path to the start of another. Given the similarity of this variant of the problem to the well known "Chinese Postman Problem" [21], we believe that this variant may also be solved to optimality in polynomial time using some sort of matching formulation.

A less obvious application of this k-mer enumeration is in suffix tree compression. For applications such as the peptide candidate generation [1], in which we only traverse the suffix tree nodes to a bounded depth k, the suffix tree of the k-mer enumeration contains the same suffixes (to depth k) as our original sequence database's suffix tree. However, since the memory requirements of suffix trees are linear in size of the underlying sequence, the suffix tree built on the k-mer enumeration will require a smaller memory footprint.

Acknowledgments

The authors thank Bjarni Halldórsson and Clark Mobarry for helpful comments and suggestions on the manuscript.

References

1. Edwards, N., Lippert, R.: Generating peptide candidates from amino-acid sequence databases for protein identification via mass spectrometry. In: Proceedings of the Second International Workshop on Algorithms in Bioinformatics, Springer-Verlag (2002) 68–81

2. Cieliebak, M., Erlebach, T., Lipták, Z., Stoye, J., Welzl, E.: Algorithmic complexity of protein identification: Combinatorics of weighted strings. Discrete Applied Mathematics **137** (2004) 27–46
3. Eng, J., McCormack, A., Yates, J.: An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. Journal of American Society of Mass Spectrometry **5** (1994) 976–989
4. Perkins, D., Pappin, D., Creasy, D., Cottrell, J.: Probability-based protein identification by searching sequence databases using mass spectrometry data. Electrophoresis **20** (1997) 3551–3567
5. Bafna, V., Edwards, N.: SCOPE: A probabilistic model for scoring tandem mass spectra against a peptide database. Bioinformatics **17** (2001) S13–S21
6. Apweiler, R., Bairoch, A., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M.J., Natale, D.A., O'Donovan, C., Redaschi, N., Yeh, L.S.L.: UniProt: the Universal Protein knowledgebase. Nucl. Acids. Res. **32** (2004) D115–119
7. Welcome to UniProt — UniProt [the Universal Protein Resource] [online, cited June 24, 2004]. Available from: `http://www.uniprot.org/`
8. Kersey, P., Hermjakob, H., Apweiler, R.: VARSPLIC: Alternatively-spliced protein sequences derived from SWISS-PROT and TrEMBL. Bioinformatics **16** (2000) 1048–1049
9. UniProt/Swiss-Prot Tools [online, cited June 24, 2004]. Available from: `http://www.ebi.ac.uk/swissprot/tools.html`
10. CSC/ICSM Proteomics Section Home Page [online, cited June 24, 2004]. Available from: `http://csc-fserve.hh.med.ic.ac.uk/msdb.html`
11. The BLAST Databases [online, cited June 24, 2004]. Available from: `ftp://ftp.ncbi.nlm.nih.gov/blast/db/`
12. NRP (Non-Redundant Protein) Database [online, cited June 24, 2004]. Available from: `ftp://ftp.ncifcrf.gov/pub/nonredun/`
13. EBI Databases — International Protein Index [online, cited June 24, 2004]. Available from: `http://www.ebi.ac.uk/IPI/IPIhelp.html`
14. Garey, R., Johnson, D.: Computers and Intractability: A guide to the theory of NP-completeness. W. H. Freeman and Company, San Francisco (1979)
15. Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press (1997)
16. Bains, W., Smith, G.: A novel method for nucleic acid sequence determination. Journal of Theoretical Biology **135** (1988) 303–307
17. Lysov, Y., Floretiev, V., Khorlyn, A., Khrapko, K., Shick, V., Mirzabekov, A.: DNA sequencing by hybridization with oligonucleotides. Dokl. Acad. Sci. USSR **303** (1988) 1508–1511
18. Drmanac, R., Labat, I., Bruckner, I., Crkvenjakov, R.: Sequencing of megabase plus DNA by hybridization. Genomics **4** (1989) 114–128
19. Pevzner, P.A.: l-tuple DNA sequencing: Computer analysis. J. Biomol. Struct. Dyn. **7** (1989) 63–73
20. de Bruijn, N.: A combinatorial problem. In: Proc. Kon. Ned. Akad. Wetensch. Volume 49. (1946) 758–764
21. Kwan, M.K.: Graphic programming using odd or even points. Chinese Mathematics **1** (1962) 273–277