BIOINFORMATICS ORIGINAL PAPER
Vol. 23 no. 22 2007, pages 3016–3023
doi:10.1093/bioinformatics/btm417

Gene expression

PepSplice: cache-efficient search algorithms for comprehensive identification of tandem mass spectra

Franz F. Roos[1,*,†], Riko Jacob[4,†], Jonas Grossmann[2], Bernd Fischer[3], Joachim M. Buhmann[3], Wilhelm Gruissem[2], Sacha Baginsky[2] and Peter Widmayer[1]

[1]Institute of Theoretical Computer Science, [2]Institute of Plant Science, [3]Institute of Computational Science, ETH Zurich, CH-8092 Zurich, Switzerland and [4]Institut fuer Informatik, TU Munich, D-85748 Garching, Germany

Received on March 4, 2007; revised on July 24, 2007; accepted on August 11, 2007

Advance Access publication September 3, 2007

Associate Editor: John Quackenbush

ABSTRACT

Motivation: Tandem mass spectrometry allows for high-throughput identification of complex protein samples. Searching tandem mass spectra against sequence databases is the main analysis method nowadays. Since many peptide variations are possible, including them in the search space seems only logical. However, the search space usually grows exponentially with the number of independent variations and may therefore overwhelm computational resources.

Results: We provide fast, cache-efficient search algorithms to screen large peptide search spaces including non-tryptic peptides, whole genomes, dozens of posttranslational modifications, unannotated point mutations and even unannotated splice sites. All these search spaces can be screened simultaneously. By optimizing the cache usage, we achieve a calculation speed that closely approaches the limits of the hardware. At the same time, we control the size of the overall search space by limiting the combinations of variations that can co-occur on the same peptide. Using a hypergeometric scoring scheme, we applied these algorithms to a dataset of 1 420 632 spectra. We were able to identify a considerable number of peptide variations within a modest amount of computing time on standard desktop computers.

Availability: PepSplice is available as a C++ application for Linux, Windows and OSX at www.ti.inf.ethz.ch/pw/software/pepsplice/. It is open source under the revised BSD license.

Contact: franz.roos@alumni.ethz.ch or jacob@in.tum.de

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

In the last few years, tandem mass spectrometry has become a standard tool for high-throughput protein identification (Domon and Aebersold, 2006). Peptide tandem mass spectra are usually identified by searching them against protein databases, which in turn are mostly predicted from the DNA. In practice, if the genome of an organism is sequenced, a protein database search is usually sufficient to identify most tandem mass spectra whose signal is of reasonable quality.

Direct searches on the DNA are also sometimes used as complementary approaches (Choudhary *et al.*, 2001; Colinge *et al.*, 2005; Kuster *et al.*, 2001; Yates *et al.*, 1995) to verify and refine gene models. In such whole genome searches, the DNA is translated to protein using all six possible reading frames. Different gene prediction models (Mathe *et al.*, 2002) may yield contradictory results, which call for experimental data to resolve the conflict. However, eukaryotic genomes are usually a lot larger than their corresponding protein databases.

Splicing is ubiquitous in eukaryotes and contributes to protein diversity. Alternative splicing can generate several protein versions from the same gene, and the resulting protein versions may co-exist in the cell. Splicing has been studied extensively at the mRNA level, but has received little attention at the level of peptide tandem mass spectrometry, probably not least because splice site searches are computationally very intensive.

Previous work on spliced peptide searches consists of theoretical calculations on search complexity and statistical issues (Chen, 2001), especially on the frequency of random matches as a function of peptide length and genome size. The authors provide algorithms to identify spliced peptides on the genome, but only for searching a single spectrum against a sequence database and they test the algorithms only on a small simulated dataset. However, nowadays a typical dataset contains tens of thousands or even hundreds of thousands of experimental spectra. We show that searching many spectra simultaneously results in considerable efficiency gains. Colinge *et al.* (2005) published results on experimental data using their in-house commercial search tool OLAV/Phenyx (Colinge *et al.*, 2004) and focused on the biological aspects. They worked on tandem mass spectra of human proteins, used first-order Markov chains to predict splice donor and acceptor sites and limited the search area to regions around whole genome hits.

Even common protein database searches often represent a major bottleneck in many proteomics research facilities. Whole genome searches are more resource intensive because all six reading frames and all the intergenic regions need to be searched as well. Splice site searches are even more demanding because donor/acceptor combinations need to be screened

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

as well. However, such extensive searches may be the key to finding novel peptides, thereby refining existing gene models and predicting new ones.

One of the most established but also one of the computationally most demanding search tools is SEQUEST (Eng *et al.*, 1994). In conjunction with PeptideProphet (Keller *et al.*, 2002), it currently seems to provide the highest identification performance (Kapp *et al.*, 2005). Newer tools such as MASCOT (Perkins *et al.*, 1999), OLAV/Phenyx (Colinge *et al.*, 2004) and X!Tandem (Craig and Beavis, 2003, 2004) are quite fast. Information about algorithmic efficiency considerations is limited though, whereas the scoring is usually documented in more detail. While it is always possible to increase the computing power, algorithmic optimization may provide large efficiency gains and provide the same results at a much lower computing cost.

In this article, we focus on speed optimization and search space management for searches of tandem mass spectra against sequence databases. We present very fast, cache-optimized database search algorithms, whose speed closely approaches the theoretically achievable limit. Our implementation allows for searches of posttranslational modifications, whole genomes, point mutations and even unannotated splice sites, all simultaneously in an efficient and systematic way. Moreover, we demonstrate how this strategy translates into biologically interesting results.

In the algorithm design, we pay particular attention to general hardware properties, namely the fact that there are several storage levels of various speeds and sizes, such as the CPU cache, the RAM and the hard disk. Algorithms that take into account several storage hierarchies are called cache-aware, or cache-oblivious if they do not rely upon parameters about the properties of the storage levels. Such algorithms focus on the optimal order in which the data transfers should take place.

For the scoring, we use the hypergeometric model as described by Sadygov and Yates (2003), which has also recently been integrated into X!Tandem (Maclean *et al.*, 2006). The hypergeometric model is based on the shared peak count but also takes into account the peptide length and the number of peaks in the theoretical and the measured spectrum. Even though the hypergeometric model yields a *P*-value, we additionally estimate the false discovery rate by carrying out all searches both on the normal database and on a reversed peptide database. As in Sadygov and Yates (2003), we use the negative logarithm of the *P*-value as score. To obtain unambiguous identifications, we use the difference between the best and the second best score as discrimination criterion in the false discovery rate estimation. The speed optimization algorithms are independent of the scoring though.

To demonstrate the relevance to biological research, we applied the software to 1 420 632 tandem mass spectra. The underlying biological sample originated from an Arabidopsis plant cell culture. We searched the data against an Arabidopsis protein database and against the Arabidopsis genome (125 Mb), considering a variety of search spaces simultaneously, such as semi- and non-tryptic peptides, various posttranslational modifications, point mutations and a huge number of potential splice sites.

To prevent the combination of those search spaces from becoming prohibitively large, we restrict the number of variations that may co-occur per peptide, i.e. variations are mutually exclusive to some extent. This allows the user to combine all the different search spaces in one and the same search while avoiding an uncontrolled combinatorial explosion. The user only needs to specify the restriction parameters to limit the search space size. The search is then carried out as a single job on the CPU and the results for all search spaces are merged in one final summary.

Thanks to the cache-optimization and the search space restrictions, our implementation was able to rapidly screen a search space comprising semi-tryptic peptides, 29 different posttranslational modifications and the whole genome of Arabidopsis (125 Mb) on a single CPU. The throughput was more than 8 spectra per second, which exceeds the measurement speed of most current mass spectrometry instruments.

2 ALGORITHMS

2.1 The I/O model

In current computers, the speed of the CPU is often so high that the calculations themselves are not the bottleneck, but rather the transfer of data between the storage and the CPU. Computer storage consists of several levels, whose size and speed are usually inversely correlated. In current mass market computers, the random access time and the size of the hard disk are in the order of 10^{-2} s and 10^{11} bytes, while for the CPU cache, the values are rather 10^{-9} s and 10^{6} bytes, so the CPU cache is roughly a million times smaller than the hard disk, but also a million times faster.

In the I/O model (Aggarwal and Vitter, 1988), the running time is measured in I/O operations instead of CPU operations. The model has three components: a CPU, a memory that is infinitely fast but has a limited capacity of M elements and a disk that is slow but infinitely large. The CPU can only directly access data in the memory. Data on the disk can be accessed only after it is loaded into memory. Transfer of data between disk and memory takes place in entire blocks that have a size of B elements. Transferring one block costs one I/O. The number of I/Os is used as the performance measure. We assume $M \geq 4B$ and use $M' = M - 2B$, which is the amount of memory available to store spectra, after one block is reserved to store the current best score and similar information and another block is reserved to scan over another set of spectra.

In the case of spectrum database searches, let X be a spectrum dataset with N_X spectra and let Y be a peptide dataset with N_Y peptide sequences. The spectra are to be scored against the peptide sequences. Let k be the number of scores. If all against all elements are scored, $k = N_X N_Y$. In spectrum database searches however, only a small subset of all scores needs to be calculated since the parent mass of the peptide and the spectrum should match within some mass tolerance, so $k \leq N_X N_Y$.

Here, an element in the sense of the I/O model is a spectrum, either measured or synthesized from the protein database, or from a specific (spliced) part of the DNA, potentially modified by a PTM. In this modeling, we assume that the I/O algorithm

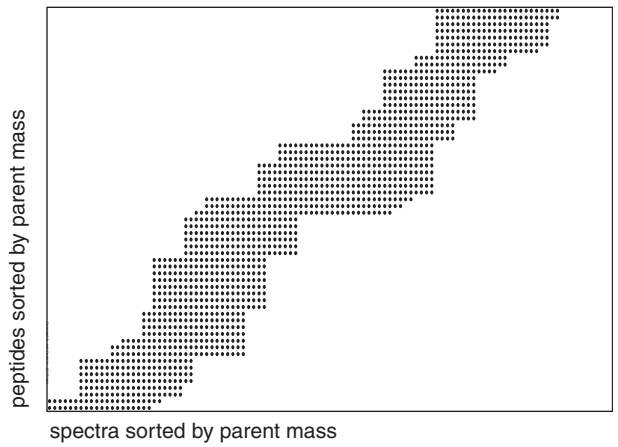

Fig. 1. Parent mass tolerance band within which scorings are required. The border increases monotonically from lower left to upper right. One axis represents the dataset X, the other axis the dataset Y, both sorted according to their parent mass. The area of the tolerance band is proportional to the number of scores k.

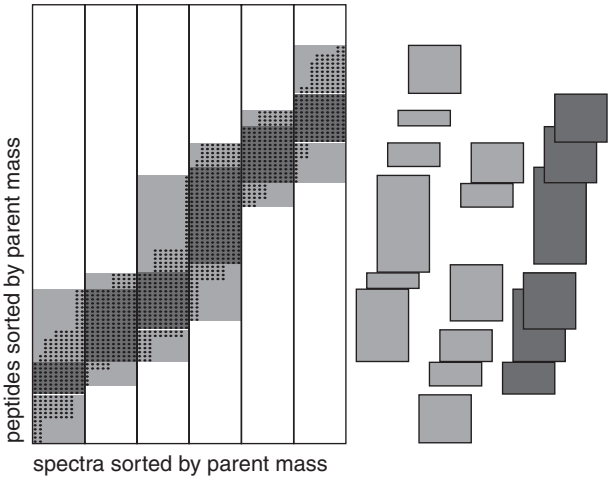

Fig. 2. The StripWise algorithm sweeps the tolerance band in vertical strips of width M' and within each strip by increasing row. Hence, an element of X (spectra, x axis) is loaded once, whereas an element of Y (y axis, peptides) is loaded once for every strip. All load operations work on full blocks and need only be performed if the corresponding parent masses are in the tolerance band. The peptides and spectra that are simultaneously loaded can be scored and are depicted as gray rectangles. The shading (dark/light gray) illustrates how I/O operations are justified (light: loading once, dark: many scores), as detailed in the proof of Theorem 2.1.

can only access the parent mass of a spectrum or score it against another spectrum. This reflects situations where the I/O algorithm is required to be independent of the scoring scheme, such that further analysis of a spectrum is meaningless (and hence disallowed in the model).

In the following, we assume that both the measured spectra and the synthesized spectra are sorted according to parent mass. This is a non-trivial assumption because sorting N elements takes $\Theta\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{M}\right)$ I/Os (Aggarwal and Vitter, 1988), which can (for small k) dominate the overall running time. This number of I/Os is in general already necessary for the special case $k = N_X = N_Y$ (R.Jacob, manuscript in preparation). Now, this assumption allows us to focus stronger on the number of I/O operations required for scoring two sets against each other.

Now, for a given measured spectrum the synthesized spectra with similar parent mass are consecutive in their list (and vice versa). The situation is depicted in Figure 1 that explains why the subset of scores that actually need to be calculated is called tolerance band.

The number of scorings within the tolerance band is usually proportional to the number N_X of experimental spectra and the number N_Y of theoretical spectra. However, it is inversely proportional to the parent mass tolerance between an experimental and a theoretical spectrum, i.e. the higher the mass precision of an instrument, the fewer scores need to be calculated. As a rule of thumb, one can estimate $k \approx y\ \frac{Nx \times Ny \times \text{tolerance}}{mass_{\max} - mass_{\min}}$.[1] The parent mass distribution is assumed to be uniform here, which is a simplification.

[1]Example: dataset with 100 000 spectra, search space of 2×10^8 peptides (penalty limit 1.5), mass tolerance window of $+5\,\text{Da} - (-1\text{Da}) = 6\text{Da}$, precursor mass range of $3000\text{Da} - 1000\text{Da} = 2000\text{Da}$. This gives $k \approx \frac{10^5 \times 2 \times 10^8 \times 6\text{Da}}{2000\text{Da}} = 6 \times 10^{10}$, which means that approximately 60 billion scores need to be calculated. In a test run, we actually counted 60 billion scores for 123 000 spectra instead of 100 000 spectra.

2.2 Stripwise tolerance band algorithm

One straightforward possibility to calculate all scores in the tolerance band is to proceed row by row. This is highly efficient provided that the tolerance band width never exceeds the memory capacity. In that case, the overall cost is equivalent to loading both datasets once, i.e. $\frac{N_X+N_Y}{B}$, which is the scanning bound for both datasets (see Algorithm 1 in Supplementary Material for proof). However, especially for large spectrum datasets and large precursor mass tolerances, the tolerance band may be so large that the cumulated size of all the spectra in a row exceeds the memory capacity. In this case, spectra will be evicted before the row is completed and most of the evicted spectra need to be reloaded in the next row. Neighboring rows usually have many spectra in common. To prevent such inefficiencies, we can organize the computation in a stripwise fashion that we call StripWise algorithm (Fig. 2). The width of the strips respects the memory size.

Theorem 2.1. *For two sorted sets of spectra of sizes N_X and N_Y, the algorithm StripWise computes all k scores within the tolerance band in $O\left(1 + \frac{N_X+N_Y}{B} + \frac{k}{MB}\right)$ I/Os.*

Proof. Every element of X is loaded precisely once as part of its full block, leading to a total of $O\left(\frac{N_X}{B}\right)$ I/Os. Loading a block of B elements of Y is justified either because it allows many scorings or it makes some global progress. The first case happens if all $B \cdot M'$ scores that can now be performed are inside the tolerance band (dark gray rectangles in Fig. 2). This leads to $O\left(\frac{k}{BM}\right)$ I/Os. The second case happens if one

of the potential scores is unnecessary because it is outside the tolerance band, the upper or lower boundary of the tolerance band intersects the light gray rectangle in Figure 2. Because both boundaries are monotonic they can each intersect with at most $N_X/M' + N_Y/B$ such rectangles (the 'next' rectangle is either 'right' or 'up'). Hence, the overall number of I/Os of StripWise is $O\left(\frac{N_X+N_Y}{B} + \frac{k}{MB}\right)$ as stated in the theorem.

2.3 Recursive tolerance band

It is possible to design an algorithm that is I/O efficient without knowing the parameters B and M (Frigo *et al.*, 1999). Here, this is done with a recursive approach where we split the task at a parent mass and recursively compute the scores for all spectra of smaller parent mass and then the others. This split happens either for the measured or the theoretical spectra, whichever are more. The recursion ends if the active set either consists of precisely one spectrum of each type, in which case we compute the score. The recursion ends also if the active set is such that all of its scores are out of the tolerance band. We call this algorithm RecursiveBand. Because both sets of spectra are arrays sorted by parent mass, splitting does not require to touch all spectra. Indeed, one recursive invocation accesses only a constant number of spectra directly. Asymptotically, this yields the same I/O-performance as StripWise.

Theorem 2.2. *For two sorted sets of spectra of sizes N_X and N_Y, the algorithm* RecursiveBand *computes all k scores within the tolerance band in $O\left(1 + \frac{N_X+N_Y}{B} + \frac{k}{MB}\right)$ I/Os in the cache-oblivious model.*

Proof. In the cache-oblivious model it is assumed that I/O operations are performed optimally, such that the analysis may prescribe specific I/O operations. If $N_X + N_Y < M$ the bound holds trivially by loading all spectra once. Here, we focus on recursive calls where the two active sets fit together into memory, but this is not the case for the caller. Such a call hence costs at most M/B I/Os because all recursive calls find the needed data in memory. The size of the active set is at least $M/2$ (otherwise we would have considered the caller). If $N_X > M/4$ and $N_Y > M/4$ we can conclude that the active set comprises at least $M/4$ spectra of each type, leading to at least $M^2/16$ possible scores. Now, similar to the analysis of StripWise, we distinguish the calls depending on the tolerance band. The total number of calls that are completely within the tolerance band is at most $16k/M^2$, leading to a total of $O\left(\frac{k}{BM}\right)$ I/Os. The number of calls who intersect the border (not all possible scores are within the tolerance band), or all calls if $N_X \leq M/4$ or $N_Y \leq M/4$, is by monotonicity of the border at most $4\frac{N_x+N_y}{M}$, leading to a total of $O\left(\frac{N_X+N_Y}{B}\right)$ I/Os. The remaining calls form the inner nodes of a rooted binary tree whose leafs are calls that are already accounted for with at least one I/O, in the worst doubling the I/O cost.

2.4 General lower bounds

It is well known that accessing N different items requires $\lceil N/B \rceil$ I/O operations, such that the $O\left(1 + \frac{N_X+N_Y}{B}\right)$ terms in our running times are necessary. The proposed algorithms have in fact asymptotically optimal I/O behavior, as implied by the following theorem.

Theorem 2.3. *Assume an I/O algorithm computes a total of k different scores and each score computation requires precisely two specific spectra to be in memory. Then the algorithm must perform $\frac{k}{BM}$ I/O operations.*

Proof. As a normalization, we can assume that the algorithm computes a score at the earliest possible time. Hence, all scores are performed following some input operation. Any input operation brings at most B new spectra in. Because there are at most M spectra in memory, the number of scores that can be computed before the next I/O operation is at most MB.

2.5 Timing experiments: I/O cost versus CPU cost

We tested the actual search speed of the implementation to obtain an estimate on the I/O time and CPU time needed per score. To this aim, we calculated each score not only once but repeatedly. We assume that right after the score has been calculated, the data needed for the score still reside in the fastest possible cache and will be available with the smallest possible delay when the score is recalculated right afterwards. We tested the approach on an Intel Xeon 3.0 GHz processor using different repetitions and dataset sizes. For a dataset comprising 152 838 spectra, we obtained a CPU time of approximately 250 CPU cycles per score (wall clock time including L1 cache latency). The overall computing time was approximately 450 CPU cycles per score, which means that at most 200 CPU cycles per score are attributable to I/O delays (Fig. 3). This means that we succeeded in reducing memory transfer time to below CPU time.

2.6 Timing experiments: benchmarking versus X!Tandem

In a benchmarking experiment, we compared the search speed of our tool against the search speed of X!Tandem

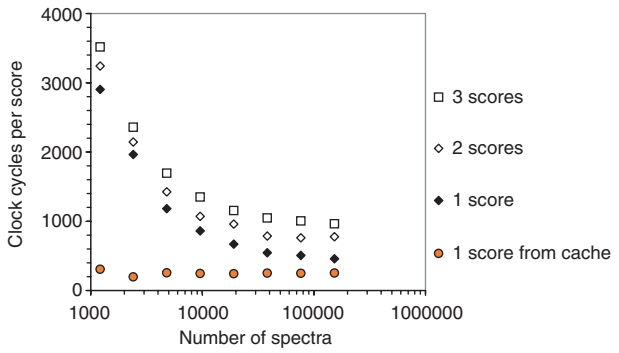

Fig. 3. We calculated all scores either once, or repeatedly a second and a third time for datasets of different sizes. The additional cost for a repeated score calculation (circles) is very similar for all dataset sizes and corresponds to the inevitable delay that is caused by the retrieval from the L1 cache and the CPU calculations. The I/O cost per score depends on the dataset size. The larger the dataset, the more I/O cost savings are realized and the cost per score approaches the inevitable CPU/L1 cost.

(Craig and Beavis, 2003, 2004), an open source tandem mass spectrum database search tool that is generally considered to be fast. On a dataset of 2078 spectra, we did a fairly time-consuming search that is comparable between different tools. We found it rather challenging to identify a set of parameters for which X!Tandem and PepSplice would have to search an identical and rather large search space, since both use different heuristic pruning approaches to actually reduce such large search spaces. We chose a non-tryptic search, i.e. with unspecific cleavage. In the X!Tandem parameter file, it was recommended to allow, e.g. 50 missed cleavages in that case, presumably because each peptide bond is a potential missed cleavage in that kind of search. For PepSplice, we did not even limit the number of missed cleavages altogether. However, there is an implicit limit given by the parent mass distribution of the spectra, since it is pointless to generate peptides exceeding the highest parent mass in the spectrum dataset plus the mass tolerance. The mass tolerance was $-1/+5$ Da around the monoisotopic parent mass. Our tool searched through 4.93×10^8 peptides and calculated 2.61×10^9 scores, completing the search in 41 min (0.84 spectra per second), whereas X!Tandem version 06-09-15 took 316 min to complete the search (0.11 spectra per second). Even on a much larger dataset of 12 545 spectra, our tool still took only 82 min. The timing refers to a single core of an Intel Pentium 3 GHz dual-core CPU. We could process two PepSplice jobs in parallel without any slowdown, the machine gave the performance of two single CPUs.

3 BIOINFORMATICS METHODS

3.1 Spectrum preprocessing

We obtained better identification performance when only the most intense peaks were used for scoring. In ion trap tandem mass spectra, the peak intensity in the central region of the spectrum is higher than in the low-mass and high-mass region. Therefore, we normalized (flattened) the peak intensity within each spectrum using an approach proposed by Pletscher *et al.* (2006) (see Supplementary Material). Per 100 Da parent mass of a spectrum, we then retained the highest 10 peaks, i.e. for a spectrum of parent mass 1500, we retained a total of 150 peaks after normalization. Afterwards, we binarized and discretized the spectra using a bin spacing of 1.00048 Da (Perkins *et al.*, 1999).

3.2 Size of peptide search space

The number of fully tryptic peptides in a protein database is approximately equal to the number of tryptic cleavage sites if the digest is complete. If 2 out of 20 amino acids are K or R, the number of tryptic peptides is a 10th of the number of amino acids in the protein database. For the Arabidopsis database with a size of 12 million amino acids, we counted 1.1 million tryptic, unmodified peptides. However, a plethora of other peptide variants can be derived from the same protein database. If one missed cleavage is admitted, the number rises to 2.9 million, for semitryptic peptides, the number becomes 25 million, for a set of 21 PTMs (at most one per peptide), the number is 19 million peptides. For whole genome searches in Arabidopsis, the number rises to 20 million peptides.

3.3 Limiting the combinatorial explosion

The size of the search space depends on a variety of biological effects. Including many independent peptide variations at once into a search usually has a multiplicative effect on the number of peptides that the search space comprises. However, if these variations are rare and independent, we assume that the combination of them is consequently even rarer. While the search space grows exponentially in the permitted variations, the number of additional, reliable identifications may be very limited because most of the combinations are improbable while the higher frequency of false positives requires more conservative score thresholds for the acceptance of identifications.

Therefore, we limit ourselves to the more plausible combinations and instead of multiplying different search spaces with each other, we basically add them to each other by exploring each search space separately. We do not separate them completely though, we still allow for some degree of combinations between the search spaces. We achieve this by assigning a weighted penalty to each variation and by limiting the cumulated penalty per peptide. This gives great flexibility in combining plausible variations while limiting the search space. We used the following penalties: 0.3 per missed cleavage, 0.3 per oxidized methionine, 1.2 per posttranslational modification, 1.2 per non-tryptic terminus, 1.2 for whole genome search hits, 3.2 for single nucleotide substitutions enumerated from the genome. For illustration, a peptide with two missed cleavages and a posttranslational modification incurs a penalty of $2 \times 0.3 + 1.2 = 1.8$. These penalty values approximately correspond to the decimal logarithm of the increase in the search space that each variation causes, i.e. increasing the penalty limit from 1.2 to 2.2 increases the number of peptides in the search space by an order of magnitude. All the possible variations per peptide are enumerated recursively.

The enumeration is based on common FASTA protein or DNA databases as input from which all peptide variations are generated on the fly in the memory. The great majority of peptide variations have a different parent mass than the original peptide. Therefore, a modified duplicate object is created for each variation. This child object may in turn again be duplicated to give additional variations. However, each modified child object has higher penalties than its parent object, so eventually the recursion is terminated once the penalty limit is reached. Variations are enumerated on several recursion levels, in the following top-down hierarchy: (1) Unmodified amino acids. This level comprises tryptic, semi-tryptic and non-tryptic (unspecifically cleaved) peptides. When reading DNA, a reading window progresses along the protein or DNA sequence and continuously and exhaustively generates peptide objects from all six reading frames at once. When searching for alternative splice sites, all tryptic peptides with gaps corresponding to potential GT-AG introns are enumerated as well, using a slightly larger reading window that spans the gap. (2) Enumeration of point mutations takes place based on the previously enumerated, unmodified peptides; (3) on the next level, N- and C-terminal modifications are enumerated; (4) finally, non-terminal modifications are enumerated. During the entire recursion, all peptide objects are continuously collected in a peptide object buffer. Every time the buffer is full, the peptide objects are sorted by parent mass and then processed against the measured spectra in the cache-optimized manner as described. A short list of best-matching peptide objects is maintained for each spectrum throughout the search. Peptide objects that are not elements of such a list are continuously discarded.

While we based our penalty system on the size of the peptide search space, some variations may be of much more biological interest in an experiment than others. The weighting may be adjusted according to subjective goals, thereby balancing the relation between computational cost and biological 'profit'. One could also develop a weighting based on expected frequencies of phenomena, possibly in conjunction with a validation system. Regardless of the weighting chosen, the strength

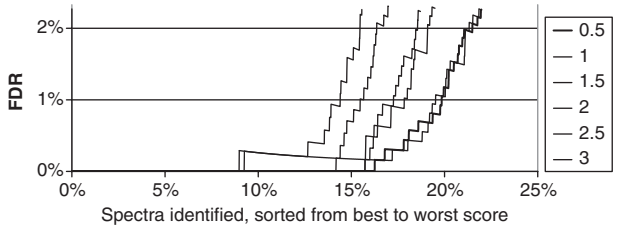

Fig. 4. The larger the search space, the higher the false discovery rate and the lower the fraction of all spectra that can be identified. At a penalty limit of 0.5, which corresponds to a search space of 3.5 million peptides, more than 20% of all doubly charged spectra are identified here at a false discovery rate of 1%. Conversely, at a penalty limit of 3.0, which corresponds to 3.6 billion peptides, only ~14% of all spectra can be identified at a false discovery rate of 1%.

of the penalty system is that a great variety of peptides can be searched simultaneously with few parameters, using moderate computing power and at a moderate false discovery rate.

3.4 Estimation of false discovery rates

The larger the search space, the higher the risk of false positive assignments. We determine the false discovery rate as follows: for each peptide, we also add its reversed sequence to the database. We do not reverse the C-terminal amino acid though since this is most often K or R due to the tryptic digest. ACDEK thus becomes EDCAK. If a spectrum is assigned to a reversed peptide, this strongly suggests that the assignment is incorrect. But since there are equally many forward and reversed sequences in the database, a spectrum with a lack of interpretable signal is almost as likely to be randomly assigned to one of the many forward sequences in the database. Every hit from the reverse database is a random database hit. A hit from the forward database can either be a true identification or a random hit. Regarding it in the other way: A random hit results almost equally likely in a hit from the forward database as a hit from the reverse database. Thus, among all forward hits, we can estimate the number of random hits to be the number of hits from the reverse database (Balgley *et al.*, 2007; Elias and Gygi, 2007; Elias *et al.*, 2005). Therefore, we can estimate the false discovery rate for the forward hits above the acceptance cutoff[2] as $\frac{\text{number of reverse}}{\text{number of forward}}$. Indeed, we observe that at low scores, forward and reverse sequences occur at approximately a 50:50 ratio, while at high scores, virtually no reverse sequences are found, as expected. In Figure 4, we show how the cumulated rate of reverse identifications increases as the quality of the assignment score decreases.

We apply further checks to estimate the rate of false positives. We also use decoy approaches other than the reversed sequence approach: for the posttranslational modifications (PTMs), we additionally search non-existing PTMs and for the whole genome search we add a fly chromosome to the Arabidopsis genome. We generally attempt to search as comprehensively as possible to avoid misexplaining spectra by closely related sequences, e.g. misexplaining a PTM as a splice site.

[2]Elias *et al.* estimate the false discovery rate as (2 × number of reverse)/(number of forward + number of reverse) for *forward* and *reverse* hits above the acceptance cutoff. Our estimate of the FDR is smaller as long as the number of forward hits is larger than the number of reverse hits. Furthermore, the false discovery rate is called false positive rate in Elias *et al.*

Table 1. Identifications of four searches with increasing search space, ranging from 6.5×10^6 to 1.4×10^{10} peptides in size

	Spectra	Unique peptides	FDR (%)
1 420 632 spectra, limit 0.6, 5 h, 6.5×10^6 peptides			
Tryptic, 0 missed cleavage	168 940	30 316	1.0
Tryptic, 1 missed cleavage	19 377	4841	1.0
Tryptic, 2 missed cleavage	1058	294	1.0
1 420 632 spectra, limit 1.5, 50 h, 2.0×10^8 peptides			
19 real + 10 decoy PTMs	6485	1700	1.0
Only N-terminus tryptic	1701	465	1.0
Only C-terminus tryptic	4704	1676	1.0
Genome only	1078	241	1.0
98 437 spectra, limit 2.7, 27 h, 2.3×10^9 peptides			
Non-tryptic	36	21	2.8
2 PTMs on same peptide	92	51	1.1
98 437 spectra, limit 3.3, 209 h, 1.4×10^{10} peptides			
Point mutations	350	220	1.0
Genome only, spliced	61	22	1.7

We used a mass tolerance of −1/+5 Da around the monoisotopic parent mass. Semi-tryptic peptides are shown in two categories, either with non-tryptic N-terminus or C-terminus. There is a clear bias towards tryptic C-terminus and non-tryptic N-terminus, probably due to the fact that a tryptic C-terminus contains a basic residue and thus a positive charge, which is not necessarily the case for a tryptic N-terminus. Total 168 940 peptides are fully tryptic with no missed cleavage site, versus only 1058 with two missed cleavage sites, which indicates that the tryptic digest was complete or nearly complete. For the time intensive searches, we extracted a subset of 98 437 unidentified high-quality spectra from the 1 420 632 spectra using the QualScore tool (Nesvizhskii *et al.*, 2006). Search hours refer to time on Intel Pentium 3 GHz dual-core processors, where we (conservatively) consider each dual-core as two separate CPUs. The default false discovery rate used was 1%. Higher false discovery rates are indicated if only few spectra were identified per subgroup, which made empirical estimation of the number of false positives more difficult.

To estimate the false discovery rate, grouping together identifications from very different search space sizes may be problematic. A score that is sufficiently high to identify a tryptic peptide may not be sufficient to identify a genomic splice site. We therefore extracted homogeneous subgroups of identifications that shared similar search space sizes and made an individual false positive estimation for each subgroup. This approach is quite conservative and resulted in fewer identifications than an estimation on mixed search spaces.

4 BIOLOGICAL RESULTS

It has been suggested that spectra which remain unassigned in an initial search should be submitted to more extensive searches (Nesvizhskii *et al.*, 2006; Sadygov *et al.*, 2004). In a similar approach, we searched our spectra using four different search spaces ranging from 6.5×10^6 to 1.4×10^{10} theoretical peptides. The number of identifications in each search is summarized in Table 1. A detailed summary of the posttranslational modifications is shown in Table 2 and a distribution of the identifications on the genome is depicted in Figure 5.

The biological sample stemmed from an Arabidopsis cell culture. It was fractionated and then measured on an LTQ ion trap tandem mass spectrometer, as described in a previous publication of this journal (Fischer *et al.*, 2006).

Table 2. We found 6485 PTM identifications, of which 165 were decoy PTM identifications

Modification	AA	[Da]	Δ	[Da]	Spectra
Methionine oxidations	M	131	+16	147	2181
N-terminal pyroglutamate	Q	128	−17	111	1983
N-terminal acetylation	X		+42	42	421
Methylester	E	129	+14	143	300
Mono-methylation	K	128	+14	142	245
Methylester	D	115	+14	129	203
Di-methylation	N	114	+28	142	202
Methylation	N	114	+14	128	197
Methylation	Q	128	+14	142	104
Di-methylation	K	128	+28	156	96
Phosphorylation	S	87	+80	167	88
Hydroxylation	P	97	+16	113	74
Oxidation	W	186	+16	202	74
Mono-methylation	R	156	+14	170	57
N-terminal pyroglutamate	E	129	−18	111	39
Methylation	C	103	+14	117	32
Tri-methylation	K	128	+42	170	19
Phosphorylation	Y	163	+80	243	3
Phosphorylation	T	101	+80	181	2
Decoy	S	87	−67	20	14
Decoy	S	87	−47	40	14
Decoy	S	87	−27	60	31
Decoy	S	87	−7	80	12
Decoy	S	87	+13	100	54
Decoy	S	87	+33	120	10
Decoy	S	87	+53	140	7
Decoy	S	87	+73	160	12
Decoy	S	87	+93	180	2
Decoy	S	87	+113	200	9

Decoy PTMs are much less frequently identified than most of the real PTMs, even though the amino acid serine that we used for the decoy PTMs is fairly frequent in Arabidopsis. The unmodified serine residue has a monoisotopic mass of 87.03203 Da.

4.1 Co-occurrence of several variations

For illustration, we discuss a few cases where we found several variations at once on the same peptide:

(1) The peptide Q111IVAPVPHDSYSVLSVSSSGK was identified as a whole genome hit including a PTM. The exon that underlies the TAIR protein database is just one single nucleotide too short to accommodate the identification. A competing gene model from EuGene (Foissac *et al*., 2003) is compatible with the hit, however. We also identified the same peptide without the pyroglutamate modification, which supports the identification and the gene model of EuGene.

(2) Examples for differential modifications: E143FLELAEGLKGSLK versus E143FLELAE143GLKGSLK, GITIDIALW202K156FETTK versus GITIDIALW202K170FETTK.

(3) We identified four peptide variants with both pyroglutamate and Trp oxidations (Q111IGVIGW202GSQGPAQAQNLR, 1 spectrum), with pyroglutamate only (35 spectra), with Trp oxidation only (2 spectra) and with pyroglutamate and a missed cleavage (Q111IGVIGWGSQGPAQAQNLRDSLVEAK, 3 spectra). This nicely illustrates that our search approach permits to capture many peptide variations at once.

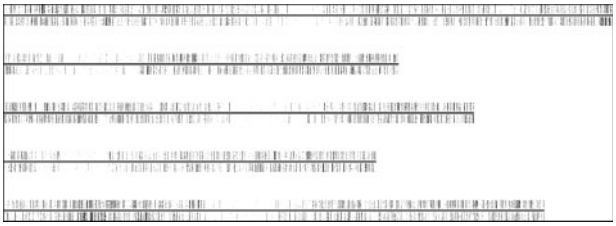

Fig. 5. The figure shows the five chromosomes of Arabidopsis as horizontal lines. Each dot represents a spectrum that was identified both in the protein database and in the genome. Identifications in forward reading frames are shown above the chromosome, reverse reading frames below. Around the centromeres, only few identifications were found.

4.2 Splice sites

For spliced peptides, we applied a maximum gap length of 3000 nt. Total 61 spectra (22 peptides) were best explained by an unannotated splice site, i.e. they conflicted with an exon prediction. If the predicted exon is shorter than the real exon, whole genome hits are likely to occur in the unpredicted part. We found several examples where newly predicted splice sites co-occur within the same open reading frame as whole genome hits. The most striking case is located in the open reading frame At1g64790, where we find two whole genome hits without any hit in the protein database and two splice site hits without any hit in the protein database. The splice hits are supported by 3 and 4 spectra, the whole genome hits by 14 and 15 spectra, respectively (Fig. 6). The ORF spans 16 677 nt and contains 58 predicted exons. Several competing gene predictions contradict each other around the position 24083000 on the chromosome. An alternative Arabidopsis protein prediction (gi5042415) is 2698 amino acids long instead of 2441 and accommodates both whole genome identifications and one of the spliced peptide identifications. This confirms three of our four hits. Our results are also compatible with homologous gene predictions in rice (Roos, 2006). Using our algorithms, we can therefore detect spliced peptides with reasonable confidence, in spite of the huge search space.

5 CONCLUSION

We conclude that by systematically submitting tandem mass spectra to extended searches, a significant number of new peptides can be identified, including posttranslational modifications, semi-tryptic peptides, whole genome hits and even splice site hits, which may help to refine genome annotation.

As a result of speed optimization, these searches are feasible on standard computing equipment. We show that cache optimization greatly helps to minimize search time, as does careful search space management. By defining the search space via

```
gi|42562949|ref|NP_176659.2| unknown protein [Arabidopsis thaliana]
gi|5042415|gb|AAD38254.1| similar to translational activator [Arabidopsis thaliana]
Score = 4267 bits (11067),  Expect = 0.0, Method: Composition-based stats.
Identities = 2407/2514 (95%), Positives = 2410/2514 (95%), Gaps = 95/2514 (3%)

                                                          IGMLNAVQELASAP
Query  241   RRLGALSMVMCLSEKSSNPDTIEAMFASVKAIIG------------------VQELASAP  282
             RRLGALSMVMCLSEKSSNPDTIEAMFASVKAIIG                  VQELASAP
Sbjct  446   RRLGALSMVMCLSEKSSNPDTIEAMFASVKAIIGGSEGRLQSPHQRIGMLNAVQELASAP  505

             EGK                              LSILSAVASWASR
Query  283   EGKYIGSLSRTICSFLIACYKDE-------------ASWASRSSVAIQPNLVSFIAAGLK  329
             EGKYIGSLSRTICSFLIACYKDE              ASWASRSSVAIQPNLVSFIAAGLK
Sbjct  506   EGKYIGSLSRTICSFLIACYKDEGNEDVKLSILSAVASWASRSSVAIQPNLVSFIAAGLK  565

                                NPDTISQISDLLSPLIQLVK
Query  330   EKEALRRGHLRC-------------ISDLLSPLIQLVKTGFTKAVQRLDGIYALLIVSKI  376
             EKEALRRGHLRC             ISDLLSPLIQLVKTGFTKAVQRLDGIYALLIVSKI
Sbjct  566   EKEALRRGHLRCVRIICRNPDTISQISDLLSPLIQLVKTGFTKAVQRLDGIYALLIVSKI  625

                                          ...

                                                SGPLPVDTFTFIFPILER
Query  796   FDFRPSVDKAGKTYEGLFERIVNGLSISCKSGPLPVDTFTFIFPVLYHVLGVVPAYQASV  855
             FDFRPSVDKAGKTYEGLFERIVNGLSISCKSGPLPVDTFTFIFPVLYHVLGVVPAYQASV
Sbjct  1038  FDFRPSVDKAGKTYEGLFERIVNGLSISCKSGPLPVDTFTFIFPVLYHVLGVVPAYQASV  1097
```

Fig. 6. The four hits in At1g64790 are aligned against two conflicting protein predictions. The first one corresponds to the TAIR protein database. The two whole genome hits (IGMLNAVQELASAPEGK, LSILSAVASWASR) are explained by the second sequence. One of the splice hits is also explained by the second sequence (NPDTISQISDLLSPLIQLVK), but not the other splice hit (SGPLPVDTFTFIFPILER).

a penalty system, many different peptide variations can be searched at once while the overall size of the search space is kept under tight control.

We were able to search the whole genome of Arabidopsis (125 Mb), semi-tryptic peptides and 29 PTMs simultaneously at a rate of 8 spectra per second on a single CPU, which is faster than most instruments are currently able to measure.

ACKNOWLEDGEMENTS

This research was supported by TH-41/02-2 and also by the Functional Genomics Center Zurich, the Swiss Initiative in Systems Biology (SystemsX.ch: C-SPMD and C-MOP) and by ETH grant TH-5/04-3.

Conflict of interest: none declared.

REFERENCES

Aggarwal,A. and Vitter,J.S. (1988) The input output complexity of sorting and related problems. *Commun. ACM*, 31, 1116–1127.

Balgley,B.M. *et al*. (2007) Comparative evaluation of tandem ms search algorithms using a target-decoy search strategy. *Mol. Cell Proteomics*.

Chen, T. (2001) Gene-finding via tandem mass spectrometry. In *RECOMB 2001 Proceedings of the Fifth Annual International Conference on Computational Biology*. Montreal, Canada, pp. 87–94.

Choudhary,J.S. *et al*. (2001) Interrogating the human genome using uninterpreted mass spectrometry data. *Proteomics*, **1**, 651–667.

Colinge,J. *et al*. (2004) High-performance peptide identification by tandem mass spectrometry allows reliable automatic data processing in proteomics. *Proteomics*, **4**, 1977–1984.

Colinge, J. *et al*. (2005) Experiments in searching small proteins in unannotated large eukaryotic genomes. *J. Proteome Res*., **4**, 167–174.

Craig,R. and Beavis,R.C. (2003) A method for reducing the time required to match protein sequences with tandem mass spectra. *Rapid Commun. Mass Spectrom*., **17**, 2310–2316.

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

Domon,B. and Aebersold,R. (2006) Mass spectrometry and protein analysis. *Science*, **312**, 212–217.

Elias,J.E. and Gygi,S.P. (2007) Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nat. Methods*, **4**, 207–214.

Elias,J.E. *et al*. (2005) Comparative evaluation of mass spectrometry platforms used in large-scale proteomics investigations. *Nat. Mehods*, **2**, 667–675.

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.

Fischer,B. *et al*. (2006) Semi-supervised lc/ms alignment for differential proteomics. *Bioinformatics*, **22**, e132–e140.

Foissac,S. *et al*. (2003) Eugene's hom: a generic similarity-based gene finder using multiple homologous sequences. *Nucleic Acids Res*, **31**, 3742–3745.

Frigo, M. *et al*. (1999) *Cache-oblivious algorithms*. *Master Thesis*. MIT.

Kapp,E.A. *et al*. (2005) An evaluation, comparison, and accurate benchmarking of several publicly available ms/ms search algorithms: sensitivity and specificity analysis. *Proteomics*, **5**, 3475–3490.

Keller,A. *et al*. (2002) Empirical statistical model to estimate the accuracy of peptide identifications made by ms/ms and database search. *Anal. Chem*., **74**, 5383–5392.

Kuster,B. *et al*. (2001) Mass spectrometry allows direct identification of proteins in large genomes. *Proteomics*, **1**, 641–650.

Maclean,B. *et al*. (2006) General framework for developing and evaluating database scoring algorithms using the tandem search engine. *Bioinformatics*.

Mathe,C. *et al*. (2002) Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Res*., **30**, 4103–4117.

Nesvizhskii,A.I. *et al*. (2006) Dynamic spectrum quality assessment and iterative computational analysis of shotgun proteomic data: toward more efficient identification of post-translational modifications, sequence polymorphisms, and novel peptides. *Mol. Cell Proteomics*, 5, 652–70.

Perkins,D.N. *et al*. (1999) Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, **20**, 3551–3567.

Pletscher,P. *et al*. (2006) Peptide assignment validation: telling what's wrong without actually knowing what's right. *Semester Thesis*. ETH.

Roos,F.F. (2006) Algorithms for peptide identification by tandem mass spectrometry. *Ph.D Thesis 16844*, ETH Zurich, http://e-collection.ethbib.ethz.ch/ecol-pool/diss/fulltext/eth16844.pdf.

Sadygov,R.G. and Yates,J.R., r. (2003) A hypergeometric probability model for protein identification and validation using tandem mass spectral data and protein sequence databases. *Anal. Chem*., **75**, 3792–3798.

Sadygov,R.G. *et al*. (2004) Large-scale database searching using tandem mass spectra: looking up the answer in the back of the book. *Nat. Methods*, **1**, 195–202.

Yates,J.R., r. *et al*. (1995) Mining genomes: correlating tandem mass spectra of modified and unmodified peptides to sequences in nucleotide databases. *Anal. Chem*, 67, 3202–10.