Supplementary Notes

Xolik: finding cross-linked peptides with maximum paired scores in linear time

Jiaan Dai*, Wei Jiang*, Fengchao Yu*, Weichuan Yu

July 17, 2018

Contents

*Jiaan Dai, Wei Jiang and Fengchao Yu contributed equally to this work.

1 Proof of correctness

Theorem 1. *Algorithm* Xolik *returns the maximum paired score among all possible pairs satisfying the constraint of the precursor mass.*

Proof. For simplicity, we suppose similarity scores are computed in advance. Let $S[i]$ denote the similarity score between the mass spectrum and the ith peptide. At the end of each iteration, we prove the following properties:

1. $[I_{bf}, I_{be}]$ indicates the range of indices that satisfies $PM - M[I_f] - XM - \epsilon_1 < M[j] < PM - M[I_f] - XM + \epsilon_1$ for $j \in [I_{bf}, I_{be}]$.

2. Elements $e \in D$ are stored in descending order.

3. Elements $e \in D$ satisfy $e \in [I_{bf}, I_{be}]$.

4. For $e_1 \leq e_2$ with $e_1 \in D, e_2 \in D$, there is $S[e_1] \leq S[e_2]$.

5. The front of the deque D is the index with the largest $S[j], j \in [I_{bf}, I_{be}]$.

In the initialization stage:

1. Step 1 moves I_{be} to the index whose corresponding mass is just smaller than M_u. Step 2 moves I_{bf} from $I_{be}+1$, and stops when $M[I_{bf}-1] \geq M_l$. So the first property is proved.

2. A new element e (i.e., a peptide index) can only be pushed into the deque D from the back. Because I_{bf} monotonically decreases, elements inside D are stored in descending order.

3. Before the initialization stage, D is empty. Therefore, after pushing elements, all elements are in the range of $[I_{bf}, I_{be}]$.

4. $e_1 \leq e_2$ means that e_1 is pushed into the deque later.

   (a) If D is empty before pushing e_1 into D, then e_1 will be pushed into D directly.

(b) If there is one element e_2 in D before pushing e_1 into it, then we have the following two circumstances:

i. If $S[e_1] > S[e_2]$, e_2 will be popped out from the back. And e_1 will be pushed back into D. We still have one element in the deque.

ii. If $S[e_1] \leq S[e_2]$, e_1 will be pushed from the back. In this case, the property is proved.

(c) If there is more than one element in D before pushing e_1 into it, we use mathematical induction to prove the property holds in this case. Proof 4 (b) shows the property holds in base cases. Assume the property holds before pushing e_1. Comparing e_1 with e_2 from the back ensures that all $e_2 > e_1$ with $S[e_2] < S[e_1]$ will be popped out. When it stops, it means that there is $e_3 > e_1$ and $S[e_3] \geq S[e_1]$ (or the deque is empty). Because before pushing e_1 Property 4 holds, for all $e \geq e_3$, there is also $S[e] \geq S[e_3] \geq S[e_1]$. In this case, Property 4 still holds after pushing e_1 back into D.

5. Before pushing elements, the deque is empty. Therefore, the index with the largest $S[e]$ ($e \in [I_{bf}, I_{be}]$) will be the front of the deque, from the proof of Property 4.

In each iteration:

1. The proof of Property 1 follows the same procedure as in 1) in the initialization stage.

2. The proof of Property 2 follows the same procedure as in 2) in the initialization stage.

3. Property 2 (i.e., all elements $e \in D$ are stored in descending order) holds in the previous iteration, and we stop PopFront() when we see an element e with $e \leq I_{be}$. This ensures that all $e > I_{be}$ have already popped out. Also, when decreasing I_{bf}, only e with $M[e] \geq M_l$ will be pushed into the deque. Therefore Property 3 holds.

4. Property 4 is satisfied in the previous iteration. Removing $e > I_{be}$ does not change this property. Therefore, pushing a new element from the back follows the same

behavior as described in the proof in the initialization stage. Hence, Property 4 holds.

5. After removing $e > I_{be}$, the elements in D are the overlapped index range from the previous I_f to the current I_f. And Property 4 holds in D.

   (a) If the index e_{max} with the maximal score $S[e]$ is in the overlapped part, then it is already the front of D after removing $e > I_{be}$. Pushing any new element e_1 from the back will not change the front because there is $S[e_1] \leq S[e_{max}]$ for all new elements e_1.

   (b) If the index e_{max} is among the newly added e_1, then after pushing this e_{max} into the deque, it will pop out all elements that are already in the deque from the back. At the same time, it becomes the front of D. Therefore, the front of the deque is the index with maximal $S[e]$ ($e \in [I_{bf}, I_{be}]$).

At the end of each iteration, the properties hold. From Property 5, we know that the front of the deque is the index whose corresponding score is the largest in the range. Therefore, for each I_f, we find $\max\{S[I_f] + S[j] \mid j \in [I_{bf}, I_{be}]\}$. Then iterating all I_f, we are able to get the constrained $\max\{S[i] + S[j]\} = \max\{\max\{S[I_f] + S[j] \mid j \in [I_{bf}, I_{be}]\} \mid \forall I_f\}$. □

2 Proof of time complexity

Theorem 2. *Algorithm* Xolik *can be finished in $O(n)$ time.*

Proof. At first, PopBack(), PushBack() and PopFront() of a deque are of constant time complexity.

Let n be the number of scored candidates. In Step 1 (Fig. 1 in the manuscript), the number of condition checks is at most $2 \times n_{be} + 2$ (two while loops), where n_{be} is the number of times that I_{be} decreases. Because I_{be} will at most decrease n times, after looping all I_f, Step 1 will cost a total of $2 \times n + 2 \times n = O(n)$. In Step 2, I_{bf} also monotonically decreases, and the number of condition checks at Line 18 is at most $n + n = O(n)$ in total.

The deque will at most have n elements pushed back, and thus the number of pop-backs is also at most n. The condition check at Line 19 will run at most $n + n$, which is the number of pop-backs and the number of times I_{bf} decreases. In Step 3, all operations are of constant time complexity. After iterating all I_f, the time complexity of Step 3 is $O(n)$.

Retrieving a cached score costs constant time, while computing a new score requires $O(\tilde{t})$. If the preprocessed XCorr (Eng *et al.*, 2008) is used, $\tilde{t}$ will be the average number of theoretical ions. Therefore, besides the operations in the matching step, i.e., the algorithm mentioned above, we need extra running time for computing new scores, which is $O(\tilde{t} \cdot p)$. Here p denotes the number of distinct peptides that are in valid cross-linked solutions. As $p \leq n$, the time complexity of the scoring is $O(\tilde{t} \cdot n)$.

Therefore, the whole algorithm including the scoring costs $O((1+\tilde{t}) \cdot n)$ time. Since $\tilde{t}$ depends on the database as well as the digestion setting, which is a constant, the whole algorithm will be finished in $O(n)$ time. □

3 Scoring peptides in the algorithm

In pLink (Yang *et al.*, 2012), Kojak (Hoopmann *et al.*, 2015) and Chen's speed-up implementation (Chen *et al.*, 2001), all single peptides are scored beforehand. However, scoring single peptides in advance is not compulsory in Xolik and can be relaxed through a simple modification. With this modification, Xolik reduces the amount of scoring in solving the cross-linked peptide identification problem.

Let us start from the analysis of the search space of cross-linked peptides. Let P be the list of candidate peptides, and it contains n peptides in total. Then the peptide-peptide combinations are

$$C = P \times P = \{(p_i, p_j) \mid p_i \in P, p_j \in P\}, \tag{1}$$

where p_i and p_j denote the ith and jth peptide, respectively. The size of C is n^2. After applying the constraint on the precursor mass (i.e., $|PM - (M[i] + M[j] + XM)| < \epsilon_1$),

the remaining combinations become

$$C' = \{(p_i, p_j) \mid (p_i, p_j) \in C, |PM - (M[i] + M[j] + XM)| < \epsilon_1\}, \tag{2}$$

where $M[i]$ and $M[j]$ denote the mass of the ith and jth peptide, respectively. It is easy to see that n^2 is the tight upper bound of the size of C'.

However, we do not need the whole P to generate C'. Instead, we need

$$P' = \bigcup_{(p_i, p_j) \in C'} \{p_i, p_j\}. \tag{3}$$

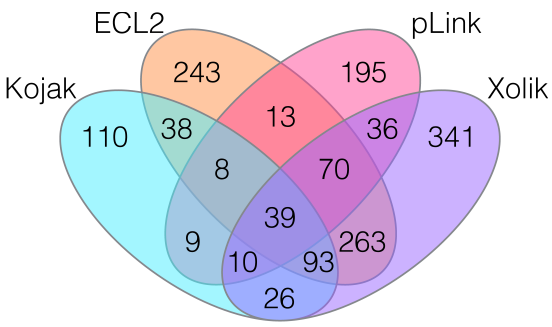
Obviously,

$$card(P') \leq card(P) = n. \tag{4}$$

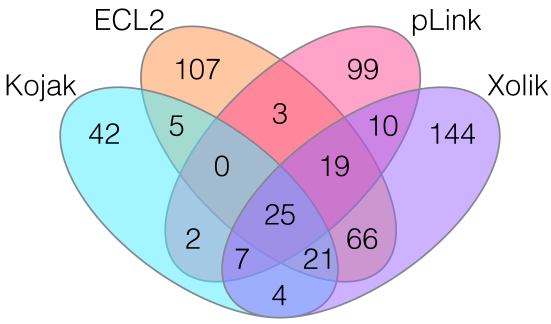
where $card()$ denotes the cardinality.

P' is a smaller peptide list to construct C', and therefore, only peptides in P' have to be scored in order to figure out the maximum paired score. If we score all single peptides beforehand and at the same time $card(P') \ll card(P)$, it means that we waste a lot of computation power on scoring those "useless" single peptides.

In Xolik, such a waste is avoidable. Instead of scoring all single peptides in advance before running the Xolik algorithm, the computation of scores is postponed until we really need the scores for comparison. When we retrieve a score, we ensure that the peptide corresponding to this score is one of the elements in a pair $(p_i, p_j) \in C'$, so that, at the same time, this peptide is an element in P'. The score of this peptide is then computed at the time that we retrieve it. Once the score is computed, it is cached for further use. Finally, only the peptides in P' are scored and scored once, and no computation power is wasted on scoring peptides in $P \setminus P'$.



(a) Venn diagram of PSMs.



(b) Venn diagram of non-redundant cross-linked peptides.

Figure 1: Venn diagrams of the identification results of ECL2, pLink, Kojak and Xolik using the *E. coli* Ribosome dataset under 5% FDR control. (a) shows the Venn diagram of reported PSMs, and (b) shows the Venn diagram of non-redundant cross-linked peptides. As shown in the Venn diagrams, the overlapping part between Xolik and ECL2 is much larger than others. This reflects that the only difference between Xolik and ECL2 is in the treatment of the precursor mass constraint.

4 Detailed comparison using the *E. coli* ribosome dataset

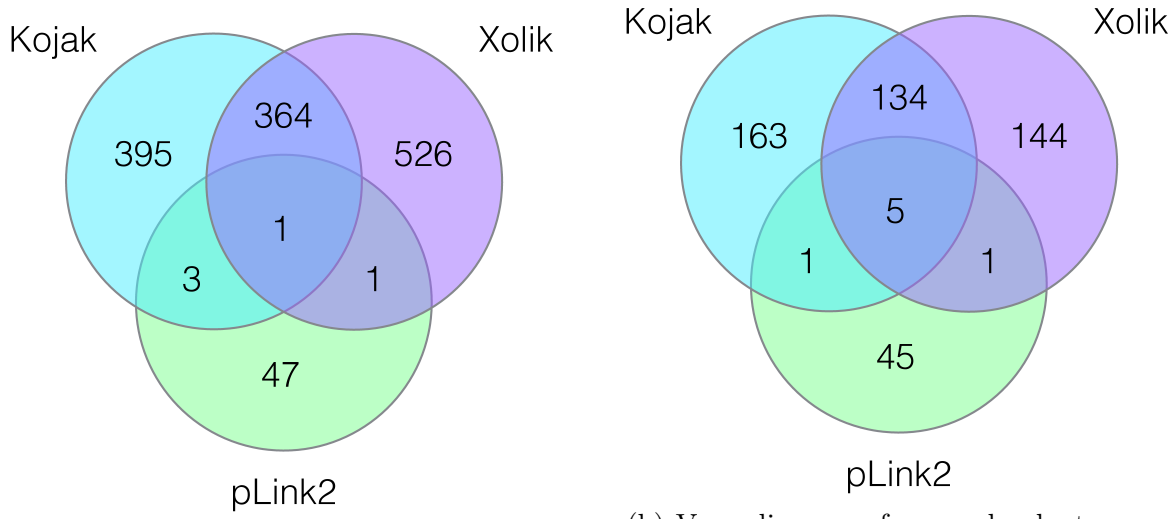
To examine how the identified PSMs are distributed, we compare the reported PSMs of all tools. A Venn diagram of the identified cross-linked PSMs is shown in Figure 1(a). More than half of the identifications reported by Xolik can be verified by other tools. Also, the overlap between Xolik and ECL2 is large, reflecting that the only difference between Xolik and ECL2 is in the treatment of the precursor mass constraint, which is the key difference between the pair matching algorithms in Xolik and ECL2. Because ECL2 doesn't strictly apply the precursor mass constraint, identifications outsides the MS1 tolerance range will be reported. These contribute a part of the difference. What's more, the occurrence of the identifications outside the MS1 tolerance range will affect the threshold determined by the FDR controlling algorithm. As a consequence, some identifications passing the threshold in Xolik may not pass the threshold in ECL2, and vice versa. These also contribute to the difference in the reported identifications between Xolik and ECL2.

There is a possibility that more than one spectrum corresponds to the same pair of peptides. To examine the ability of discovering cross-linked sites, we compare the non-

redundant cross-linked peptides identified by these tools. The Venn diagram is shown in Figure 1(b). The pattern of this Venn diagram is similar to the previous one. The diagram shows that Xolik identifies more unique cross-linked peptides than the other tools. Since most of the procedures are the same in ECL2 and Xolik, the performance of ECL2 on identifying cross-linked sites is close to that of Xolik.

5 Detailed comparison using the human protein dataset

Venn diagrams of the identification results using the human protein dataset are shown in Figure 2.



(a) Venn diagram of PSMs.

(b) Venn diagram of non-redundant cross-linked peptides.

Figure 2: Venn diagrams of the identification results of pLink2, Kojak and Xolik using the human protein dataset under 5% FDR control. (a) shows the Venn diagram of identified PSMs, and (b) shows the Venn diagram of non-redundant cross-linked peptides.

6 E-value estimation procedure

We follow the same procedure as ECL2 (Yu *et al.*, 2017) to estimate the E-value of the original XCorr score. The procedure is described as follows:

1. We collect 30000 scores of cross-linked peptides using the original MS1 tolerance.

2. If there are fewer than 30000 scores using the original MS1 tolerance, we enlarge the MS1 tolerance by 1 Da to collect more scores until 30000 scores are collected.

3. We build a score histogram and do a linear regression on the tail of the empirical log-survival function, using the same method as implemented in Comet (Eng *et al.*, 2013).

4. We use the fitted line to transform the XCorr score of the identified cross-linked peptides to the E-value, using the same procedure as in Comet.

7 Illustration of the entire search space

Here we would like to use an scatter plot to illustrate the necessity of searching the entire space exhaustively. In Figure 3, each dot in the plot denotes one cross-linked peptide pair identified by Xolik in the HeLa dataset (Section 3.4 in the main text). The vertical axis denotes the rank of the first single peptide identification, and the horizontal axis denotes the rank of the second single peptide identification. Here we arrange the single peptides in such an order that the first single peptide always ranks higher than the second single peptide, which explains why the dots only occupy the upper right triangular area. The three colored squares and the red rectangle denote the corresponding top N ranges of Kojak, pLink, Protein Prospector, and pLink2, respectively. It turns out that there are indeed some cross-linked peptides whose two single peptides are outside the top N ranges.

Different tools may use different scoring functions. Consequently, the ranks in other tools may not necessarily be the same as those in Xolik. When a PSM is outside the top N search space in Xolik, it is not necessarily outside the top N search space of other tools due to the difference in scoring functions. Figure 3 illustrates the importance of searching in the entire search space.

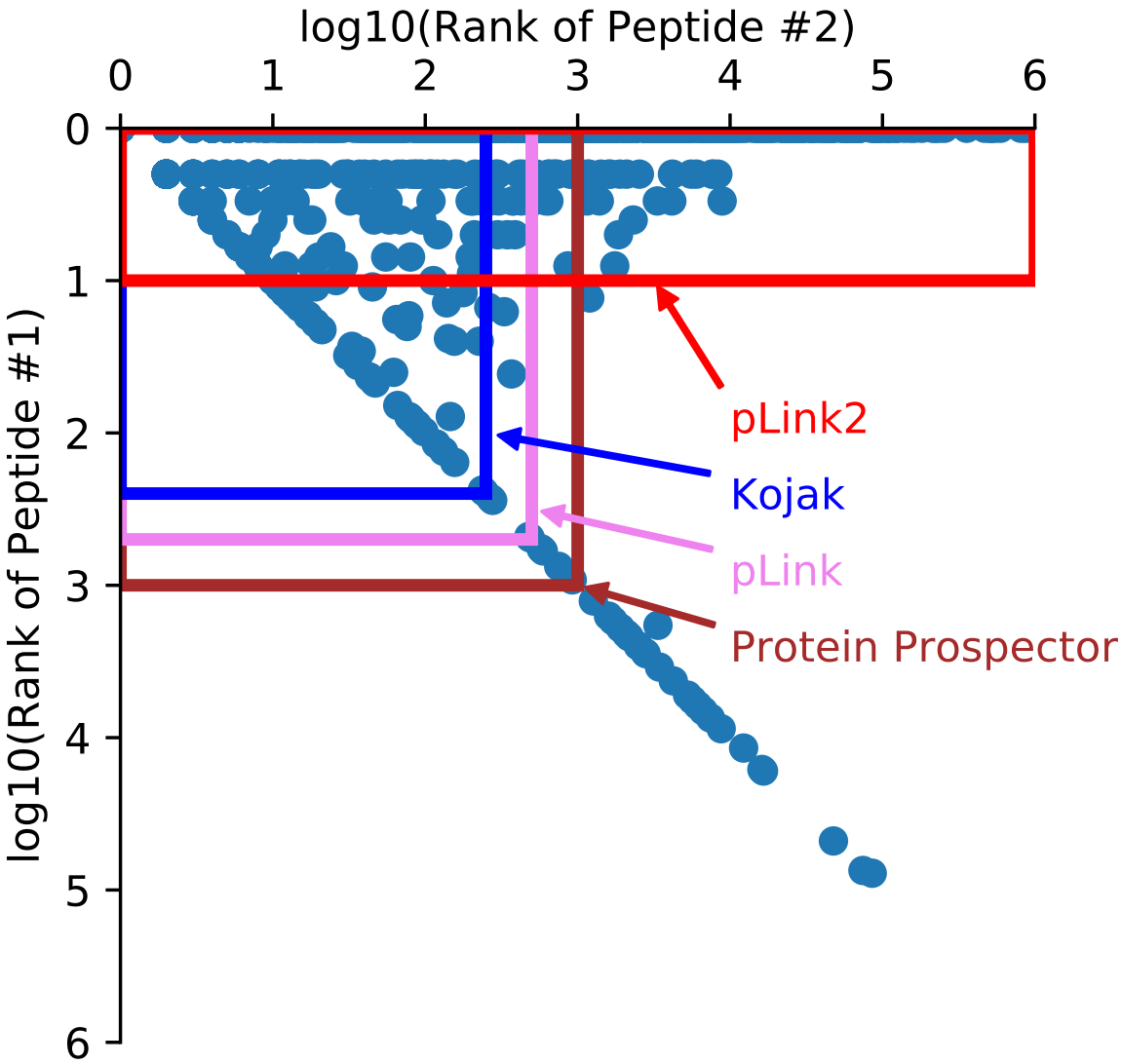


Figure 3: Illustration of the search space.

8 An example spectrum outside the search space of Protein Prospector

We provide a positive example to show the importance of searching the entire search space. We choose Protein Prospector as the example, because it searches both single peptides up to rank 1000 by default, which is the largest among the tools compared in Figure 3. This example spectrum can be confidently identified by Xolik but is outside the default search space of Protein Prospector (Trnka *et al.*, 2014), though the ranks of linear peptides in Xolik and those in Protein Prospector are not the same. This spectrum is with the scan number of 39917 in the data file xlink_MCM6_Asynch_rep2.mzXML in the HeLa dataset. Xolik assigns the sequence ALKTFVK(3)–DVEQQFKYTQPNICR(7) to this spectrum, with ranks 4212th and 1st, respectively. This sequence is consistent with the results reported by Makowski *et al.*, 2016.

Protein Prospector missed this spectrum using its default search space configuration, namely keeping the top 1000 hits of linear peptide candidates. When we enlarged Protein Prospectors search space to top 5000 linear peptide candidates, Protein Prospector can identify this spectrum with the same sequence with ranks 1533rd and 1st, respectively. It shows that the default search space of Protein Prospector may not be enough to identify all cross-linked peptides.

The search parameters and the annotation are shown in Figure 4. We accessed prospector.ucsf.edu on 25 May 2018 to analyze the above spectrum using Protein Prospector (v5.22.0). Here, we did not add the decoy sequences because Protein Prospectors decoy sequences are generated by random shuffling, which is different from Xolik. Even without decoy sequences, one of the ranks is outside the top 1000. It can be expected that the rank will be greater than the current one after appending the decoy sequences.

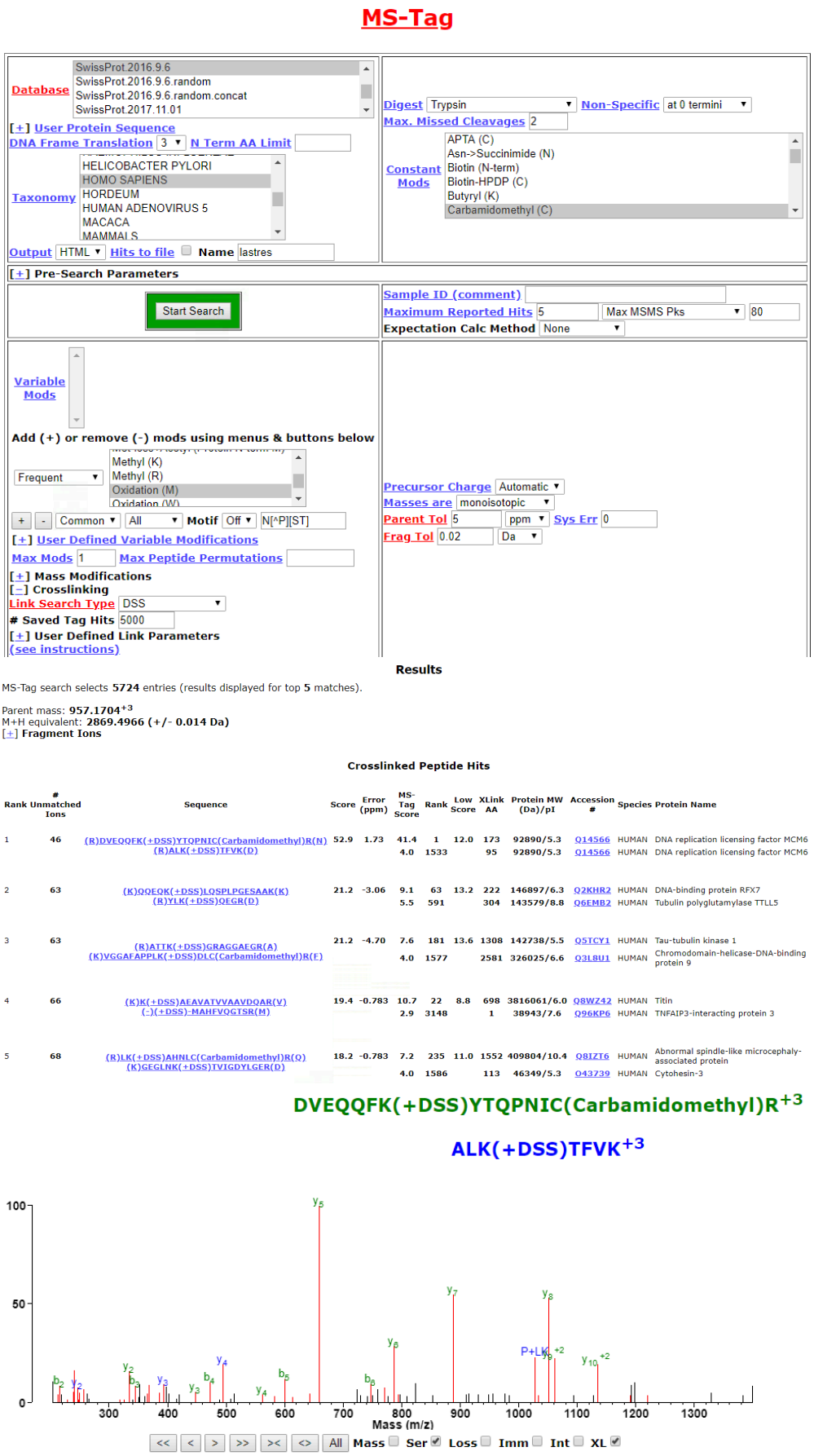


Figure 4: An example spectrum outside the search space of Protein Prospector.

References

Chen, T., Jaffe, J., and Church, G. (2001). Algorithms for identifying protein cross-links via tandem mass spectrometry. *Journal of Computational Biology*, **8**(6), 571–583.

Eng, J., Fischer, B., Grossmann, J., and MacCoss, M. (2008). A fast SEQUEST cross correlation algorithm. *Journal of Proteome Research*, **7**(10), 4598–4602.

Eng, J., Jahan, T., and Hoopmann, M. (2013). Comet: An open-source MS/MS sequence database search tool. *Proteomics*, **13**(1), 22–24.

Hoopmann, M., Zelter, A., Johnson, R., Riffle, M., Maccoss, M., Davis, T., and Moritz, R. (2015). Kojak: Efficient analysis of chemically cross-linked protein complexes. *Journal of Proteome Research*, **14**(5), 2190–2198.

Makowski, M., Willems, E., Jansen, P., and Vermeulen, M. (2016). Cross-linking immunoprecipitation-MS (xIP-MS): Topological analysis of chromatinassociated protein complexes using single affinity purification. *Molecular and Cellular Proteomics*, **15**(3), 854–865.

Trnka, M., Baker, P., Robinson, P., Burlingame, A., and Chalkley, R. (2014). Matching cross-linked peptide spectra: Only as good as the worse identification. *Molecular and Cellular Proteomics*, **13**(2), 420–434.

Yang, B., Wu, Y.-J., Zhu, M., Fan, S.-B., Lin, J., Zhang, K., Li, S., Chi, H., Li, Y.-X., Chen, H.-F., Luo, S.-K., Ding, Y.-H., Wang, L.-H., Hao, Z., Xiu, L.-Y., Chen, S., Ye, K., He, S.-M., and Dong, M.-Q. (2012). Identification of cross-linked peptides from complex samples. *Nature Methods*, **9**(9), 904–906.

Yu, F., Li, N., and Yu, W. (2017). Exhaustively identifying cross-linked peptides with a linear computational complexity. *Journal of Proteome Research*, **16**(10), 3942–3952.