A graph-based approach for modification site assignment in proteomics

Dafni Skiadopoulou[1,2] Harald Barsnes[2] Lukas Käll[3] Marc Vaudel[1,2,4]

[1]Mohn Center for Diabetes Precision Medicine, Department of Clinical Science, University of Bergen, Bergen, Norway
[2]Computational Biology Unit, Department of Informatics, University of Bergen, Bergen, Norway
[3]Science for Life Laboratory, School of Engineering Sciences in Chemistry, Biotechnology and Health, KTH Royal Institute of Technology, Stockholm, Sweden
[4]Department of Genetics and Bioinformatics, Health Data and Digitalization, Norwegian Institute of Public Health, Oslo, Norway

Abstract

Summary: Assigning protein post-translational modifications to acceptor sites requires the distribution of modifications in a way that maximizes localization scores while avoiding chemically impossible configurations. We provide an efficient graph-based approach to this problem implemented both as standalone implementation and integrated in the PeptideShaker software tool.
Availability and Implementation: An open source implementation in Python is available at https://github.com/ProGe
modifications-matching under a GPL-3.0 license. An open source implementation in Java is available at github.com/compomics/compomics-utilities under an Apache-2.0 license.

1 Introduction

In mass spectrometry-based proteomics, mass spectra of fragmented peptides are matched against a database of protein sequences using a proteomic search engine, allowing the high throughput identification of peptide sequences together with their modification status [1]. The localization of post-translational modifications (PTMs) on protein sequences is an important functional information [2]. Furthermore, since amino acid substitutions can have the same mass difference as a modification, an incorrectly localized modification can be misinterpreted, yielding a false variant call [3]. To evaluate the localization of modifications on a peptide sequence, multiple localization scores were developed that estimate the likelihood for a given acceptor site to be occupied for every modification of every peptide to spectrum match (PSM) [4, 5].

Once the localization scores have been computed, the modifications are assigned to the sites of most likely localization before further processing, e.g. error rate estimation using Percolator [6]. It is important to note that due to differences in scoring, and especially since search engines often only consider the most intense peaks in mass spectra, the peptide maximizing the modification localization scores is not necessarily the best scoring peptide in the search engine results. And since the search engines also often have limitations in terms of how many modification sites combinations they consider, the peptide maximizing localization scores might not even be in the list of peptide candidates returned by the search engine.

In order to find the peptide maximizing the modification localization scores, the modifications found by the search engine need to be assigned to the site with highest localization score without yielding a configuration that is not chemically possible: modifications have different types of acceptor sites, e.g. amino acids or termini, and cannot be stacked on the same acceptor site. While maximizing the scores is trivial with a couple of modifications, e.g. phosphorylation and oxidation, the problem becomes more complex when combining many modifications with possible conflicting sites, e.g. in the study of histones

[7]. Furthermore, since this needs to be conducted on millions PSMs per experiment, the site assignment needs to be fast.

Here, we propose a graph-based approach that models modifications and their acceptor sites, and returns a configuration that maximizes localization scores with controlled processing time. The modification assignment problem is reduced to the minimum weight matching problem in bipartite graphs, by modelling modifications and acceptor sites as vertices, connected by edges whose weights are calculated based on the corresponding modification localization scores. We provide a standalone Python implementation as well as a Java implementation in the compomics-utilities library [8] and demonstrate its usage in PeptideShaker [9].

2 Methods

2.1 Model

In this work the modification site localization problem is addressed by a graph approach. For each PSM, a weighted bipartite graph ($G = \{V, E, w\}$) is used to model all possible combinations of modifications on the amino acid chain. In this graph we determine two kinds of vertices, the ones that represent the different modifications (i.e. $D = \{d_1, d_2, \ldots, d_k\}$) and the ones that represent all their possible acceptor sites (i.e. $A = \{a_1, a_2, \ldots, a_n\}$). The set of vertices V is then formed by the union of the sets A and D (i.e. $V = A \cup D$). For each modification, an edge is formed between the corresponding vertex and each possible acceptor site. A weight $w_{i,j}$ is assigned to each edge $e_{i,j}$ corresponding to the localization score of the modification d_i to the acceptor site a_j.

Using this model we reduce the modification site assignment problem to the maximum

Minimum or maximum?

weight maximum cardinality matching problem on the resulting bipartite graph. The solution of this problem will be a set of pairwise non-adjacent edges satisfying two conditions. The first one is that the matching will be of maximum cardinality which in this modeling ensures that all the vertices that represent a modification will be matched. The second condition is that the weight of the matching is the maximum which means that it provides the combination of modifications and acceptor sites that give the maximum localization score in total.

2.2 Implementation

We provide an open source Python-based implementation of our approach available under a GPL-3.0 license (github.com/ProGenNo/peptides-modifications-matching). We used the networkX library (networkx.org) implementation of Karp's algorithm for the rectangular linear assignment problem [10]. Since this algorithm minimizes the total weight of the resulting matching the weights of the edges from the model described above were transformed accordingly so as the maximum localization score corresponded to the minimum edge weight. The script of the implementation receives as input a text file where a description of the available PSMs is included. It requires one line per PSM with the information of the spectrum and peptide IDs as well as the modifications with their possible acceptor sites and the corresponding localization scores. Further information are available at the github repository.

We also provide an open source Java-based implementation freely available as part of the compomics-utilities library [8] under an Apache-2.0 license. The implementation is located in the *com.compomics.util.experiment.identi* package and it makes use of the JGraphT package [11] with a similar model transformation as ~~in the case above~~ for the python implementation. The implementation in compomics-utilities is readily usable *via* the PeptideShaker tool [9]. We have cited PeptideShaker in the intro as well, is the citation here required? same for compomics on this paragraph

TODO Marc: Write wiki page in compomics-utilities

2.3 Histone example

A dataset acquired on a synthetic histone peptide with different combinations of modifications was searched using MS-Amanda

PMID 33759252

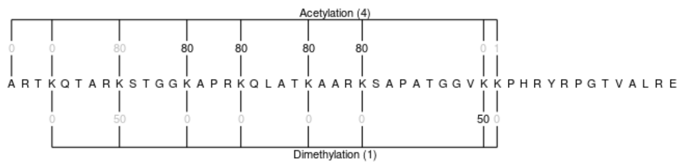
Acetylation (4)
0 0 80 80 80 80 80 0 1
A R T K Q T A R K S T G G K A P R K Q L A T K A A R K S A P A T G G V K K P H R Y R P G T V A L R E
0 50 0 0 0 0 50 0
Dimethylation (1)

Figure 1: Graph representation of a peptide with four acetylations and one dimethylation with all their possible acceptor sites and the respective localization scores.

using SearchGUI

PMID 29774740

. The result file of MS-Amanda was processed using PeptideShaker where all modifications were scored using PhosphoRS

PMID 22073976

.

TODO Marc: Check with Veit

2.4 Performance benchmark

We generated sets of peptides of length 30 with multiple random sets of modifications. For each peptide, the number of distinct modifications was randomly generated between one and ten. For every modification, up to six modification sites were randomly selected, and the number of modified residues was randomly set between one and the number of modification sites. Finally, modification localization scores were given randomly for all modification sites and the matching of modification to site was conducted using the Java implementation. The code was run for different batches of peptides using different number of threads on a laptop computer, and the processing time was recorded.

Subsequently, we generated peptides in a similar fashion, but fixed the number of distinct modifications, modification sites, and modified residues. We then benchmarked the processing time for all possible combinations using 1,000 peptides per thread and all threads available.

Each benchmark was replicated ten times. The code used to benchmark the application is available in the *com.compomics.util.experiment.identification.modification.peptide_mapping.performance* package of the compomics-utilities library.

3 Results

3.1 Histone modification localization example

An example of a graph model for a modified peptide is presented in Fig. 1, where one site was reported by the search engine to carry a dimethylation and four were found to carry an acetylation. The maximum weight matching then assigned the modifications to the best scoring residues while avoiding to put the dimethylation on the same amino acid as an acetylation. Note that our approach can be applied to peptides carrying any combination of modification identified by proteomic search engines.

TODO

A word on conflict resolution and ”corner cases”?

3.2 Performance

Benchmarking the performance of the matching algorithm demonstrated that around one million peptides could be processed per minute and per thread. Since the peptides are scored individually, they can be processed in parallel and as soon as approximately 1,000 peptides were available, the processing time increased linearly with the number of peptides and the a minor increase in processing time was observed when using multiple threads, which we impute to Java having to manage many more objects (Figure 2). For a given peptide, the modification matching time increased with the number of different modifications, the number of sites, and the number of modified residues (Figure 3). In all configurations, the modification

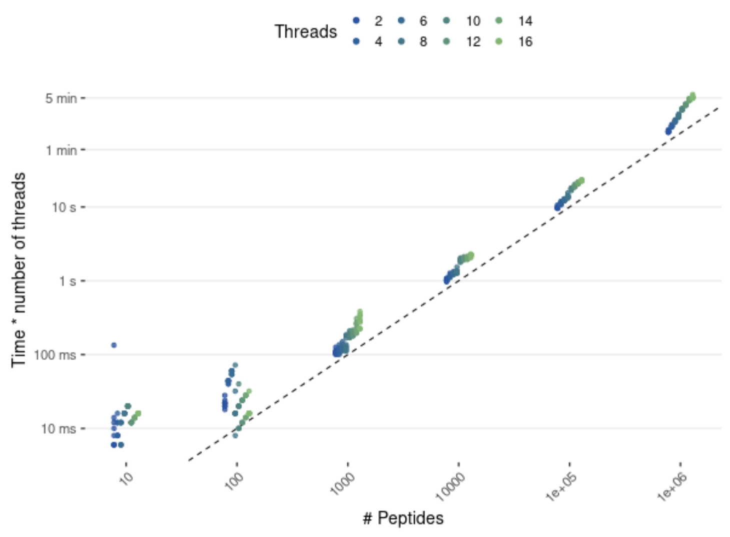

Figure 2: Processing time multiplied by the number of thread plotted against the number of peptides for benchmark runs of equal numbers of peptides. Note that log scales are used on both axes. Runs with different number of threads are jittered from left to right and plotted with different colors.

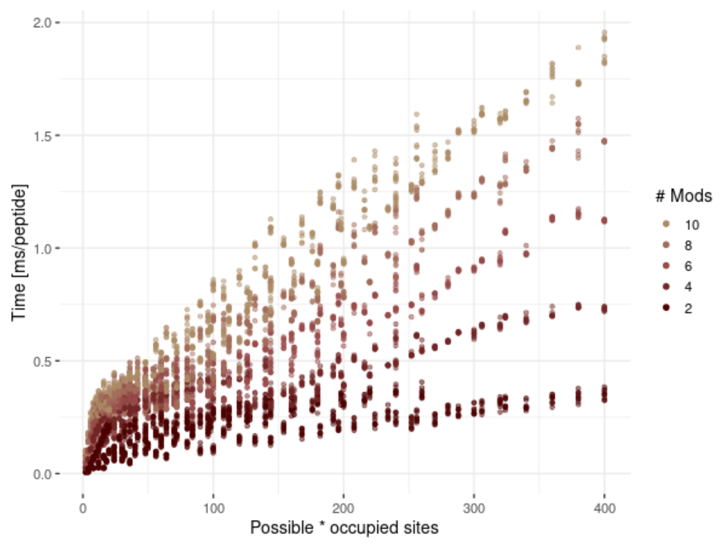

Figure 3: For different baches of 1,000 peptides per thread using 16 threads, processing time per peptide against number of sites multiplied by number of modified residues. Peptides with different number of distinct modifications are colored differently.

site assignment took only few seconds for thousands of peptides, which is negligible compared to the search time, which can be of several hours, as was the case for the Histone example.

4 Conclusion

5 Acknowledgements

This work was supported by the Research Council of Norway (project #301178 to MV) and the University of Bergen.

This research was funded, in whole or in part, by the Research Council of Norway 301178. A CC BY or equivalent license is applied to any Author Accepted Manuscript (AAM) version arising from this

submission, in accordance with the grant's open access conditions.

References

[1] R. Aebersold and M. Mann, "Mass spectrometry-based proteomics," *Nature*, vol. 422, pp. 198–207, Mar 2003.

[2] S. Loroch, C. Dickhut, R. P. Zahedi, and A. Sickmann, "Phosphoproteomics–more than meets the eye," *Electrophoresis*, vol. 34, pp. 1483–1492, May 2013.

[3] M. K. Bunger, B. J. Cargile, J. R. Sevinsky, E. Deyanova, N. A. Yates, R. C. Hendrickson, and J. L. Stephenson, "Detection and validation of non-synonymous coding snps from orthogonal analysis of shotgun proteomics data," *Journal of Proteome Research*, vol. 6, no. 6, pp. 2331–2340, 2007. PMID: 17488105.

[4] R. J. Chalkley and K. R. Clauser, "Modification site localization scoring: strategies and performance," *Mol Cell Proteomics*, vol. 11, pp. 3–14, Feb. 2012.

[5] D. C. H. Lee, A. R. Jones, and S. J. Hubbard, "Computational phosphoproteomics: From identification to localization," *PROTEOMICS*, vol. 15, no. 5-6, pp. 950–963, 2015.

[6] M. The, M. J. MacCoss, W. S. Noble, and L. Käll, "Fast and accurate protein false discovery rates on large-scale proteomics data sets with percolator 3.0," *Journal of The American Society for Mass Spectrometry*, vol. 27, pp. 1719–1727, Nov 2016.

[7] H. Huang, S. Lin, B. A. Garcia, and Y. Zhao, "Quantitative proteomic analysis of histone modifications," *Chemical Reviews*, vol. 115, pp. 2376–2418, Mar 2015.

[8] H. Barsnes, M. Vaudel, N. Colaert, K. Helsens, A. Sickmann, F. S. Berven, and L. Martens, "compomics-utilities: an open-source java library for computational proteomics," *BMC Bioinformatics*, vol. 12, p. 70, Mar 2011.

[9] M. Vaudel, J. M. Burkhart, R. P. Zahedi, E. Oveland, F. S. Berven, A. Sickmann, L. Martens, and H. Barsnes, "Peptideshaker enables reanalysis of ms-derived proteomics data sets," *Nature Biotechnology*, vol. 33, pp. 22–24, Jan 2015.

[10] R. M. Karp, "An algorithm to solve the m $\times$ n assignment problem in expected time o(mn log n)," *Networks*, vol. 10, no. 2, pp. 143–152, 1980.

[11] D. Michail, J. Kinable, B. Naveh, and J. V. Sichi, "Jgrapht—a java library for graph data structures and algorithms," *ACM Trans. Math. Softw.*, vol. 46, may 2020.