Unipept 6.0: Fast Semi-exact Peptide Matching With a Memory Conservative Index for UniProtKB

Bram Devlaminck

Abstract

Unipept is an ecosystem of software tools developed for fast metaproteomics data analysis. In the past few years, Unipept has been optimized for tryptic peptide search, with an inefficient workaround for peptides with missed cleavages. However, newer fields such as immunopeptidomics work with non-tryptic peptides. Unipept 6.0 introduces a new index structure that allows ultra-fast peptide matching for arbitrary peptides. This makes Unipept applicable in domains where trypsin is not the protease of choice, and also removes the performance penalty when dealing with missed cleavages. With these advancements, Unipept 6.0 is 10 to 100 times faster than Unipept 5.x when handling tryptic peptides with missed cleavages. In addition to these advancements, Unipept 6.0 maintains a superior performance compared to similar tools, ensuring efficient and accurate metaproteomics data analysis.

1 Introduction

Metaproteomics research has been invaluable in gaining a better understanding of the function of each organism in a microbiome ecosystem [1]. This started with small-scale experiments from as little as three proteins [2], but has evolved over the years to experiments with more than 100 000 peptides [3], [4]. Traditionally, trypsin is the most used protease in the (meta)proteomics field. Unipept [5]–[12], one of the first major tools to support metaproteomics data analysis, was specifically developed with this in mind. As a result, Unipept uses an index structure where the UniProtKB [13] database is preprocessed by splitting every protein according to the cleavage pattern created by trypsin. This allows Unipept to precompute the Lowest Common Ancestor (LCA) for every tryptic peptide digestion from UniProtKB. This approach has proven its efficiency in the past few years, but also has its limitations.

From time to time, trypsin will miss a cleavage site, creating a so-called missed cleavage. Since Unipept assumes there are no missed cleavages, peptides with such missed cleavages will not be present in the current Unipept index. To solve this problem, a solution was retro-fitted to handle these missed cleavages. When this option is enabled, Unipept scans the input peptides for missed cleavage positions, splits the peptides and perform separate lookups for every tryptic fragment. The resulting sets of protein matches are intersected, which delivers a set of possible protein matches. Every protein in this set is scanned left-to-right to ensure that the original peptide is present, and not only its separate tryptic fragments. Finally, the LCA of the found proteins is calculated. All these extra steps, including the on-the-fly calculation of the LCA, have a significant impact on performance.

A second disadvantage of the current Unipept index, is the inability to process non-tryptic peptides. Similar to peptides with missed cleavages, these are not present in the index and no efficient workaround exists for this problem.

This restriction was initially not a significant issue since Unipept was originally created for metaproteomics, where trypsin is widely used. However, other fields such as immunopeptidomics [14] require analysis tools to be able to cope with non-tryptic peptides. Upstream analysis software such as MS^{2}Rescore [15] and MS^{2}PIP [16] have recently been adding support for immunopeptidomics research. By adding support for arbitrary peptide matching, Unipept will also automatically support these newer research fields. As an additional benefit, support for arbitrary peptide matching also removes the performance penalty associated with handling missed cleavages. This requires a new index structure with the following properties:

1. For every peptide (tryptic or non-tryptic), all protein matches in UniProtKB should be found.
2. The maximum memory usage when building the index for the complete UniProtKB database should be around 1 to 2 TB due to current hardware limitations.

3. Search performance should be on par with Unipept 5.x for tryptic peptides.

4. The new index should facilitate all Unipept 5.x analyses.

5. Isoleucine and leucine should be equatable during search.

In short, we want an index that does not remove any of the features from Unipept 5.x, while eliminating the restriction that only tryptic peptides can be found efficiently.

2 Methods

At the core of our new index structure lies a suffix array. This suffix array is a data structure that can be built using the libdivsufsort [17] or libsais [18] library. Both provide a linear time algorithm with a $5n + O(1)$ memory complexity, where n is the size of the text that needs to be indexed. Depending on the used text, one implementation is faster than the other.

2.1 Sparse Suffix Arrays

While a complete suffix array delivers the best performance, the index itself is large. The size of the suffix array can be reduced by introducing a sampling step to create a so-called sparse suffix array (SSA). This variant of a suffix array only stores every k-th suffix of the input text. This results in an SSA which is only $\frac{1}{k}$ of the original suffix array size. The disadvantage of using an SSA is that it becomes impossible to search for peptides having less than k amino acids, and that searching in general becomes slower. This trade-off does not introduce any problem, since Unipept already limits the search to peptides with 5 or more amino acids. This restriction was introduced since mass spectrometers can't read peptides with less than 5 amino acids. Furthermore, such extremely short peptides occur in a lot of proteins. Subsequently, this yields extremely generic functional and taxonomic analysis results, which is not very informative.

2.2 Equating Isoleucine and Leucine

Our solution to equate isoleucine and leucine consists of two parts. Firstly, we don't index the original text, but a slightly modified version. We replace every occurrence of leucine by isoleucine, which essentially equates them during construction of the suffix array. During search, we perform the same translation on every peptide. As a result, we find every match with I and L equated. This search is performed using the original, unmodified text. This is important, since the original text is needed in an additional second step if we don't want to equate I and L.

In this second step, we filter away unwanted matches from the first step. This is achieved by checking every I and L location in the original, unmodified peptide with the corresponding amino acid in the original, unmodified text. A mismatch indicates that I and L were wrongfully equated. Only the I and L locations have to be checked since the suffix array search already ensures that all other characters match in the original text.

2.3 Analyses

Our solution performs the functional and taxonomic analysis at runtime. While it would be advantageous to precompute this, the suffix array and text do not contain enough information to do this. This is a trade-off we made during development between optimal memory usage and speed.

2.4 Maximum Matches

A single peptide can occur in thousands of proteins. Because of this, performing the analyses at runtime can become slow. Furthermore, the analyses for such peptides yields a generic result in most cases. More precisely, earlier research by the Unipept team [19] has shown that out the 1.3 billion tryptic peptides in UniProtKB 2023_03, only around 13 000 had more than 10 000 protein matches. From these 13 000 peptides, 95% of the peptides had `root` as LCA. This allows, with minimal loss of information, to assume that every peptide with more than 10 000 matches has `root` as LCA. Additionally, this limits the amount of information returned per query.

3 Results

Three main factors are considered during testing. The memory usage and speed during construction, the resulting index size while hosting, and search performance.

3.1 Constructing the Index

Constructing a suffix array for UniProtKB 2024_01 requires around 735 GB of RAM and 5 hours of computing time using the libdivsufsort library. Because of the high memory needs, we use the HPC of Ghent University, where the high-memory cluster has 16 nodes with each 2×64-core AMD EPYC 7773X (Milan-X @ 2.2 GHz) and 940 GiB of RAM. We use one core of a single node, since the construction phase is single threaded. The resulting suffix array is around 700 GB in size (+ an additional 88

GB for the text). Our goal is to host this new index on the Unipept servers, which have around 0.5 TB of memory available. To make this possible, we immediately perform a sampling phase with sparseness factor $k = 3$ at the end of the construction process on the HPC. This results in a reduced index size of $\frac{700}{3} + 88 \approx 322$ GB. Next to this peptide search index, Unipept needs extra information (such as the UniProt accession number, NCBI taxon ID and functional annotations per protein) to perform the provided analyses. This results in another 25 GB of RAM needed. Figure 1 visualizes the size of each component of the resulting index.

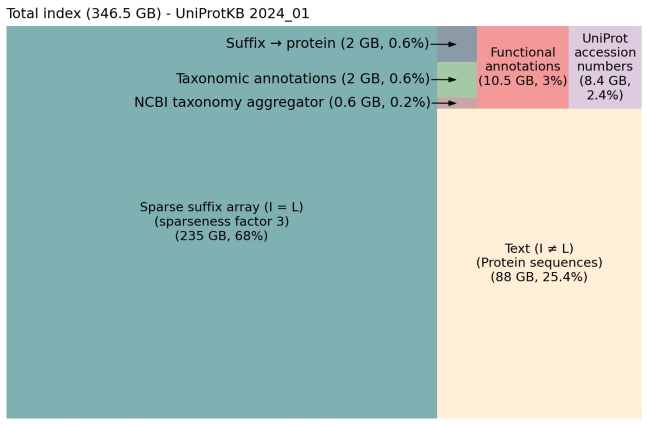

Figure 1: Visualisation of the size of each component of Unipept 6.x index for UniProtKB 2024_01.

3.2 Querying the Index

Unipept 6.x supports two types of search. Most importantly, we can efficiently search all matches and perform the taxonomic and functional aggregation at runtime. Since the analysis is executed at runtime, we already have all the information per matched protein. With Unipept 6.x we provide search without aggregation as a separate option. This was not possible with Unipept 5.x since the aggregations were precomputed and retrieving all information from disk per protein was too slow. Both search options in Unipept 6.x perform similar. Albeit that the first option is slightly faster. This is caused by the additional information per protein that needs to be serialized, since the functional annotations are not aggregated. In this section we only discuss the search with analyses, since this is what Unipept 5.x supports and the other search option performs very similar.

Missed cleavages To evaluate the peptide search performance we use 6 files containing around 25 000 peptides each. These are samples from the SIHUMIx experiment [20], [21], where trypsin is used as a protease. This means that as well as tryptic peptides, there are also peptides present with naturally occurring missed cleavages. Figure 2 shows the execution time for both Unipept 6.x and 5.x. To make the comparison as fair as possible, the *filter duplicate peptides* setting is turned off, and *advanced missed cleavage handling* is turned on. This ensures that both indices search every peptide from the input file (including duplicates and peptides with missed cleavages). Unipept 6.x is 10 to 100 times faster, which clearly removes the performance penalty that was currently associated with handling missed cleavages. Note that the Unipept 5.x index will not find peptides resulting from using a different protease, whereas this makes no difference for the Unipept 6.x index.

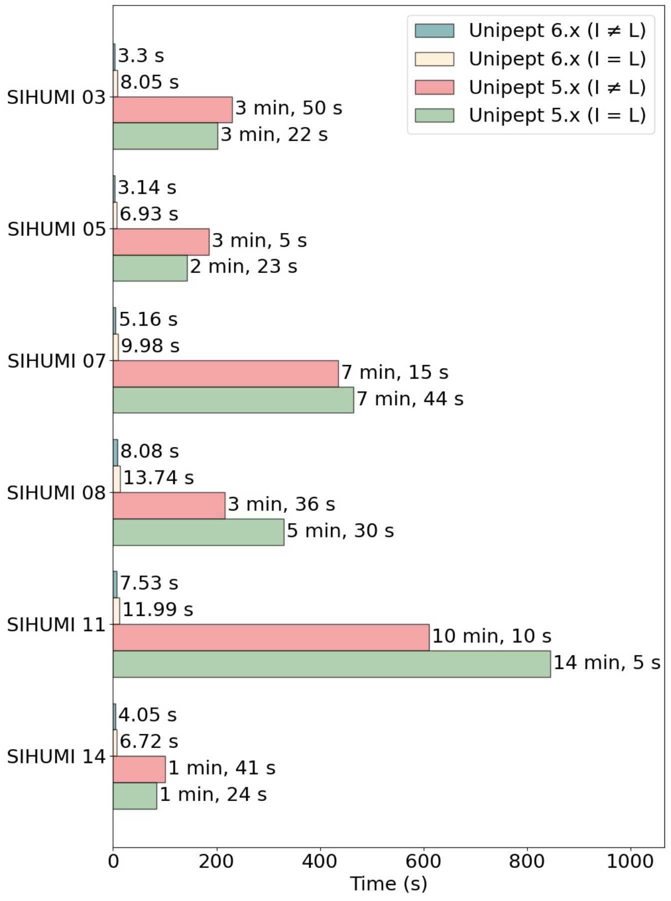

Figure 2: Execution time for Unipept 6.x and Unipept 5.x while processing different SIHUMIx samples. Each sample consists of approximately 25 000 peptides. Unipept 5.x searches with the *filter duplicate peptides* setting turned off, and *advanced missed cleavage handling* turned on. This way both indices search all peptides (included duplicates), and missed cleavages are handled by both. The used test files can be found in our GitHub repository https://github.com/BramDevlaminck/Thesis_benchmarkdata/tree/master/SIHUMI

Tryptic peptides A second interesting case to consider is when searching strictly tryptic peptides. In that case, missed cleavage handling can be disabled in Unipept 5.x, which significantly improves the performance. Figure 3 visualizes the processing time for 100 000 tryptic peptides. It is clear that the performance is comparable when I and L are equated, while Unipept 5.x is around a third faster than Unipept 6.x when I and L are not equated.

Furthermore, it is also clear that the extra filtering step when I $\neq$ L, has a performance impact in the new index. This effect was not visible in Figure 2, where the reverse seems to apply. However, this reverse effect can be explained by the fact that searching with I = L results in more matches, which requires more output to be serialized to a JSON file. When searching for larger files, there is proportionally more time spent in the search phase, which compensates for the longer serialization time.

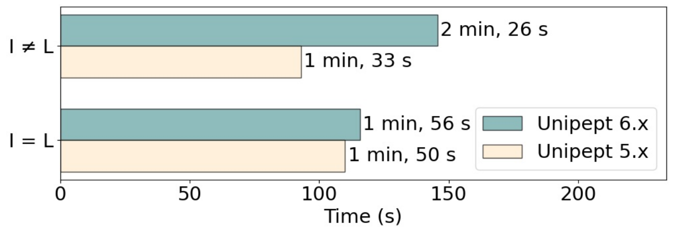

Figure 3: Processing time for 100 000 tryptic peptides for Unipept 6.x and 5.x. This is the best case scenario for Unipept 5.x where missed cleavages are **not** handled.

4 Other Tools

Tools such as the Uniprot peptide search tool [22], [23], the Expasy ScanProsite tool [24] and Unipept 5.x all have the possibility to find matches in the UniProtKB database. In this section we compare the performance and feature sets of alternative tools with Unipept 6.x. Because of the poor performance of some of these tools, we limited testing to the randomly chosen tryptic peptide `ISPAVLFVIVILAVLFFISGLLHLLVR`. Unipept 6.x uses above settings and finds all matches for the test peptide in less than 5 ms.

In terms of features, the UniProt peptide search tool is identical to Unipept 6.x. They both find all matches in UniProtKB, and have the option to equate I and L. The only difference is their performance. Searching the test peptide in Unipept 6.x only takes a few milliseconds, whereas the UniProt tool takes a few seconds up to multiple minutes.

The Expasy ScanProsite tool takes another approach. They provide a wide range of options for inexact matching. They call these motifs, and these are comparable to regular expressions where the user can use wildcards, character classes and even negations. The other major difference is that this tool does not use the whole UniProtKB database. Only the proteins that are part of a reference genome are indexed, which means only around one third of the complete UniProtKB database is indexed. Lastly, searching the test peptide takes around 5 minutes. As expected, since not the whole UniProtKB database is indexed, only a subset of the matches is found.

The last major tool we compare with, is Unipept 5.x. Unipept 6.x supports all options provided by Unipept 5.x, while removing the restriction that only tryptic peptides can be searched. Searching the test peptide took less than 5 ms, and found the same matches because our test peptide is tryptic. Table 1 gives an overview of the described differences between the tools.

	UP6	UPS	ESP	UP5
Used proteins	all	all	ref. prot.	all
Approx. match	[IL]	[IL]	flexible	[IL]
Time	< 5 ms	1 s - 20 min	5 min	< 5 ms
Searchable pept.	all	all	all	$\sim$ tryptic

Table 1: Comparison of Unipept 6.x (UP6), UniProt peptide search (UPS) tool, Expasy ScanProsite (ESP) tool and Unipept 5.x (UP5). `[IL]` in the approximate matching row means that only I and L can be equated, while $\sim$ `tryptic` in the searchable peptide row means that only tryptic peptides or tryptic peptides with missed cleavages can be found.

5 Conclusion

With a new index structure at the heart of Unipept 6.x, we have broadened Unipept's possible use cases. The new index structure makes searching peptides with missed cleavages around 10 to 100 times faster, while adding the possibility to search arbitrary peptides, regardless of the used protease, without any performance penalty. In our benchmarks, Unipept 6.x strongly outperforms its closest competitors at finding all occurrences for an arbitrary peptide longer than 2 amino acids in UniProtKB.

Our new index is optimized for searches with leucine and isoleucine equated to each other. This corresponds to the default search configuration of Unipept. However, searching with I $\neq$ L only introduces a small performance penalty. For smaller batches of peptides, the time needed to serialize the extra matches even outweighs the extra time spent during the search phase itself.

Another advantage of the new index is that there are no extra steps required to retrieve the NCBI taxon ID for each matched protein. Extra steps are required in Unipept 5.x to retrieve all the individual taxon IDs, with a significant impact on the performance. This restriction created a bottleneck in the new Peptonizer2000 tool [25] that has now been removed.

The main disadvantage of Unipept 6.x compared to Unipept 5.x is that searching tryptic peptides is slightly slower than before, especially when searching with I $\neq$ L. This slowdown is introduced by taxonomic analyses performed at runtime, whereas these could all be precalculated with the old index that was restricted to tryptic peptides. However, this slowdown is limited and acceptable. Especially when keeping in mind that samples from experiments always have some missed cleavages, which would not have been found before, without a performance hit.

6 Future Work

The Unipept 6.x index meets the pre-established requirements, but still leaves room for improvement in several areas.

A significant part of the current computation time is invested in performing the taxonomic and functional analyses at runtime. Modifying the new index to support precalculation of these analyses, while still maintaining a peak memory usage which is manageable, would drastically improve the performance. Enhanced suffix arrays (ESA) could facilitate this, but require more memory.

Another area of improvement is the index size itself. The index size could be reduced even more by making the text and suffix array more compact. Both of these components don't utilize all the bits from the allocated bytes. The suffix array only needs 37 of the 64 bits used by every entry, while the text only needs 5 bits out of each byte per character. This could reduce the total index size from 346 GB to around 215 GB. However, this would introduce extra steps to decode every access to any of these data structures. This could introduce a non-negligible performance penalty.

Another option to reduce the index size is by switching to a completely different index structure. Both the FM-index [26] and R-index [27] have shown promising results in this respect.

Lastly, Unipept does not perform any form of inexact matching, except for equating I and L. Introducing inexact matching into Unipept could allow us to deal with small biological mutations or variations introduced during the transformation from mass spectrum to peptide. Possible interesting routes to explore are regex matching using suffix arrays [28] or inexact matching using bidirectional FM-indices [29].

References

[1] T. Van Den Bossche, M. Ø. Arntzen, D. Becher, *et al.*, "The Metaproteomics Initiative: a coordinated approach for propelling the functional characterization of microbiomes," *Microbiome*, vol. 9, no. 1, p. 243, Dec. 20, 2021, ISSN: 2049-2618. DOI: 10.1186/s40168-021-01176-w. [Online]. Available: https://doi.org/10.1186/s40168-021-01176-w.

[2] R. J. Ram, N. C. VerBerkmoes, M. P. Thelen, *et al.*, "Community Proteomics of a Natural Microbial Biofilm," *Science*, vol. 308, no. 5730, pp. 1915–1920, 2005. DOI: 10.1126/science.1109070. eprint: https://www.science.org/doi/pdf/10.1126/science.1109070. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1109070.

[3] X. Zhang and D. Figeys, "Perspective and Guidelines for Metaproteomics in Microbiome Studies," *Journal of Proteome Research*, vol. 18, no. 6, pp. 2370–2380, 2019, PMID: 31009573. DOI: 10.1021/acs.jproteome.9b00054. eprint: https://doi.org/10.1021/acs.jproteome.9b00054. [Online]. Available: https://doi.org/10.1021/acs.jproteome.9b00054.

[4] T. Van Den Bossche, B. J. Kunath, K. Schallert, *et al.*, "Critical assessment of MetaProteome investigation (CAMPI): A multi-laboratory comparison of established workflows," *Nature Communications*, vol. 12, no. 1, p. 7305, Dec. 15, 2021, ISSN: 2041-1723. DOI: 10.1038/s41467-021-27542-8. [Online]. Available: https://doi.org/10.1038/s41467-021-27542-8.

[5] P. Verschaffelt, T. Van Den Bossche, L. Martens, P. Dawyndt, and B. Mesuere, "Unipept Desktop: A Faster, More Powerful Metaproteomics Results Analysis Tool," *Journal of Proteome Research*, vol. 20, no. 4, pp. 2005–2009, Jan. 2021, ISSN: 1535-3907. DOI: 10.1021/acs.jproteome.0c00855. [Online]. Available: http://dx.doi.org/10.1021/acs.jproteome.0c00855.

[6] B. Mesuere, T. Willems, F. Van der Jeugt, B. Devreese, P. Vandamme, and P. Dawyndt, "Unipept web services for metaproteomics analysis," *Bioinformatics*, vol. 32, no. 11, pp. 1746–1748, Jan. 2016, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw039. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btw039.

[7] R. Gurdeep Singh, A. Tanca, A. Palomba, *et al.*, "Unipept 4.0: Functional Analysis of Metaproteome Data," *Journal of Proteome Research*, vol. 18, no. 2, pp. 606–615, Nov. 2018, ISSN: 1535-3907. DOI: 10.1021/acs.jproteome.8b00716. [Online]. Available: http://dx.doi.org/10.1021/acs.jproteome.8b00716.

[8] B. Mesuere, B. Devreese, G. Debyser, M. Aerts, P. Vandamme, and P. Dawyndt, "Unipept: Tryptic Peptide-Based Biodiversity Analysis of Metaproteome Samples," *Journal of Proteome Research*, vol. 11, no. 12, pp. 5773–5780, Nov. 2012, ISSN: 1535-3907. DOI: 10.1021/pr300576s. [Online]. Available: http://dx.doi.org/10.1021/pr300576s.

[9] B. Mesuere, F. Van der Jeugt, T. Willems, *et al.*, "High-throughput metaproteomics data analysis with Unipept: A tutorial," *Journal of Proteomics*, vol. 171, pp. 11–22, Jan. 2018, ISSN: 1874-3919. DOI: 10.1016/j.jprot.2017.05.022. [Online]. Available: http://dx.doi.org/10.1016/j.jprot.2017.05.022.

[10] B. Mesuere, G. Debyser, M. Aerts, B. Devreese, P. Vandamme, and P. Dawyndt, "The Unipept metaproteomics analysis pipeline," *PROTEOMICS*, vol. 15, no. 8, pp. 1437–1442, Feb. 2015, ISSN: 1615-9861. DOI: 10.1002/pmic.201400361. [Online]. Available: http://dx.doi.org/10.1002/pmic.201400361.

[11] P. Verschaffelt, P. Van Thienen, T. Van Den Bossche, *et al.*, "Unipept CLI 2.0: adding support for visualizations and functional annotations," *Bioinformatics*, vol. 36, no. 14, P. Luigi Martelli, Ed., pp. 4220–4221, Jun. 2020, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/

btaa553. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btaa553.

[12] P. Verschaffelt, A. Tanca, M. Abbondio, *et al.*, "Unipept Desktop 2.0: Construction of Targeted Reference Protein Databases for Metaproteogenomics Analyses," *Journal of Proteome Research*, vol. 22, no. 8, pp. 2620–2628, Jul. 2023, ISSN: 1535-3907. DOI: 10.1021/acs.jproteome.3c00091. [Online]. Available: http://dx.doi.org/10.1021/acs.jproteome.3c00091.

[13] The UniProt Consortium, "UniProt: the Universal Protein Knowledgebase in 2023," *Nucleic Acids Research*, vol. 51, no. D1, pp. D523–D531, Nov. 2022, ISSN: 0305-1048. DOI: 10.1093/nar/gkac1052. eprint: https://academic.oup.com/nar/article-pdf/51/D1/D523/48441158/gkac1052.pdf. [Online]. Available: https://doi.org/10.1093/nar/gkac1052.

[14] R. L. Mayer and K. Mechtler, "Immunopeptidomics in the Era of Single-Cell Proteomics," *Biology*, vol. 12, no. 12, 2023, ISSN: 2079-7737. DOI: 10.3390/biology12121514. [Online]. Available: https://www.mdpi.com/2079-7737/12/12/1514.

[15] A. Declercq, R. Bouwmeester, A. Hirschler, *et al.*, "MS²Rescore: Data-Driven Rescoring Dramatically Boosts Immunopeptide Identification Rates," *Molecular & Cellular Proteomics*, vol. 21, no. 8, p. 100 266, Aug. 2022, ISSN: 1535-9476. DOI: 10.1016/j.mcpro.2022.100266. [Online]. Available: http://dx.doi.org/10.1016/j.mcpro.2022.100266.

[16] A. Declercq, R. Bouwmeester, C. Chiva, *et al.*, "Updated MS²PIP web server supports cutting-edge proteomics applications," *Nucleic Acids Research*, vol. 51, no. W1, W338–W342, May 2023, ISSN: 1362-4962. DOI: 10.1093/nar/gkad335. [Online]. Available: http://dx.doi.org/10.1093/nar/gkad335.

[17] Y. Mori, *Libdivsufsort*, https://github.com/y-256/libdivsufsort.

[18] I. Grebnov, *Libsais*, https://github.com/IlyaGrebnov/libsais.

[19] *Unipept: Important statistics*, https://github.com/unipept/unipept/wiki/important-statistics.

[20] T. Van Den Bossche, B. J. Kunath, K. Schallert, *et al.*, "Critical assessment of MetaProteome investigation (CAMPI): A multi-laboratory comparison of established workflows," en, *Nat. Commun.*, vol. 12, no. 1, p. 7305, Dec. 2021.

[21] J. L. Krause, S. S. Schaepe, K. Fritz-Wallace, *et al.*, "Following the community development of SIHUMIx – a new intestinal in vitro model for bioreactor use," *Gut Microbes*, vol. 11, no. 4, pp. 1116–1129, 2020, PMID: 31918607. DOI: 10.1080/19490976.2019.1702431. eprint: https://doi.org/10.1080/19490976.2019.1702431. [Online]. Available: https://doi.org/10.1080/19490976.2019.1702431.

[22] *UniProt peptide search tool*, https://www.uniprot.org/peptide-search.

[23] C. Chen, Z. Li, H. Huang, B. E. Suzek, and C. H. Wu, "A fast Peptide Match service for UniProt Knowledgebase," *Bioinformatics*, vol. 29, no. 21, pp. 2808–2809, Aug. 2013, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btt484. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btt484.

[24] *Expasy ScanProsite tool*, https://prosite.expasy.org/scanprosite/.

[25] T. Holstein, F. Kistner, L. Martens, and T. Muth, "PepGM: a probabilistic graphical model for taxonomic inference of viral proteome samples with associated confidence scores," *Bioinformatics*, vol. 39, no. 5, btad289, May 2023, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btad289. eprint: https://academic.oup.com/bioinformatics/article-pdf/39/5/btad289/50311697/btad289.pdf. [Online]. Available: https://doi.org/10.1093/bioinformatics/btad289.

[26] P. Ferragina and G. Manzini, "Opportunistic data structures with applications," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*, Nov. 2000, pp. 390–398. DOI: 10.1109/SFCS.2000.892127.

[27] T. Gagie, G. Navarro, and N. Prezza, "Optimal-Time Text Indexing in BWT-runs Bounded Space," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, Jan. 2018, pp. 1459–1477. DOI: 10.1137/1.9781611975031.96. [Online]. Available: http://dx.doi.org/10.1137/1.9781611975031.96.

[28] N. Elhage, *Regular Expression Search with Suffix Arrays*, https://blog.nelhage.com/2015/02/regular-expression-search-with-suffix-arrays/, Feb. 2015.

[29] T. W. Lam, R. Li, A. Tam, S. Wong, E. Wu, and S. M. Yiu, "High Throughput Short Read Alignment via Bi-directional BWT," in *2009 IEEE International Conference on Bioinformatics and Biomedicine*, 2009, pp. 31–36. DOI: 10.1109/BIBM.2009.42.