RUHR-UNIVERSITÄT BOCHUM

Medical Faculty
Medical Proteome Center
Medical Bioinformatics / Functional Proteomics

Retrieving peptides by masses in protein-graphs to allow mutationally and variationally altered peptides to be exported into FASTA

Dominik Lux, Prof. Dr. Katrin Marcus, PD Dr. Martin Eisenacher, Dr. Julian Uszkoreit
{dominik.lux, katrin.marcus, martin.eisenacher, julian.uszkoreit}@rub.de

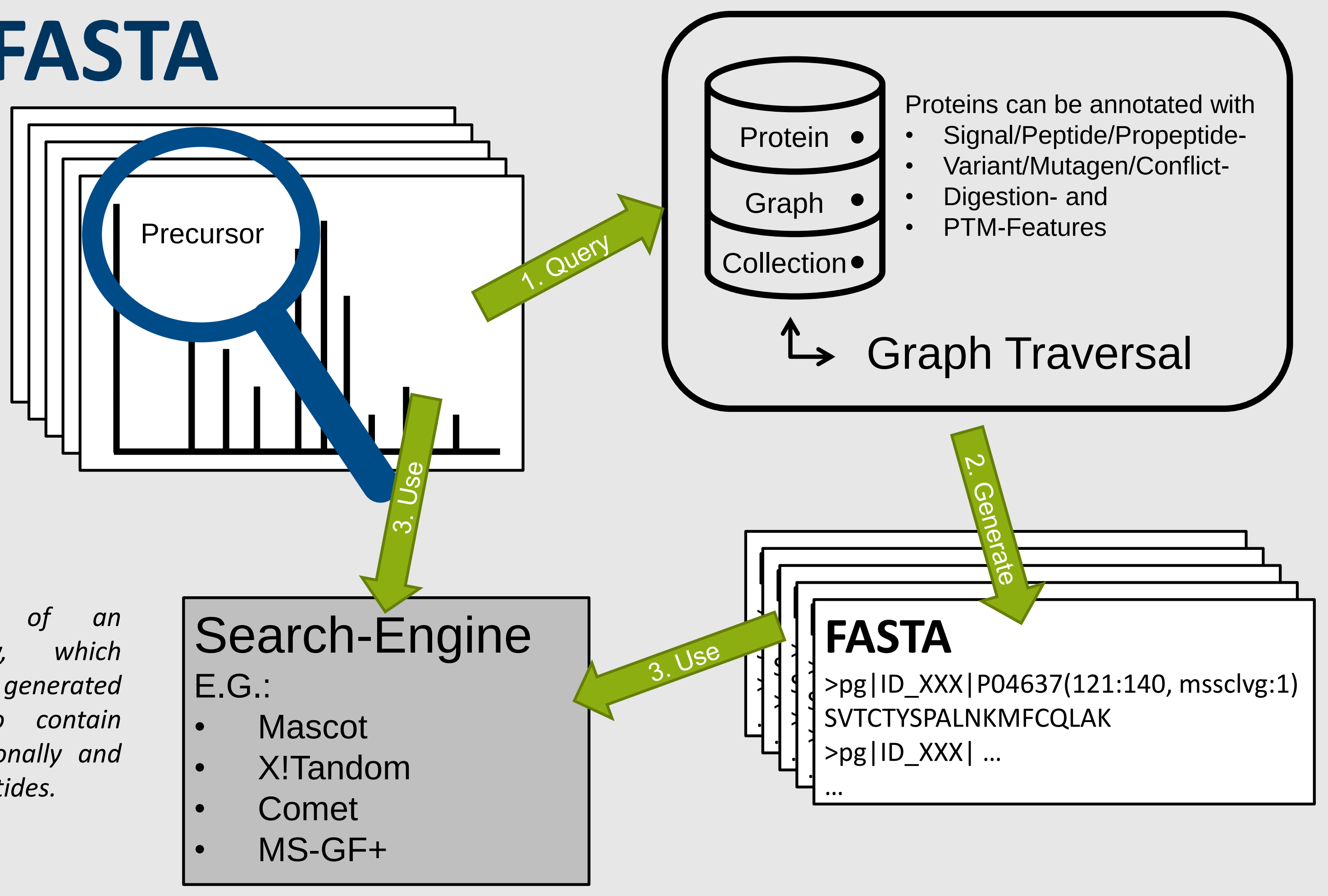

Figure 1: *Overview of an identification workflow, which utilizes protein-graphs to generated FASTA-files allowing to contain mutationally and variationally and other feature induced peptides.*

Abstract

Currently, spectra-identification relies on FASTA-files and their protein entries. By utilizing so called search engines, spectra can be annotated with peptide sequences for identification. However, FASTA-files mostly contain the canonical- sometimes isoform-sequences of proteins, which limit search engines to only identify these, failing to identify peptides containing variations and/or mutations. To enable search engines to identify such peptides, we use ProtGraph [1], a tool developed at our institute, to generate compact directed acyclic graphs from proteins, by utilizing SwissProt-EMBL entries from UniProtKB [2]. These protein-graphs encode the canonical (and isoform) sequences as well as feature information like mutational- or variational information and more. To utilize the encoded feature information for spectra-identification, an export from the graph into FASTA-format is mandatory. The huge number of peptides contained in such graphs can be challenging, creating large FASTA-files, so that a more sophisticated approach than a naïve export of all peptides is needed. An overview of the proposed approach is illustrated in **Figure 1**.

Methods

To accomplish an export into FASTA, we further extended ProtGraph to additionally include Post-Translational-Modification (PTMs). With this extension, protein-graphs encode theoretical peptide-masses accounting for PTMs. An example is illustrated in **Figure 2** and already contains 46 possible peptides (and 63 possible peptide masses) . More complex protein-graphs (like P53) contain too many peptides to be naïvely exported in a FASTA-format.

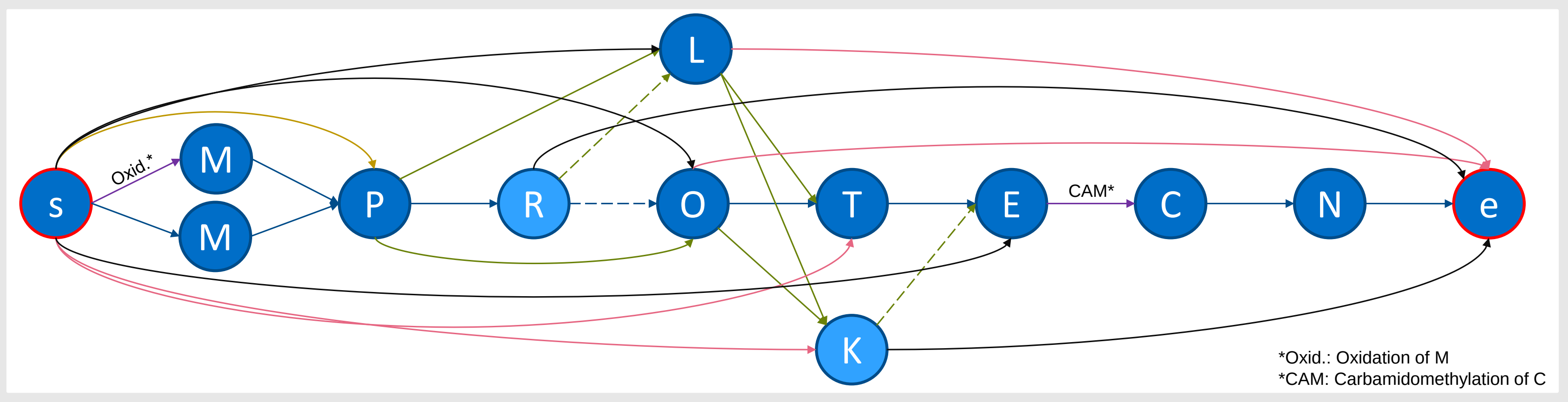

Figure 2: *Example protein-graph generated by ProtGraph. It contains various feature information, like 1 signal peptide (pink), 3 variants (green), 1 initiator methionine, which gets skipped and 2 PTMs (purple). This protein-graph is also digested (black, miscleavages are denoted by dotted lines) and holds 46 peptides.*

Node	1-Interval*		3-Interval*	
s	[113,1168]	[113,128]	[210,644]	[714,1168]
Moxid	[210,1021]	[210,494]	[618,618]	[714,1021]
M	[210,1021]	[210,494]	[618,618]	[714,1021]
P	[113,924]	[113,379]	[521,521]	[617,924]
R	[0,768]	[0,241]	[365,365]	[617,768]
O	[0,531]	[0,0]	[128,128]	[504,531]
L	[0,531]	[0,0]	[128,128]	[504,531]
T	[403,403]	[403,403]	-	-
K	[0,403]	[0,0]	[403,403]	-
E	[274,274]	[274, 274]	-	-
CCAM	[114,114]	[114,114]	-	-
N	[0,0]	[0,0]	-	-
e	[0,0]	[0,0]	-	-

**Decimals are removed for illustration purpose*

Table 1: *This table annotates the PDB-Entries with k=1 and k=3 for the protein-graph in* ***Fig. 2****, where k denotes the number of intervals a node can hold. It can be observed that with k=1 the lowest and the highest boundaries of an interval are also present with k=3. This is also true for any other arbitrary k. The higher k is set, the earlier it can be decided whether a path with a specific mass is achievable.*

Fortunately, not all peptides from a protein-graph must be exported. We can limit the number of exported peptides, fitting to the precursor-mass of a MS2-spectrum, which reduces the number of exported peptides from protein-graphs massively. So called pattern-database-entries [3] (PDBs) are utilized to annotate the monoisotopic mass of amino acids and to provide information of achievable masses from a specific node. **Table 1** illustrates PDBs with k=1 and k=3. E. g.: The PDBs on node s can be used directly to query if a specific mass in Dalton is achievable in this protein-graph.

Since protein-graphs generated by ProtGraph are directed and acyclic, a specific topological order was chosen, which is used for traversing a protein-graph isoform by isoform. Moreover, another exporter was added to ProtGraph, calculating the PDBs using a dynamic programming approach in conjunction with the topological order. The protein-graphs are then saved in a compressed-sparse-row representation in a binary file. The actual graph-traversal to return only peptides fitting to the precursor and to ultimately generate a FASTA-file has been implemented multiple times in C++, utilizing the PDBs to enhance the graph-traversal and to provide a comparison between methods.

Materials

To evaluate the performance, the whole UniProt database (21.02.2022), containing ~220M proteins was exported via ProtGraph (0.3.0) using only the canonical sequence and no feature information. k=5 was selected, as well as carbamidomethylation of C as a fixed modification and oxidation of M as a variable modification was set. This generated a 1.4 TB large binary file. The well-known tumour-suppressor P53 (P04637) was exported separately, utilizing all feature information ProtGraph can parse to assess the performance on complex protein-graphs. In this case, the same k as well as modifications were used. P53 was specifically chosen, since the number of peptides contained within is dominating the search space, if no variable modifications are used (results not shown).

The performance was tested on a server with 2 AMD EPYC 7452 CPUs (total 128 Threads) and 2 TB RAM. The source code was compiled via gcc 11.2.0 and the graph-traversal runtime on protein-graphs was measured programmatically in source code.

F953N-19
FoRUM

PRODI

References:
[1] https://github.com/mpc-bioinformatics/ProtGraph
[2] "UniProt: the universal protein knowledgebase in 2021." Nucleic acids research 49, no. D1 (2021): D480-D489.
[3] Schmidt, Tim, Lukas Kuhn, Bob Price, Johan De Kleer, and Rong Zhou. "A depth-first approach to target-value search." In Symposium on Combinatorial Search (SOCS-09). 2009.
[4] Gael Guennebaud and Benoît Jacob and others. "Eigen v3" http://eigen.tuxfamily.org 2010.

Results

We first evaluated the traversal performance on the whole UniProt dataset. The C++-implementation first loads the database into RAM (1.48TB) and then executes the graph-traversal with the specific method per protein-graph. **Figure 3** illustrates the traversal time per protein for a specific query mass. For ~50-75% of proteins the graph-traversal took 10 microseconds or less except for the implementation utilizing whole numbers to encode amino acid masses (int64). It can be observed that for all implementations the longest running protein took around 10 milliseconds to traverse. Generally a trend can be seen: The larger the queried mass gets, the longer a query will take to finish.

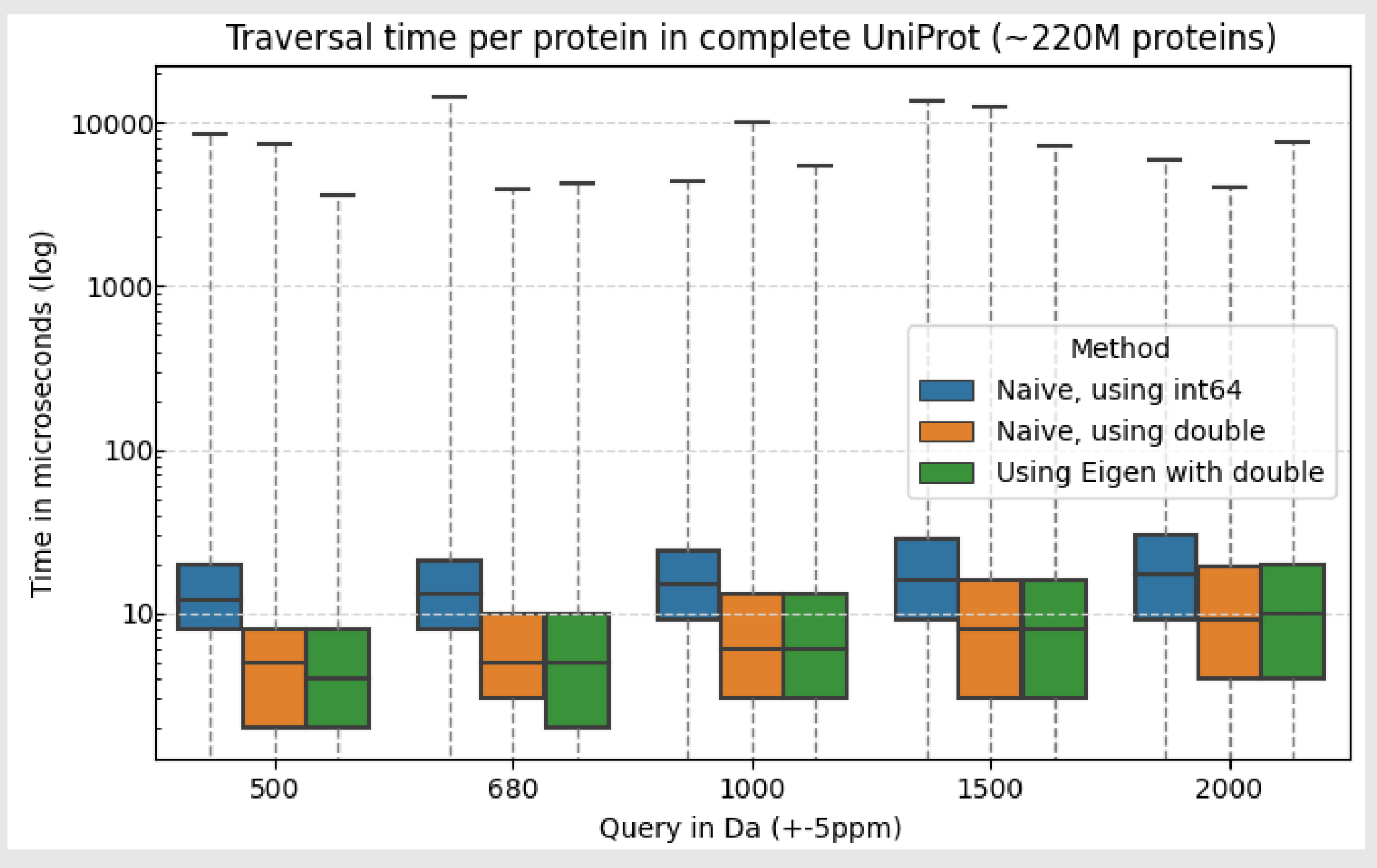

Figure 3: *Boxplots of the runtime per protein on the whole UniProt dataset. The whiskers also account for maximum and minimum runtime of a protein-graph. The UniProt dataset contains up to 1.171*10^270 possible masses for peptides. However, most of the masses can be queried efficiently, due to the simple structure of the protein-graphs.*

Figure 4 represents the sum of each point in a boxplot from **Figure 3** as a bar. This visualizes the total runtime the graph-traversal would need for a single thread. Running a mass query in parallel with 128 Threads, a result is expected to be returned under 1000 seconds.

In **Figure 5** multiple queries have been executed on the protein-graph of P53 and the runtime was measured. It can be observed, how each method behaves on a complex graph. We further tested it by limiting the number of variants a peptide can contain. Only one method was plotted, but others returned similar runtimes.

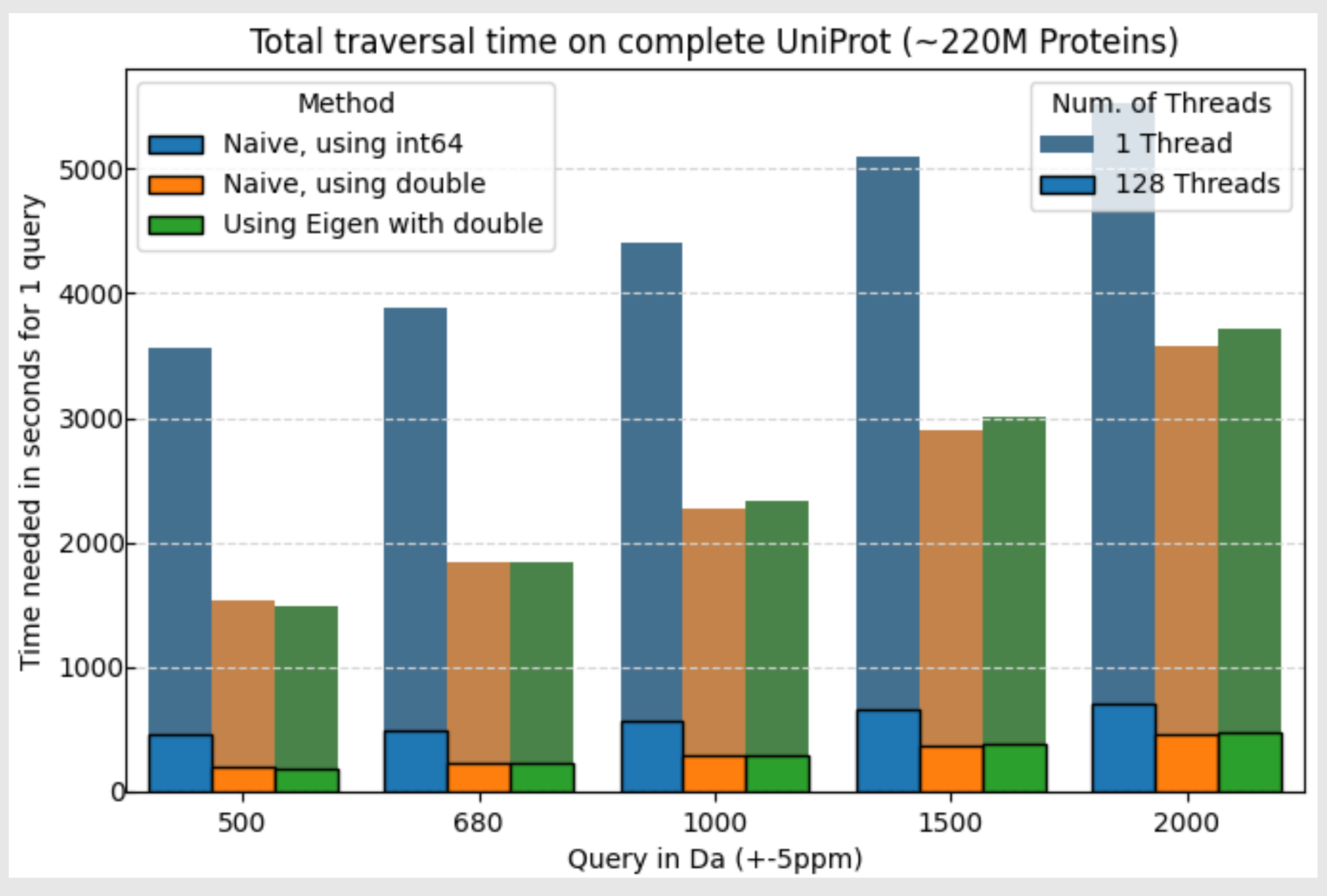

Figure 4: *Summed runtime for the whole UniProt dataset of each graph-traversal. Each traversal of a protein-graph can be executed in a thread and therefore can be executed highly parallel. This plot shows the estimated runtime for 1 query using 1 Thread and 128 Threads.*

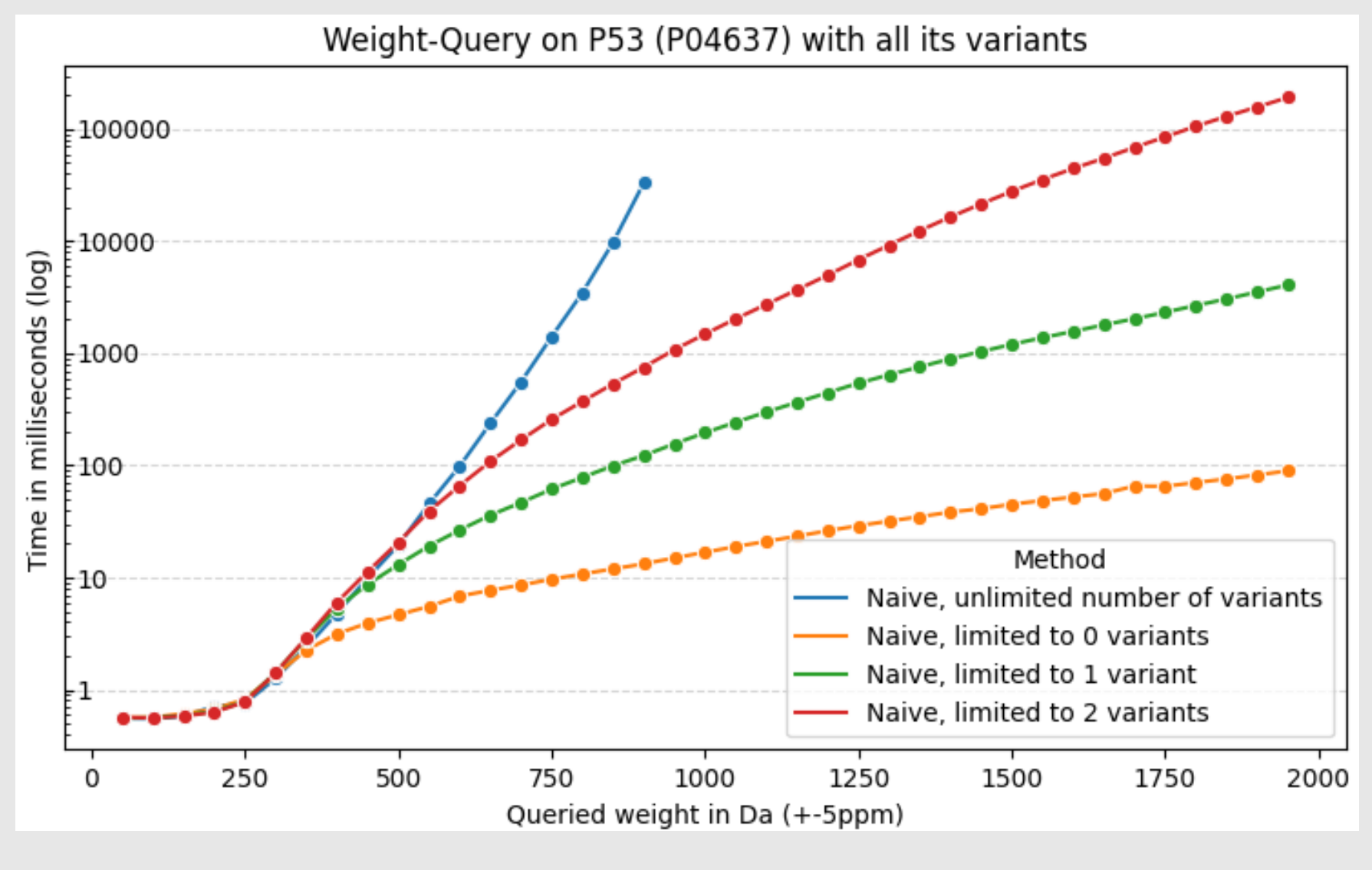

Figure 5: *Runtime of the graph-traversal on a protein-graph from P53. For each 50 Da the runtime was captured. This protein-graph encodes 9.244*10^227 possible masses for peptides. The naïve method utilizing whole numbers (int64) is displayed. Similar runtimes can be observed on other methods and are omitted.*

Discussion

We show that a fast graph-traversal and therefore a FASTA-export is possible for non-complex graphs. The graph-traversal can be distributed across multiple threads, hence is highly scalable. A quick FASTA-export is possible if enough computing power is provided for the whole UniProt dataset. However, it needs to be noted that for 1 mass-query on this dataset "infinite"-many miscleavages, "infinite"-long peptides as well as "infinite"-many variable modifications are considered. The fast retrieval is impressive (**Figure 3 and 4**), since the whole UniProt dataset contains up to 1.171*10^270 possible masses for peptides. Interestingly, utilizing double-precision instead of whole numbers as amino acid masses is almost twice as fast on the whole UniProt dataset. This could be due to some compiler optimization or the CPU architecture itself. However, the performance improvement may not be as drastic and could vary if applied on other CPU-models.

The query-time on P53 (**Figure 5**) shows that the graph-traversal on complex graphs needs to be further improved and that only limiting the number of variants is not sufficient. The poor runtime is due to the protein itself. Having over 1000 variants annotated, it generates a much more dense graph then usual. In general only a few proteins are annotated that well while other proteins with variational and mutational information, which are not as well researched as P53, can be queried quickly with the provided implementation (not shown).

As next steps, we will further explore other implementations and libraries (like the Eigen-library [4]) to increase the performance. It could be interesting to look into openBLAS or a GPU implementation for the graph-traversal. We also want to further investigate the complexity of complex protein-graphs (like P53) and are exploring approaches to reduce the amount of containing peptides in a reasonable way.