FACULTY OF SCIENCES

BUILDING THE UNIPEPT ECOSYSTEM

EMPOWERING HIGH-THROUGHPUT METAPROTEOMICS DATA ANALYSIS FOR CHARACTERIZING COMPLEX MICROBIAL COMMUNITIES

Pieter Verschaffelt

Supervisors:
Prof. Dr. Peter Dawyndt
Prof. Dr. Lennart Martens
Prof. Dr. Ir. Bart Mesuere

Dissertation submitted in fulfillment of the requirements for the degree of Doctor of Computer Science

Academic year: 2022 - 2023

GHENT UNIVERSITY

Permission for use of content

The author gives permission to make this dissertation available for consultation and to copy parts of this dissertation for personal use.

In the case of any other use, the limitations of the copyright have to be respected, in particular with regard to the obligation to state explicitly the source when quoting results from this dissertation.

Pieter Verschaffelt
March 2023

Acknowledgements

This is where I will thank a lot of people!

Summary

Proteins are ubiquitous. They are an essential part of all life on earth and they are responsible for a whole range of vital functions. Enzymes, for example, take care of converting nutrients into energy and other building blocks required for a properly functioning body. Or hormones, another class of proteins, take care of the communication between different parts and organs in our body and regulate essential processes. Not only humans, but all living organisms on our planet depend on proteins for their survival and efficient protein research is thus of significant importance.

Proteins are so-called macromolecules (i.e. very big molecules), constructed from smaller amino acids. These amino acids are floating around in our body and are connected in a specific sequence in order to form proteins with a specific function and structure. The exact sequence of these amino acids is what determines the function of the proteins and is recorded as part of our DNA. A small change to an organism's DNA could lead to some proteins not being able to be produced anymore, or altering their functionality. Since the first discovery of proteins in nature, a lot of research around these big molecules has been conducted and allows researchers to compile a big protein database where they keep track of which protein is associated with which organism and what its (known) function is.

With the help of a mass spectrometer, a very expensive and complex measuring device, and a series of advanced computer analyses, it is possible to derive which protein sequences are present in a specific environment (e.g. blood, stool, soil, etc.). Subsequently, the protein sequences can then be looked up in the protein reference database and a summary of all identified organisms in the ecosystem under study (as well as the functions that they are responsible for) can be compiled.

In order to make this process as easy as possible, we started the development of Unipept at Ghent University. Unipept is a software application that processes a series of protein fragments (i.e. peptides) and looks up each of these fragments in the protein reference database in order to construct a taxonomic and functional profile of an ecosystem under study. The taxonomic profile of a sample, together with a collection of interactive data visualizations, will help researchers to find out **who** is present in an ecosystem. In order to increase insight into **what** these organisms are doing at a specific point in time, researchers can take a look at the functional profile

of a sample (as generated by Unipept). This reasoning clearly explains the advantage of studying proteins instead of looking at the DNA found in a sample. Instead of only figuring out **who** is present, we can also find out **what** these organisms are doing.

Over the last 10 years, a lot of technical advances have been made in the field of mass spectrometers. This allows protein researchers to more easily analyse large samples in one go and generate a larger amount of data that needs to be processed by Unipept subsequently. Because Unipept was initially developed as a web application and thus always relies on a web browser (such as Google Chrome), it becomes harder and harder to keep up with the advances in mass spectrometry technology and the huge amount of data that comes with it. In Chapter 2 you can read how I tackled this problem by developing the novel Unipept Desktop application. The first version of the Unipept Desktop application allows to process samples that are up to 10 times larger than before (containing up to half a million peptides) and to compare analysis results with each other. Unipept Desktop v1.0 also allows, for the first time, to organize samples into projects and studies (allowing researchers to link similar experiments).

In Chapter 3 I go one step further and present to you version 2.0 of the Unipept Desktop application. This version is the first one to provide support for the analysis of **proteogenomics** samples. Proteogenomics is a novel research discipline that is currently emerging and consists of combining the information from a genetic experiment (derived from DNA analyses) with the information that is extracted from a protein sample. Proteins that are produced by distinct organisms can still be very similar, making it impossible for Unipept to distinguish between them. Instead of returning a list of all organisms that are detected, Unipept will typically return a list of organism groups that potentially *can* be present in the ecosystem under study. Sometimes, this information is sufficient for researchers to continue with their experiments, but often it occurs that the functional profile of the ecosystem is not informative enough.

In the study of proteogenomics, researchers are first going to perform a genetic experiment in order to find out which organisms are present in an ecosystem and in order to decrease the size of the protein search space. Only the proteins of those organisms that were identified in the first step will be considered during the subsequent protein matching step which can then potentially increase the resolution of the taxonomic and functional profile as generated by Unipept.

A lot of different functional classification systems for proteins exist and Unipept provides extensive support for three of them. The Enzyme Commission numbers (EC-numbers), Gene Ontology terms (GO-terms) and InterPro entries all have their own advantages and disadvantages and typically each focus on a distinct class of proteins.

In Chapter 4, I explain how we have expanded the **Unipept API and CLI** with support for functional annotations. The API (Application Programming Interface) is a collection of resources that are provided by Unipept and that can be used by third-party applications to integrate some of Unipept's analysis features. Unipept's CLI (Command Line Interface) is a separate software application that does not provide a graphical user interface, but that can easily be plugged into existing analysis pipelines and allows processing larger samples.

One of Unipept's major strengths is the collection of interactive data visualizations that it provides which each increase the insight of reseachers into the taxonomic and functional composition of a sample. Each of these visualizations have been developed in-house and are not only suitable for visualizing protein analysis results, but can be applied to a wide range of different data sets. In Chapter 5 I talk briefly about the development of these visualizations and their public availability as a software library.

Unipept will typically report a whole collection of GO-terms for each protein sample, which opens up the possibility for researchers to compare these samples with each other in order to find out how **similar** they are. In Chapter 6 I explain how I, together with a group of researchers from all over Europe, developed a similarity metric for GO-terms. This metric produces a value between 0 and 1 every time it is given two sets of GO-terms. The closer this values is to 1, the higher the similarity between these two sets of GO-terms.

Next to this listing of improvements that I have made to Unipept over the years, I briefly talk about a number of different projects that I was deeply involved with during my PhD in Chapter 7. Finally, a discussion about what the future might hold for Unipept, can be found in Chapter 8.

Samenvatting

Eiwitten zijn alomtegenwoordig. Ze vormen een essentieel onderdeel voor alle leven op aarde en staan in voor een heleboel levensbelangrijke functies. Zo zorgen enzymen er bijvoorbeeld voor dat ons lichaam op een efficiënte manier voedingsstoffen kan omzetten in energie en andere bouwstenen. Of zorgen onze hormonen ervoor dat verschillende cellen en organen met elkaar kunnen communiceren en reguleren ze essentiële levensprocessen. Niet alleen mensen, maar alle levende organismen op onze planeet steunen op de goede werking van eiwitten voor hun voortbestaan en dus kan efficiënt onderzoek naar deze grote moleculen van enorm belang zijn.

Een eiwit is een zogenaamde macromolecule die opgebouwd wordt uit aminozuren. Deze aminozuren zweven rond in ons lichaam en worden in een strikte volgorde aan elkaar gekoppeld om een eiwit met een bepaalde functie en structuur op te bouwen. De exacte volgorde van deze aminozuren bepaalt wat de functie van het eiwit wordt en ligt vast in ons DNA. Een kleine wijziging aan het DNA van een organisme kan er dus voor zorgen dat bepaalde eiwitten niet langer geproduceerd gaan worden, of dat ze zich anders gaan gedragen. Sinds de eerste ontdekking van eiwitten in de natuur, is er reeds een heleboel onderzoek naar deze grote moleculen uitgevoerd en hebben onderzoekers een grote databank opgesteld waarin ze voor elk gekend eiwit bijhouden bij welk organisme het hoort en wat zijn functie is (indien gekend).

Dankzij de massaspectrometer, een erg complex en duur toestel, en een reeks van geavanceerde analyses op een computer, is het mogelijk om te achterhalen welke eiwitsequenties er voorkomen in een bepaalde omgeving (zoals het bloed, de stoelgang, de grond rondom een bepaalde plant, etc.) wat ons daarna ook weer in staat stelt om deze eiwitsequenties op te zoeken in de bestaande eiwitdatabank. Door elk van de geïdentificeerde eiwitsequenties op te zoeken in deze databank, kunnen we een rapport opstellen met de organismen die voorkomen in de onderzochte omgeving en welke functies ze daar mogelijk aan het uitvoeren zijn.

Om dit hele proces zo eenvoudig mogelijk te maken werd Unipept ontwikkeld. Unipept is een softwarepakket dat een reeks van geïdentificeerde eiwitfragmenten zal opzoeken en proberen koppelen aan de eiwitten uit de grote eiwitdatabank en met behulp van deze informatie een taxonomisch en functioneel profiel zal opstellen voor het ecosysteem dat door onderzoekers

momenteel verkend wordt. Samen met een heleboel interactieve datavisualisaties zal het taxonomisch profiel van een staal de onderzoekers een duidelijk beeld geven van **wie** (dus welke organismen) er in een ecosysteem actief zijn. Om daarnaast ook een dieper inzicht te verkrijgen in **wat** deze organismen op een bepaald moment aan het doen zijn, zullen onderzoekers verder kijken naar het functioneel profiel, zoals dat door Unipept wordt gegenereerd. Deze redenering maakt ook onmiddellijk duidelijk waar de kracht in het onderzoeken van eiwitten nu in schuilt. In plaats van enkel onderzoek te voeren naar *wie* in een ecosysteem aanwezig is (zoals dat typisch gebeurt bij een genetisch experiment), wordt het opeens ook mogelijk om na te gaan *wat* er op een bepaald moment in de tijd gebeurt.

Technologie staat nooit stil en de laatste 10 jaar werden er steeds krachtigere massaspectrometers ontwikkeld. Een gevolg hiervan is dat het voor eiwitonderzoekers eenvoudiger wordt om grotere stalen in 1 keer te gaan verwerken en is het aantal eiwitfragmentjes die door Unipept geanalyseerd moeten worden constant aan het toenemen. Omdat Unipept initieel ontwikkeld werd als webapplicatie en dus steeds door een webbrowser (zoals Google Chrome) uitgevoerd moet worden, werd het steeds moeilijker om te kunnen bijbenen met de huidige verbeteringen in massaspectrometer-technologie en de enorme hoeveelheid aan data die daarbij komt kijken. In hoofdstuk 2 kan u lezen hoe ik dit probleem heb aangepakt door de **Unipept Desktop** applicatie te ontwikkelen. De eerste versie van Unipept Desktop biedt de mogelijkheid om stalen te verwerken die tot 10 keer groter zijn dan wat vroeger mogelijk was met Unipept en maakt het mogelijk om de resultaten van verschillende stalen met elkaar te gaan vergelijken. Daarnaast kan u hier ook lezen dat het vanaf nu mogelijk is om stalen op een overzichtelijke manier te ordenen en gelijkaardige experimenten met elkaar te koppelen.

In hoofdstuk 3 gaan we nog een stapje verder en stel ik versie 2.0 van de Unipept Desktop-applicatie voor. Deze versie biedt als eerste ondersteuning voor het analyseren van **proteogenomics** stalen. Proteogenomics is een nieuwe onderzoeksdiscipline die momenteel sterk aan het opkomen is, en bestaat erin van genetische informatie (afkomstig uit DNA-experimenten) te combineren met de informatie die we vanuit een eiwitstaal kunnen verkrijgen. Eiwitten die door verschillende organismen geproduceerd worden, kunnen vaak toch sterk op elkaar lijken waardoor het voor Unipept in dit geval onmogelijk is om te weten welke organismen er exact aanwezig zijn in een ecosysteem. In plaats van een lijst met gedetailleerde taxonomische informatie te rapporteren, zal Unipept typisch een klasse van organismen die *mogelijk* aanwezig zijn in het ecosysteem genereren. Soms is deze in-

formatie voldoende voor onderzoekers om verder te kunnen gaan met hun experiment, maar vaak komt het ook voor dat als gevolg hiervan het functionele profiel voor het ecosysteem dat men onderzoekt niet informatief genoeg is.

Binnen het proteogenomics onderzoeksgebied gaan onderzoekers eerst een genetisch experiment uitvoeren om uit te zoeken welke organismen potentieel aanwezig kunnen zijn binnen een ecosysteem en gaat men op basis daarvan de groep met potentiële eiwitmatches verkleinen. Enkel de eiwitten van die organismen die volgens het eerste experiment aanwezig kunnen zijn, worden ter beschouwing genomen en kunnen de resolutie van het taxonomisch en functioneel profiel van een eiwitstaal dat door Unipept gegenereerd werd verhogen.

Er bestaan een heleboel verschillende classificaties waarmee de functies die door een organisme uitgevoerd worden, aangeduid kunnen worden. Unipept biedt ondersteuning voor Enzyme Commission numbers (i.e. EC-numbers), Gene Ontology terms (i.e. GO-terms) en InterPro entries. Elk van deze classificaties heeft zijn eigen voor- en nadelen en richt zich typisch op een andere klasse van eiwitten.

In hoofdstuk 4 kan u lezen hoe we de **Unipept API en CLI** uitgebreid hebben met ondersteuning voor het rapporteren van dergelijke functionele annotaties. De API (Application Programming Interface) is een verzameling van resources die door Unipept worden aangeboden en die het mogelijk maken om de analyses van Unipept op te nemen in externe softwarepakketten. Unipepts CLI (Command Line Interface) is een apart stukje software dat niet over een grafische gebruikersinterface beschikt, maar dat net om die reden eenvoudig ingeplugd kan worden in bestaande analysepipelines en het mogelijk maakt om grotere stalen te verwerken.

Een van de sterktes van Unipept is de collectie aan visualisaties die we aanbieden en het inzicht van een gebruiker in de taxonomische en functionele samenstelling van een staal sterk kunnen verbeteren. Elk van deze visualisaties werden ontwikkeld binnen ons eigen team en zijn niet enkel bruikbaar voor het visualiseren van de resultaten uit een eiwitanalyse, maar kunnen gebruikt worden om een veel ruimere klasse aan data visueel voor te stellen. In hoofdstuk 5 beschrijf ik kort hoe we deze visualisaties hebben ontwikkeld en dat deze beschikbaar zijn voor externe gebruikers door middel van een publieke softwarebibliotheek.

Voor elk geanalyseerd eiwitstaal zal Unipept typisch een hele reeks aan GO-

termen kunnen rapporteren, en wordt het voor onderzoekers interessant om te kunnen bepalen hoe sterk twee stalen op elkaar **gelijken**. In hoofdstuk 6 kan u lezen hoe ik, samen met een groep wetenschappers uit heel Europa, een metriek heb ontwikkeld die aangeeft hoe sterk twee verzamelingen van Gene Ontology termen overeenkomen. De metriek zal voor twee verzamelingen van GO-termen een getal tussen 0 en 1 produceren waarbij een groter getal (dus dichter bij 1) aangeeft dat de twee verzamelingen sterker op elkaar gelijken.

Naast alle toevoegingen waar ik aan heb gewerkt die rechtstreeks in Unipept geïntegreerd werden, vindt u in hoofdstuk 7 een overzicht van een aantal projecten waar ik ook een grote rol in heb gespeeld en kan u in hoofdstuk 8 lezen wat de toekomst voor Unipept mogelijk inhoudt.

List of publications

Unipept 4.0: functional analysis of metaproteome data

Journal of Proteome Research, 18 (2): 606-615 **2018**

Robbert Gurdeep Singh, Alessandro Tanca, Antonio Palomba, Felix Van der Jeugt, **Pieter Verschaffelt**, Sergio Uzzau, Lennart Martens, Peter Dawyndt, and Bart Mesuere

Connecting MetaProteomeAnalyzer and PeptideShaker to Unipept for seamless end-to-end metaproteomics data analysis

Journal of Proteome Research, 19 (8): 3562-3566 **2020**

Tim Van Den Bossche, **Pieter Verschaffelt**, Kay Schallert, Harald Barsnes, Peter Dawyndt, Dirk Benndorf, Bernhard Y Renard, Bart Mesuere, Lennart Martens, and Thilo Muth

Unipept CLI 2.0: adding support for visualizations and functional annotations

Bioinformatics, 36 (14): 4220-4221 **2020**

Pieter Verschaffelt, Philippe Van Thienen, Tim Van Den Bossche, Felix Van der Jeugt, Caroline De Tender, Lennart Martens, Peter Dawyndt, and Bart Mesuere

Proceedings of the EuBIC-MS 2020 Developers' Meeting

EuPA Open Proteomics, 24: 1-6 **2020**

Christopher Ashwood, Wout Bittremieux, Eric W Deutsch, Nadezhda T Doncheva, Viktoria Dorfer, Ralf Gabriels, Vladimir Gorshkov, Surya Gupta, Andrew R Jones, Lukas Käll, Dominik Kopczynski, Lydie Lane, Ludwig Lautenbacher, Marc Legeay, Marie Locard-Paulet, Bart Mesuere, Yasset Perez-Riverol, Eugen Netz, Julianus Pfeuffer, Timo Sachsenberg, Renee Salz, Patroklos Samaras, Henning Schiebenhoefer, Tobias Schmidt, Veit Schwämmle, Alessio Soggiu, Julian Uszkoreit, Tim Van Den Bossche, Bart Van Puyvelde, Joeri Van Strien, **Pieter Verschaffelt**, Henry Webel, and Sander Willems

Unipept Desktop: a faster, more powerful metaproteomics results analysis tool

Journal of Proteome Research, 20 (4): 2005-2009 **2021**

Pieter Verschaffelt, Tim Van Den Bossche, Lennart Martens, Peter Dawyndt, and Bart Mesuere

MegaGO: A fast yet powerful approach to assess functional gene ontology similarity across meta-omics data sets

Journal of Proteome Research, 20 (4): 2083-2088 **2021**

Pieter Verschaffelt, Tim Van Den Bossche, Wassim Gabriel, Michał Burdukiewicz, Alessio Soggiu, Lennart Martens, Bernhard Y Renard, Henning Schiebenhoefer, and Bart Mesuere

Critical Assessment of MetaProteome Investigation (CAMPI): a multi-laboratory comparison of established workflows

Nature Communications, 12 (1) **2021**

Robert-Bob Hettich, Richard J Giannone, Paul Abraham, Tim Van Den Bossche, Benoit Kunath, Kay Schallert, Stephanie S Schäpe, Jean Armengaud, Magnus Arntzen, Ariane Bassignani, Dirk Benndorf, Stephan Fuchs, Timothy Griffin, Live Hagen, Rashi Halder, Celine Henry, Robert Heyer, Pratik Jagtap, Nico Jehmlich, Marlene Jensen, Catherine Juste, Manuel Kleiner, Olivier Langella, Pedro Queiros, Udo Reichl, Theresa Lehmann, Emma Leith, Patrick May, Bart Mesuere, Guylaine Miotello, Bernhard Renard, Henning Schiebenhoefer, Alexander Sczyrba, Alessandro Tanca, Kathrin Trappe, Jean-Pierre Trezzi, Sergio Uzzau, **Pieter Verschaffelt**, Martin Von Bergen, Paul Wilmes, Maximilian Wolf, Lennart Martens, and Thilo Muth

Unipept Visualizations: an interactive visualization library for biological data

Bioinformatics, 38 (2): 562-563 **2021**

Pieter Verschaffelt, James Collier, Alexander Botzki, Lennart Martens, Peter Dawyndt, and Bart Mesuere

Pout2Prot: an efficient tool to create protein (sub)groups from percolator output files

Journal of Proteome Research, 21 (4): 1175-1180 **2021**

Kay Schallert, **Pieter Verschaffelt**, Bart Mesuere, Dirk Benndorf, Lennart Martens, and Tim Van Den Bossche

UMGAP: the Unipept MetaGenomics Analysis Pipeline

BMC genomics, 21 (4): 23 (1): 1-16 **2022**

Felix Van der Jeugt, Rien Maertens, Aranka Steyaert, **Pieter Verschaffelt**, Caroline De Tender, Peter Dawyndt, and Bart Mesuere

Unipept Desktop 2.0: construction of targeted reference protein databases for proteogenomics analyses

Journal of Proteome Research, submitted (under review)

Pieter Verschaffelt, Alessandro Tanca, Marcello Abbondio, Tim Van Den Bossche, Tibo Vande Moortele, Peter Dawyndt, Lennart Martens, and Bart Mesuere

List of conferences and research stays

EuBIC Developer's Meeting 2020

January, 2020 **Nyborg, Denmark**

Together with Henning Schiebenhoefer, Tim Van Den Bossche and Bart Mesuere, I hosted a hackathon called "Mapping proteins to functions: method and benchmark development". During this hackathon, we developed a tool called "MegaGO" which has also been published in the Journal of Proteome Research. The result of this hackathon was a nice software package consisting of a web application, a command line tool and an application programming interface that is publicly accessible.

VIB: Research Software Developers Day

December, 2020 **Online**

The Research Software Developers Day, organised by VIB, was an online conference that consisted of software developers presenting their workflow and ideas about the development of research software. I was one of the presenters on this day and performed a 20-minute talk about the development of the Unipept Desktop application. This talk was followed by 10 minutes of questions and discussion afterwards. A recording of the talk is available on YouTube: https://youtu.be/2ftKfJkcJeY.

Online Metaproteomics Symposium

June, 2021 **Online**

This small online symposium mainly consisted of a quick overview and update of recent developments in the field of metaproteomics and an introduction to the recently started Metaproteomics Iniative. I was only a spectator at this online conference and did only passively participate in the activities.

Fourth Metaproteomics Symposium

September, 2021 **Luxembourg, Luxembourg**

The aim of this symposium was to provide a platform for the participants to share their latest results in their respective fields using metaproteomic methods as well as discussing recent technologic innovations and presenting newly developed bioinformatic tools. I presented the Unipept Desktop tool at this conference during a 20-minute presentation (with a 10 minute questions session afterwards).

EuBIC Winter School 2022

March, 2022 **Lisbon, Portugal**

The EuBIC-MS Winter School on computational mass spectrometry (MS) takes place every two years. Its aim is to bring together the users and developers of computational mass spectrometry tools, as well as academia and industry. The winter school started with an educational day dedicated to workshops and trainings in established computation MS tools and workflows. The following days, internationally renowned invited speakers gave lectures and practical workshops covering the aspects of identification, quantification, result interpretation, and integration of MS data. I presented a poster during this conference and participated in the workshops and talks given during this week.

Research stay at the Bundesanstalt für Materialforschung und -prüfung (BAM)

September - October, 2022 **Berlin, Germany**

Together with Tanja Holstein, I worked on a project to integrate PepGM with Unipept. PepGM is a probabilistic graphical model for taxonomic inference of proteome samples with associated confidence scores. It started out as a tool for the analysis of viral proteome samples and by better integrating Unipept with PepGM, we were able to expand the tool with support for proper metaproteomics datasets.

HUPO 2022 World Congress

December, 2022 **Cancun, Mexico**

HUPO is the Human Proteome Organization which organizes a world congress somewhere around the world each year. During these conferences, lectures by world renowned speakers and exciting networking opportunities are offered. I presented a poster about proteogenomics analysis using the Unipept Desktop application at this conference and participated in a discussion at the metaproteomics-specific session.

EuBIC Developers' Meeting 2023

January, 2023 **Ascona, Switzerland**

The EuBIC-MS Developers Meeting is organized every other year. This meeting is aimed at bringing together computer scientists and developers in the field of mass spectrometry-related bioinformatics to discuss and work together in an open and constructive spirit. The program is split between keynote lectures and multiple hackathon sessions where the participants develop bioinformatics tools and resources addressing outstanding needs in the mass spectrometry-related bioinformatics and user community.

Together with Tibo Vande Moortele and Tim Van Den Bossche, I hosted a hackathon session at this conference entitled "Exploring and solving functional analysis gaps in metaproteomics". During this week, we've had a lively discussion with the members of our hackathon team to discuss these functional analysis gaps in metaproteomics and settled on improving support for highlighting taxonomic diversity of metaproteomic samples in metabolic pathways. We started the development of a web application that allows users to upload a list of peptides, which will be taxonomically analysed. All identified taxa will be highlighted on a metabolic pathway by using the KEGG database.

List of repositories

Below, you can find a list of all repositories and code packages that have (in part) been developed or maintained by me over the course of my PhD. These repositories are all publicly accessible on GitHub. Some of the repositories that I worked on are no longer maintained or have been migrated to a different location. Only actively used and maintained repositories are listed here. Repositories set up for temporary experiments or deprecated projects are omitted for brevity.

unipept-web

/unipept/unipept-web **5.0.4**

</> TypeScript, Vue, Vuetify

The Unipept Web application is developed using TypeScript, in combination with the Vue and Vuetify frameworks. Because we are using the Vue framework, the application is structured as a collection of web components that can be reused between different pages and parts of the web app. By browsing to https://unipept.ugent.be, researchers have direct access to a browser-based application for the analysis of metaproteomics datasets.

unipept-desktop

/unipept/unipept-desktop **2.0.0**

</> TypeScript, Vue, Vuetify, Electron

All code related to the Unipept Desktop application can be found in this repository. The desktop application, in contrast to the Unipept Web application, focuses on the analysis of very large metaproteomics samples and a better (hierarchical) organization of samples that somehow belong together. Because we wanted to maximize the amount of code that can be shared between the Unipept Web and Desktop application, we chose to use the Electron framework for primary development of our most recent tool. Electron[1] is a software framework that allows desktop applications to be developed using web-based technologies (such as TypeScript, HTML, etc.).

[1] See https://www.electronjs.org/

unipept-web-components

/unipept/unipept-web-components **2.0.0**

</> TypeScript, Vue, Vuetify

Both the Unipept Web and Desktop application are internally structured using web components, a type of reusable custom web element that is responsible for a well-defined function. This repository is a software library that contains a lot of these web components that are being used by both Unipept applications. Because of this library, we only need to make a change to our code once in order to update both apps simultaneously.

unipept-api

4.6.4

/unipept/unipept-api

</> Ruby-on-Rails

The Unipept API repository contains the "backend" code that powers all data analysis requests that are made by the Unipept Web and Desktop application. Unipept's API is responsible for instructing the Unipept database and extracting the required information out of it, processing it further and providing a valid response to API calls made by the Unipept Web app, the Unipept Desktop app, or other third-party applications.

unipept-cli

3.0.2

/unipept/unipept-cli

</> Ruby-on-Rails

The Unipept command line interface (CLI) can be directly instructed by machines (it works without a graphical user interface) and can therefor easily be plugged into existing data analysis pipelines. All code in this repository has been written in Ruby-on-Rails and is mainly responsible for parsing data from the command line, sending it to the Unipept API in chuncks and properly formatting the result. Ruby is an interpreted programming language that is compatible with all major operating systems, which makes the Unipept CLI easily accessible for most users.

unipept-visualizations

2.1.0

/unipept/unipept-visualizations

</> TypeScript, D3

An important aspect of the popularity of Unipept comes from its collection of interactive data visualizations. These are not only suitable for visualizing the results of a metaproteomics data analysis, but can instead be used in a wide range of different applications (as long as the input data format adheres to certain properties). In order to promote the reuse of our visualizations by third-party app developers, we have extracted all of Unipept's visualizations into a separate software package which is available through NPM.

make-database

/unipept/make-database

</> Bash, Java, JavaScript, SQL

A big portion of Unipept's performance is the result of a lot of preprocessing that takes place during the construction of the Unipept database. This database contains a list of all tryptic peptides that are generated from the information present in the UniProtKB. The digestion of the UniProtKB and all of the associated preprocessing steps are implemented in Java and JavaScript. All separate database construction steps are then "glued" together using a shell-script that finally produces a series of `tsv`-files that can be imported into a relational database.

docker-images

1.1.1

/unipept/docker-images

</> Docker

This repository contains a collection of Docker images that can be used to easily set up a local instance of the Unipept Database and the Unipept API. These images are important for the Unipept Desktop application to function since it allows researchers to build their own custom protein reference databases, which are then hosted locally through Docker containers based upon these images.

MegaGO

1.0.0.2021.04

/MEGA-GO/MegaGO

Python, Flask, TypeScript, Vue, Vuetify

The MegaGO project started as a hackathon at the EuBIC Developers meeting in 2020. MegaGO is a tool that allows to compute the similarity of multiple sets of GO-terms. This similarity can be computed by using our web application (which is implemented in TypeScript, using Vue and Vuetify) or the command line interface. Computation of the similarity score is not performed by the web application, but by a Python package that exposes a HTTP REST API (which is developed using the Flask framework).

Pout2Prot

1.2.1

/compomics/pout2prot

Python, JavaScript, Vue

Pout2Prot aims at performing protein grouping on files generated by the Percolator software. This repository contains a Python implementation of the protein grouping algorithm and the implementation of a web application that exposes the functionality of the Python script through an easy-to-use app. The Python implementation of the protein grouping script has been transpiled to JavaScript using Transcrypt[2] and is directly executed by the browser itself. No user data is ever transmitted to the Pout2Prot web server.

SharedMemoryDatastructures

0.1.9

/pverscha/SharedMemoryDatastructures

TypeScript

This repository contains an implementation of a HashMap that can be transferred between different JavaScript threads at a very low cost. This HashMap is still in an alpha-status (not all Map-operations are supported at this point), but is already being used by Unipept under-the-hood on a daily basis since 2021 and allows for a dramatic increase in performance.

[2] See: https://www.transcrypt.org/

Table of Contents

Chapter 1

Introduction

The research presented in this PhD thesis is situated at the overlap of computer science and biotechnology. In order to fully understand and grasp the concepts that are presented throughout this thesis, it is important that I first introduce a set of terms, definitions and techniques that are extensively used throughout this work.

1.1 Biotechnological concepts

1.1.1 The central dogma in biology

Every organism in our universe is made up of cells and each of these cells contains the instructions that define what the organism is, how it behaves and how new proteins or other products should be created. Each cell contains an exact copy of these instructions. These instructions can be compared with a recipe book that describes and instructs a cook to bake a specific kind of cake. All recipes are collected in this book, and every cell has an exact copy of this recipe collection. Whenever a client in the restaurants requests a dish, the request is sent to the kitchen where the cook selects the appropriate recipe from the recipe book and starts making the dish.

This is a simple analogy to explain how DNA and RNA are used throughout the cells of an organism to create new proteins (Figure 1.1). Our book of recipes represents the DNA that is present in every cell. RNA, on the other hand, can be regarded as a copy of a single recipe in the book, while the final dish itself corresponds to a single protein.

We can continue to use this analogy to explain exactly what the central

dogma of biology is and how it works. The first step in making a dish is to choose a recipe from the book, a process that's called **transcription**. In transcription, a section of DNA is copied into an RNA-sequence (more specifically messenger-RNA, or mRNA), which serves as a template for building proteins later on.

Next, all the necessary ingredients for the dish are gathered. These can be thought of as the raw materials that are needed to build a protein. For proteins specifically, these raw materials are amino acids, which are brought to the site of **protein synthesis** by a molecule called **transfer-RNA** (or tRNA).

Once all the ingredients have been delivered to the construction site, the cell follows the instructions in the encoded recipe (or thus the instructions encoded by a strand of messenger-RNA, or mRNA), during a process called **translation**. During the translation phase, this mRNA template is read by a ribosome which assembles all required amino acids in the correct order to form a protein.

The concept explained above, is called the **central dogma of biology**. It is one of the most fundamental principles of molecular biology and was first depicted in 1958 by Francis Crick who also reformulated it in a manuscript published in Nature in 1970 (Crick, 1970). The central dogma explains how proteins in organisms are constructed and is very important to grasp in order to understand a lot of the basic principles of biology.

1.1.1.1 DNA

DNA stands for **D**eoxyribo**N**ucleic **A**cid and, as explained above, contains the instructions on how all proteins in an organism can be constructed. A copy of the complete DNA of an organism is present in every cell of this organism and is organised in chemical structures that we call **chromosomes**. These so-called chromosomes mostly appear in pairs (humans, for example, have 23 pairs of chromosomes).

Figure 1.1: The central dogma in biology can easily be explained with the recipe book analogy. The DNA corresponds to a recipe book that contains the recipe for every dish that can be made. A copy of a single recipe corresponds to the concept of RNA and a dish corresponds to a single protein that's been completely assembled.

Chromosomes themselves are then further divided into specific regions that we refer to as **genes**. A gene is a well-defined region of DNA that corresponds to the instructions required to construct one specific protein (and thus corresponds with one recipe of our book of recipes analogy that we used earlier).

The DNA is made up of a sequence of nucleotides (simple molecules that can be chained together) that are always represented by the four letters A, C, G and T.

The famous double helix structure of DNA was first discovered by Francis Crick, James Watson, Rosalind Franklin and Maurice Wilkins after Franklin obtained images of DNA using X-ray crystallography. This discovery was done in 1953 and is of very big importance for the research that is presented in this work.

1.1.1.2 RNA

RNA stands for **R**ibo**N**ucleic **A**cid and is chemically very similar to DNA. Instead of Thymine, RNA uses its complement Uracil (represented by U), but the other three nucleotides A, C and G remain the same. Unlike DNA, RNA is found in nature as a single strand folded onto itself, rather than a paired double strand.

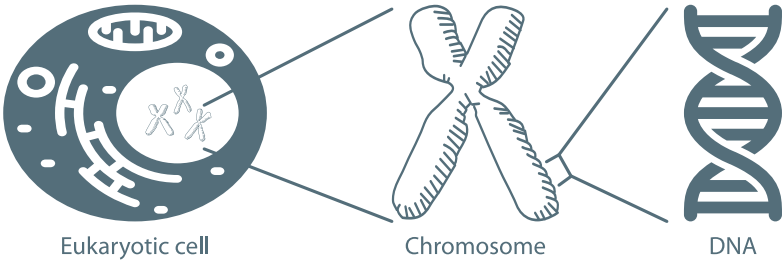

Figure 1.2: Relation between a cell, chromosomes, genes and DNA. All chromosomes are collected in the cell nucleus. A chromosome is a structure that consists of DNA and the DNA consists of well-defined pieces that we refer to as genes.

There exist different types of RNA that are each responsible for an important process in the creation of proteins. Cellular organisms (such as humans) use messenger RNA (mRNA) to convey genetic information that is subsequently used to direct the synthesis of specific proteins. A second important type of RNA is the transfer RNA (tRNA) which is used for bringing amino acids to the site where protein synthesis takes place. Lastly, a third important type is the ribosomal RNA (rRNA) that is responsible for chaining together amino acids to form finished proteins.

1.1.1.3 Proteins

Proteins are large, complex molecules composed of long chains of amino acids. Although hundreds of amino acids exist in nature, only 20 of them will appear in proteins. Every amino acid has its own chemical properties and is generally represented by a single capital letter: K stands for lysine, R stands for arginine, etc. It is the specific ordering of these amino acids that defines the function and structure of a protein.

Proteins can have a variety of different functions in an organism. They can act as enzymes (that help to catalyze chemical reactions), they can support the immune system, act as transporters, take care of communication between important processes (hormones), etc. In short, proteins are a

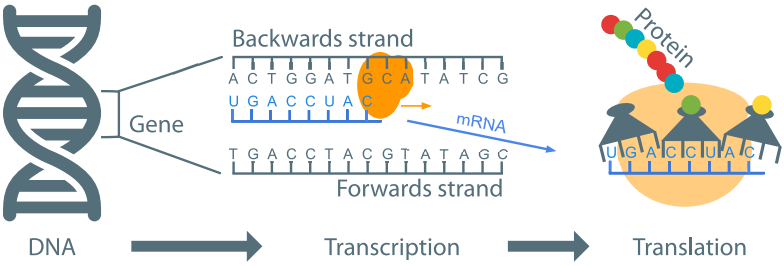

Figure 1.3: Schematic depiction of how information recorded in a gene is converted (or synthesized) into a protein. The DNA in a gene is first converted to mRNA (transcription). The mRNA will then be read in groups of 3 nucleotides at a time and matched with amino acids. The final sequence of amino acids constructed this way corresponds with a protein.

critical component of living organisms and play many important roles in maintaining health and supporting life processes.

A protein will be constructed from an mRNA strand by the **ribosomes**. These are large molecules that link amino acids together to form proteins in a process called protein synthesis. See Figure 1.3 for a schematic overview of the protein synthesis process.

1.1.2 (Meta)genome, (meta)transcriptome and (meta)proteome

The **genome** of an organism can be defined as the collection of DNA that is present in this organism. It defines which proteins can potentially be constructed by the organism. Secondly, the **transcriptome** of an organism is simply the set of all RNA transcripts in an organism and already provides a little more information about which parts of the DNA do actually encode proteins. Lastly, the **proteome** of an organism is the collection of all proteins that can be expressed by an organism.

In this thesis, we mainly discuss the terms **meta**genome, **meta**transcriptome and **meta**proteome. Instead of respectively refering to the collection of genes, transcripts and proteins that can be expressed by a single organ-

ism, the **meta** prefix denotes that we are respectively talking about the set of genes, transcripts or proteins that can be expressed by a **collection** of different organisms (typically of the same biological environment). See Figure 1.4 for a schematic display of the proteome and metaproteome.

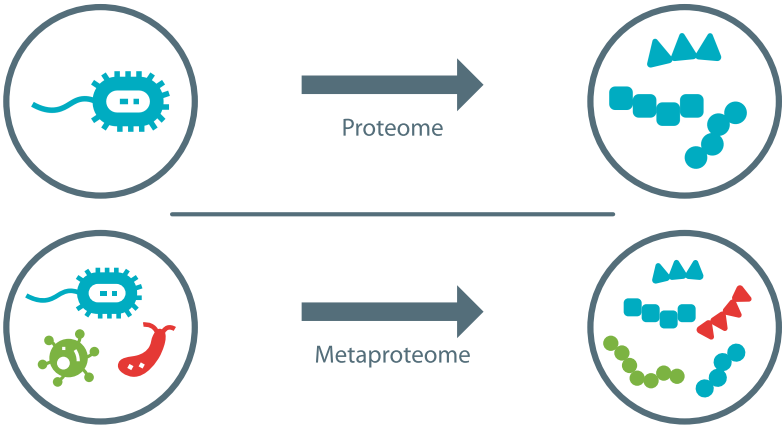

Figure 1.4: A schematic overview of the proteome and the metaproteome. The proteome is defined as the proteins that can be expressed by a single organism. The metaproteome, on the other hand, is then defined as the set of proteins that can be defined by a collection of organisms.

Since our DNA provides the instructions for all proteins that can possibly be expressed, it provides no suitable information about which proteins are really being expressed at a specific moment in time. By exploring the genome, it is thus possible to deduce what an organism is capable of doing, but not what it actually *is* doing right now. Not all pieces of an organism's DNA have a "meaning" or will lead to suitable proteins. Around 98% of the human genome is **non-coding**, meaning that these parts of the DNA will never be synthesized to a meaningful protein, but rather to regulatory sequences, non-coding genes or something that has not been discovered yet.

Studying the transcriptome of an organism has several advantages over studying the genome. First of all, it allows researchers to understand dy-

namic changes that can present themselves during the transcription process in a cell. The transcriptome reflects the dynamic changes in gene expression that occur in response to environmental cues, developmental stages, and disease conditions. In contrast, the genome remains mostly static, with relatively stable genetic sequences. As the transcriptome of an organism captures the expression of genes at a specific time and in a specific context, it provides more information to a researcher and it might help them to understand the underlying mechanisms of diseases, drug responses and other complex biological processes. Finally, the transcriptome can also be used to study the regulation of gene expression, including alternative splicing, post-transcriptional modifications, and non-coding RNA expression.

The proteome is the most informative piece of information and tells researchers which proteins are being synthesized by an organism at a specific point in time. It allows us to study the functional profile of an organism and goes one step further than studying the transcriptome.

1.1.3 Shotgun metaproteomics (analysing the metaproteome)

In this work, we focus on analysing the **metaproteome** of an ecosystem. We will first explain how proteins can be identified from a biological sample by using a very advanced device called a **mass spectrometer**. Currently, most researchers are using a technique called **shotgun proteomics** when analysing a protein sample and follow a predefined set of steps. Each of the different steps in shotgun proteomics (Figure 1.5) will be covered in detail in this section.

1.1.3.1 Protein digestion

Since a protein is typically a molecule that is too big to be analysed by a mass spectrometer, it first needs to be cut into smaller fragments or **peptides**. The process of cutting a protein into peptides is called **protein digestion** and is performed using a specific enzyme: trypsin. Other proteases (i.e. enzymes that can be used to digest proteins) exist, but trypsin is by far the most

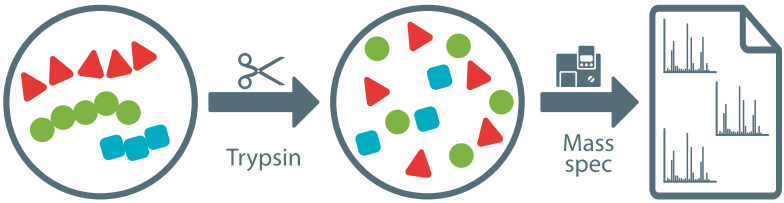

Figure 1.5: Overview of the different steps in shotgun metaproteomics. When processing an input sample using a shotgun metaproteomics pipeline, the proteins in the input sample are first digested by trypsin. The sample containing the remaining peptides is then fed into a mass spectrometer that produces a collection of observed mass spectra.

popular one since it digests and usually *cuts* the protein at a fixed position: whenever the amino acids lysine (K) or arginine (R) are encountered, the protein will most-probably be cleaved by trypsin (except when lysine or arginine are directly followed by proline (P) or because trypsin missed a cleavage site, which does happen sometimes). All peptides that are the result of a tryptic digest of proteins are called **tryptic peptides**. See Figure 1.6 for an example of a tryptic digestion of 2 proteins.

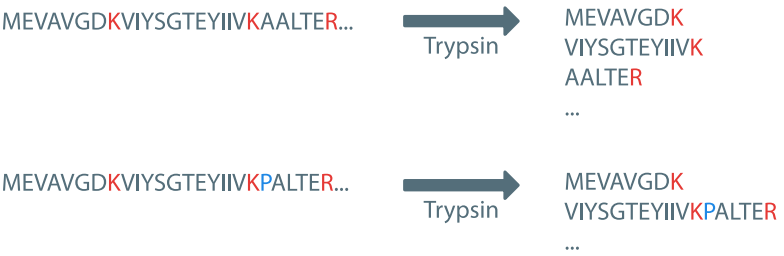

Figure 1.6: Digestion of two proteins using the protease trypsin. The amino acids depicted in red are either lysine (K) or arginine (R) and indicate the location where trypsin will cleave the protein (except if one of these is directly followed by proline (P). In the second protein there will be no cleave after the second occurrence of lysine, since the next amino acid is proline.

1.1.3.2 Mass spectrometry

Before we can start to explain the different steps in shotgun metaproteomics, we need to provide a basic understanding of the mass spectrometer. A mass spectrometer is a very advanced and expensive device that allows us to measure the "weight" of molecules. Instead of measuring the force of gravity on an object (which is what a traditional scale does), a mass spectrometer uses magnetic and electric fields to measure the **mass-to-charge** (m/z) ratio of the particles in the input sample.

These "mass-to-charges" for each of the particles that were found in the input sample can be visualised as a **mass spectrum** (see Figure 1.7). Each peptide that's fed into the mass spectrometer produces such a mass spectrum, which is not necessarily unique (i.e. different peptides can produce the same mass spectrum) and the following step in shotgun metaproteomics consists of mapping these mass spectra onto peptide sequences.

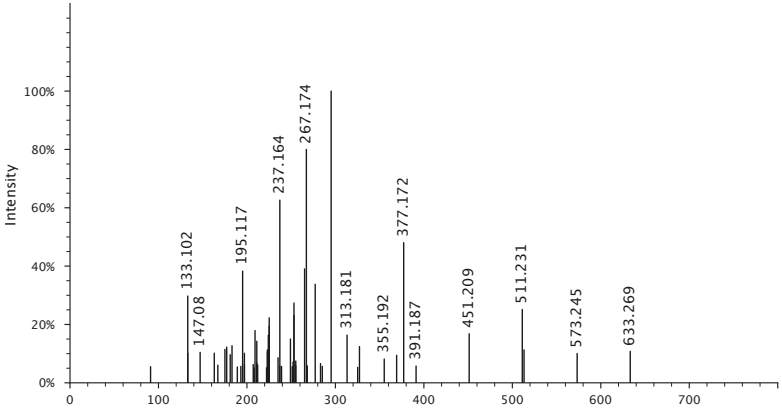

Figure 1.7: Example of a mass spectrum for the molecule "7,15-O-diacetyl-5-O-benzoyl-3-propanoyl-13,17-oxy-14-oxopremyrsinol", taken from (Bittremieux, 2020). The different peaks in the spectrum correspond with the observed "mass-to-charge" ratio of the different particles of the input sample.

1.1.3.3 Matching mass spectra with peptide sequences

Now, in order to map mass spectra back to peptide sequences, researchers use **search engines**. These search engines are complex software applications that contain a list of peptides and the corresponding expected mass spectra and that try to match the experimentally obtained spectra with the theoretically modelled mass spectra in their database.

How exactly this is done is out-of-scope for this work, but more information can be found in (Benchmark and improving methods in metaproteomics informatics, 2022). The most important thing to realize at this point is that the data that comes out of the mass spectrometer (i.e. the mass spectra) can be converted into the peptide sequences that most probably occur in the input sample. The CompOmics group at Ghent University, led by Prof. Lennart Martens, is specialised in the development of novel search engines, such as MS2Rescore.

Once a list of peptide sequences has been determined, the data is finally ready to be sent to Unipept for further downstream analysis (Figure 1.8).

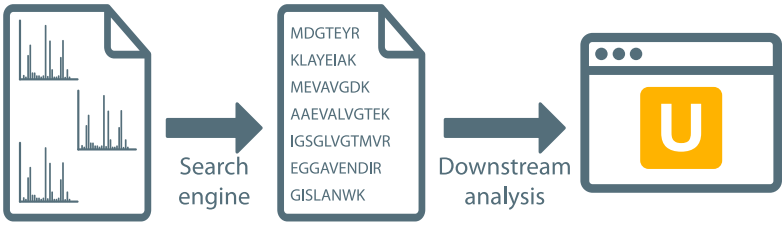

Figure 1.8: The mass spectra identified by a mass spectrometer can be mapped onto a list of peptides by a search engine. Finally, these peptides can be transferred to Unipept for further downstream analysis.

1.2 Unipept

Unipept is an ecosystem of software tools that are mainly focussed on the analysis of metaproteomics datasets. Prof. Dr. Bart Mesuere, co-supervisor

of this PhD thesis, initially started the Unipept project at Ghent University in 2010 in the context of his PhD. Since its early days, Unipept has undergone a big transition, while still maintaining its initial focus of providing an excellent user experience and top-of-the-line performance when compared to similar tools.

1.2.1 The Unipept ecosystem

All tools in the Unipept ecosystem work by taking a list of tryptic peptide sequences as input and matching these peptides with the proteins in a **protein reference database** in order to provide a taxonomic and functional summary for this input sample. Every Unipept tool has its own target audience and functionality. See the list in the next subsection for more information on the tools that currently exist and what each of these is aimed at.

1.2.1.1 Unipept Web application

The first real Unipept tool that was presented to the outside world, is the Unipept Web application. This app is accessible at https://unipept.ugent.be and provides a user friendly web component that allows users to perform metaproteomics analysis and consult the results that are presented to them as a collection of visualizations and tables. All information that is provided by our web application can easily be exported (and can subsequently be imported into other tools for further analysis).

1.2.1.2 Unipept CLI

For power-users that require the metaproteomics analysis of many large samples, we provide the Unipept command line interface (CLI). This command line interface allows Unipept's analysis to be plugged into existing analysis pipelines and allows machines to automate specific steps. Unlike the web application, the CLI has no graphical user interface and mainly provides textual output (support for visualizations is also limited).

1.2.1.3 Unipept API

The Unipept API (Application Programming Interface) is a collection of endpoints that allow third-party applications to integrate some of Unipept's functionality into their own workloads. Data can be sent to each endpoint using `HTTP POST` or `GET` requests to our servers, after which the Unipept server will respond with the requested results as a `JSON`-formatted object.

1.2.1.4 Unipept Desktop

Unlike the previous three tools, the Unipept Desktop application has been added to our ecosystem very recently and does not necessarily needs to communicate with the Unipept servers in order to function. The desktop application combines some of the advantages of the web app, CLI and API into one and allows users to process large metaproteomic samples using a user friendly graphical user interface (making it more accessible to less tech-savvy users). It is similar to the CLI in the way that it allows to process much larger samples than the web application, and requires less technical skills to operate. The biggest difference with the web application is the fact that it cannot be plugged into existing analysis pipelines as easily, but it's also not designed to do so.

1.2.1.5 UMGAP (Unipept MetaGenomics Analysis Pipeline)

UMGAP, or the Unipept MetaGenomics Analysis Pipeline, is a bit of an outsider because it focuses on the analysis of metagenomics data (instead of metaproteomics data). This pipeline has been developed by Dr. Felix Van der Jeugt and is only accessible from the command line. Since all tools presented in this thesis focus solely on the analysis of metaproteomics data, we will not go into more detail here.

1.2.2 The Unipept metaproteomics analysis pipeline

In this section, we are going to take a look at how exactly a metaproteomics data sample is processed by Unipept. Every input sample consists of a

list of tryptic peptide sequences. Non-tryptic peptides will be ignored by Unipept, since they cannot be matched with the information in our **peptide reference database**.

1.2.2.1 Construction of the peptide reference database

Grasping how a peptide reference database is being used by Unipept is already one of the most important things that help to understand how Unipept processes input samples and generates a taxonomic and functional profile for an ecosystem. A peptide reference database can be seen as some kind of information resource that maps peptides onto organisms and functions, associated with these peptides. In order to construct this database, we extract all proteins (including their mapping to organisms and functions) from the UniProtKB resource (The UniProt Consortium, 2019). UniProt is an organization that focuses on providing a protein database that is as complete as possible. It contains full protein sequences and information about the organisms and functions that these proteins are associated with.

Since Unipept expects a user to provide a list of peptides (instead of proteins), we need to transform the information from the UniProtKB resource into a peptide reference database. Internally, Unipept does so by performing an in-silico tryptic digest of the proteins encountered and cleaves them by the rules imposed by trypsin (see Figure 1.9). Note that one tryptic peptide typically matches with more than one protein. This makes sense since the proteins from organisms that have co-evolved closely over the years can be very similar to each other and that some of the peptides that result from tryptic digestion of these proteins are identical. This actually happens relatively frequently and can even occur between organisms of very different lineages (sometimes purely due to chance).

Since users only provide peptides to Unipept (and no other information), it is impossible to distinguish between the different proteins that match with one of the peptides from the input (e.g. if an input peptide appears in three

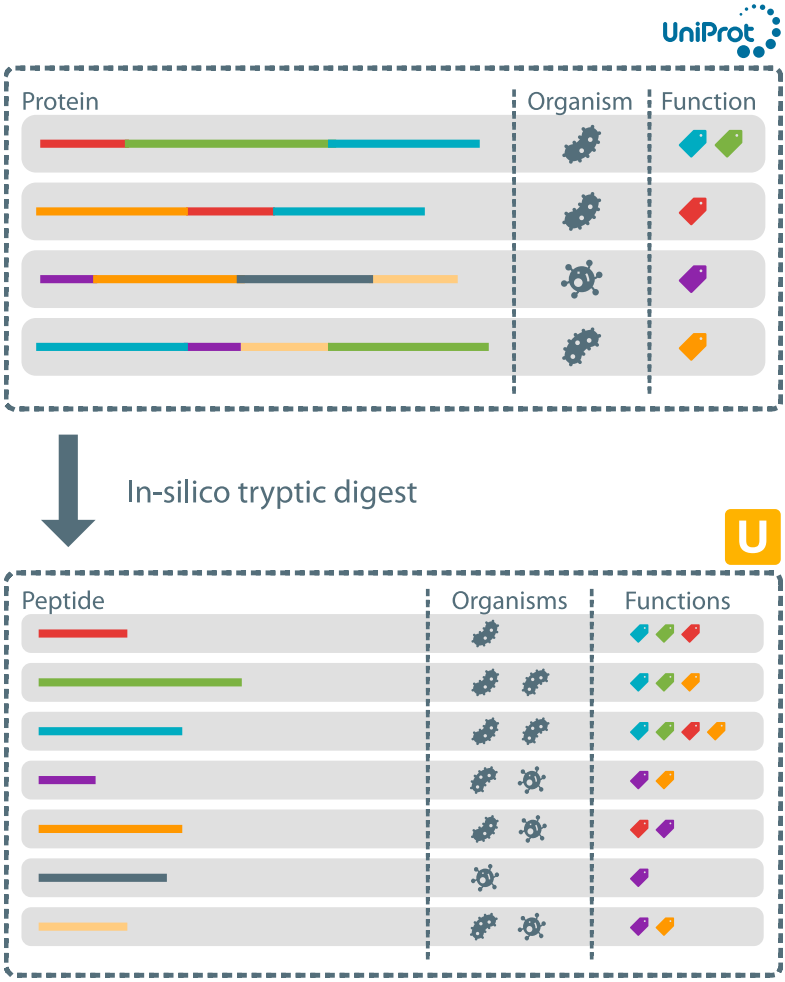

Figure 1.9: The Unipept database is constructed by extracting the proteins from the UniProtKB resource and performing an in-silico tryptic digest on each of the proteins.

different proteins A, B and C, it is impossible for Unipept to know if it should draw conclusions from protein A, B or C). In order to overcome this issue, we simply report only information that is applicable to all matched proteins (or thus information that is correct in either the situation that the peptide originated from either protein A, protein B or protein C in the example above).

Summarizing taxonomic information

For taxonomic information, we aggregate all organisms that are associated with the protein matches of a single peptide and compute the lowest common ancestor for this set of organisms. This lowest common ancestor is the most specific taxon in the NCBI taxonomy that is a direct or indirect parent of all organisms in the collection. In Figure 1.10, the lowest common ancestor of the organisms in the input set is the root node. This example is one that occurs relatively frequently and in which case no valuable information can be reported by Unipept (apart from the fact that an organism was indeed found). Compare this example to the one given in Figure 1.11. In this case, the lowest common ancestor of the two viruses from the input set is the *Coronaviridae* taxon.

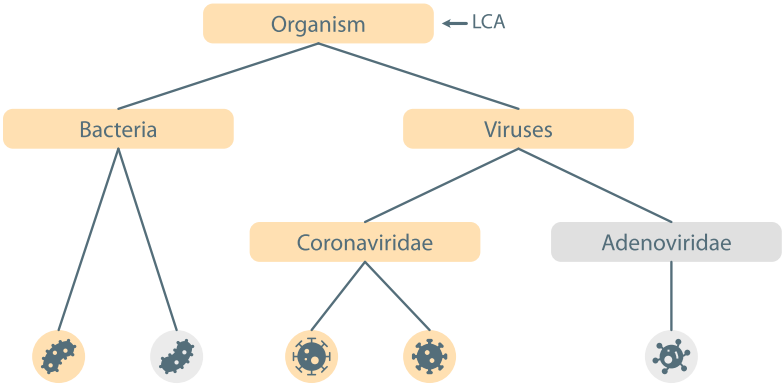

Figure 1.10: If the organisms in orange are selected, the lowest common ancestor of these organisms in the NCBI taxonomy will be the root node (organism), because this is the most specific node that is a parent of all organisms in the collection.

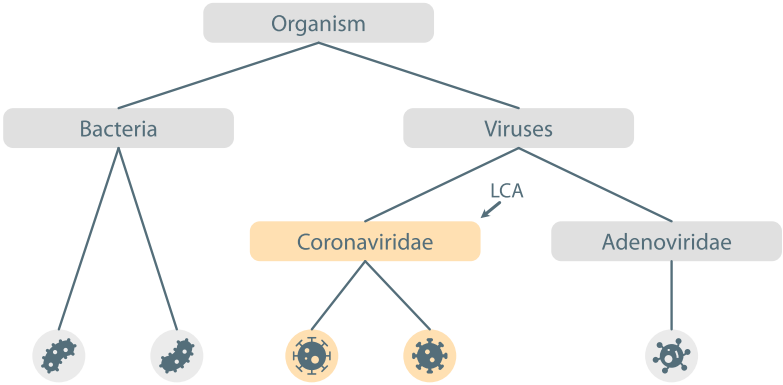

Figure 1.11: If the organisms in orange are selected, the lowest common ancestor of these organisms in the NCBI taxonomy will be *Coronaviridae*, because this is the most specific node that is a parent of both organisms.

Instead of thus reporting all organisms that are associated with a peptide, Unipept reports the lowest common ancestor of these organisms (as demonstrated in the examples above).

Summarizing functional information

In order to summarize the functional information of a single peptide, Unipept simply counts how many times each functional annotation appears in the matched set of proteins. To make this more concrete, if a peptide occurs in proteins A, B and C, and function X is associated with protein B and C, and function Y is associated with protein A, then function X will be reported as occurring twice for this peptide and function Y will be reported to occur once.

1.2.2.2 Processing a metaproteomics dataset

In order now to process a list of peptides, Unipept will match each of the input peptides with its peptide reference database and report all lowest common ancestors and functional annotations found per peptide. For a single sample, it will then provide something in the line of "species x was

found to occur in 12 out of 153 total peptides" (and it will do so for every identified taxon). This information is not only presented in a textual fashion, but will also be rendered by a collection of interactive data visualizations (which have been designed and implemented in-house (Verschaffelt *et al.*, 2022)).

Part I

Unipept Desktop

On February 3, 2011, the first version of the Unipept Web application was released. This date marked the start of Unipept as a tool that could be used by researchers outside of Ghent University. The web application has undergone an enormous evolution since then and has been expanded with numerous new tools and features in order to aid researchers in analysing metaproteomics data. Over the years, the Unipept ecosystem has further been expanded with an API that allows our analysis pipelines to be integrated with third-party tools and a command line interface (CLI) that can be used to automate the analysis of large datasets. Lately, the size and complexity of metaproteomics datasets have both increased, making it harder for a web application to process and produce the expected results. Since web applications are managed by a web browser, access to compute resources and storage space on the user's local machine is limited (to ensure stability and security of the web apps managed by the browser).

In order to power the next generation of metaproteomics data analysis, I started working on the Unipept Desktop application. This application is developed using the Electron framework[1] which allows us to use web technologies (such as JavaScript, HTML, CSS, etc.) to build desktop applications. Since we already have a functioning web application, we want to reuse as much of the work we have put into this for the new desktop application (which is why Electron is the perfect choice). Most of the rationale and reasoning behind this is explained in Chapter 2.

Once the desktop application reached feature parity with the web application, I started on expanding it with new features that could not be realised with our web application. The most important of these features is the ability to build targeted protein reference databases, which can increase the taxonomic and functional resolution of a metaproteomics or proteogenomics data analysis. Proteogenomics is a novel research discipline that utilizes the findings of a metaproteomics data analysis to guide a subsequent metaproteomics analysis. Unipept Desktop 2.0 was released on February

[1] https://www.electronjs.org/

10, 2023, and is the first release of our desktop application that supports all of these new features. Proteogenomics, and how the construction of targeted protein reference databases was realised, is discussed in Chapter 3.

Chapter 2

Unipept Desktop: a faster, more powerful metaproteomics analysis tool

I first started working on the Unipept Desktop application in August of 2019 (I started my PhD around the same time). After an extensive period of testing, version 1.0 of the Unipept Desktop application was released on January 14, 2021. It improves upon the web application in a number of different ways and allows the Unipept ecosystem in particular to keep up with the constant increase in metaproteomics sample sizes. Philippe Van Thienen, the first master student that I guided, helped me to restructure our web application into web components (promoting maximal reuse of the code between the desktop app and the web app). Everything that has been improved, or that has changed, between the Unipept Web application and the Unipept Desktop application, is discussed in this chapter.

I have been the lead developer of this desktop application since the start and once version 1.0 was ready for release, I started working on an application note for the Journal of Proteome Research. This application note is included and discussed in this chapter.

This chapter contains a verbatim copy of the application note by (Verschaffelt, Van Den Bossche, Martens, et al.*, 2021) as published in Journal of Proteome Research.*

Abstract — Metaproteomics has become an important research tool to study microbial systems, which has resulted in increased metaproteomics data generation. However, efficient tools for processing the acquired data have lagged behind. One widely used tool for metaproteomics data interpretation is Unipept, a web-based tool that provides, amongst others, interactive and insightful visualizations. Due to its web-based implementation, however, the Unipept web application is limited in the amount of data that can be analyzed. In this manuscript we therefore present Unipept Desktop, a desktop application version of Unipept that is designed to drastically increase the throughput and capacity of metaproteomics data analysis. Moreover, it provides a novel comparative analysis pipeline and improves the organization of experimental data into projects, thus addressing the growing need for more performant and versatile analysis tools for metaproteomics data.

2.1 Introduction

Metaproteomics is a relatively young research field that focuses on the study of microbial environments and complex ecosystems, and of the interactions between the organisms involved, through the analysis of the proteins extracted from these environments. Over the past years, the technology to identify proteins from such complex samples has been greatly improved, allowing metaproteomics to transition from relatively small studies to large scale experiments (Rechenberger *et al.*, 2019; Wilmes *et al.*, 2015). The key enabling technologies for this transition are improved mass spectrometers and more powerful proteomics approaches, which have both come a long way since the introduction of metaproteomics analysis in 2004 (Rodríguez-Valera, 2004; Yates, 2019). To allow efficient processing of the resulting increase of acquired data, various dedicated tools have been made avail-

able to support metaproteomics data analysis (Muth *et al.*, 2015; Van Den Bossche *et al.*, 2020), but even with this increased bioinformatics support, many challenges still need to be overcome, especially regarding downstream analysis of the obtained identifications (Schiebenhoefer *et al.*, 2019).

Unipept is a leading tool for such downstream metaproteomics data analysis (Herbst *et al.*, 2016) that currently consists of a web application (Gurdeep Singh *et al.*, 2019), a web service, and a command line tool (Verschaffelt *et al.*, 2020). The Unipept web application provides users with the ability to analyze a metaproteomics sample and extract taxonomic and functional information from environmental samples derived from a variety of origins, ranging from the human gut to biogas plants. The Unipept web application provides users with interactive visualizations and allows them to, for example, filter out all functions that are associated with a specific taxon. Due to its web-based nature, however, the size and number of samples that can be analyzed by Unipept are limited. And while it is currently possible to analyze larger data sets using the Unipept CLI, this requires more sophisticated bioinformatics skills and does not provide the interactive link between taxa and functional annotations.

Because of the browser limitations, it can already take a substantial amount of time to process relatively small samples (e.g. containing up to a few thousand identified peptides) using Unipept, depending on the specific search configuration used. These limitations have become an issue, as the advances in metaproteomics have not only increased data set sizes, but have also increased the number of data sets that need to be processed (Zhang and Figeys, 2019).

In order to accommodate this evolution, the throughput of metaproteomics data analyses needs to increase as well, in turn requiring tools that are not constrained in the amount of memory and CPU resources they are allowed to consume. Moreover, analysis results also need to be retained for future reference, ideally in a project-based approach that can group multiple samples, and the corresponding results should be easily shareable

with other researchers.

For specific applications, it is also important that all data is processed offline or on-site rather than being sent over the internet. For instance, sensitive medical data is often not allowed to be sent to external services for processing, but must be kept in-house to safeguard patient confidentiality and privacy.

All of the above issues need to be resolved in order to support the growing interest in, and reach of, metaproteomics. We therefore here present the Unipept Desktop Application, a novel cross-platform desktop application designed to specifically overcome these challenges while also retaining the functionality that exists in the current web app.

2.2 Implementation

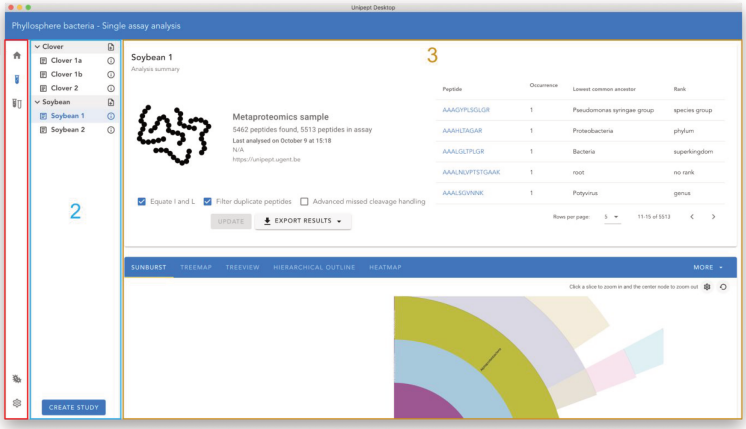

Figure 2.1: Screenshot of the Unipept Desktop application. The analysis page of the desktop application is depicted here and consists of three main parts: the sidebar that is used to navigate between the different analysis pipelines and functions of the application (1), the project explorer that displays a hierarchical view of the project (2) and the content view that renders analysis results (3).

The Unipept desktop application provides three different types of analyses: *i*) single assay analysis, *ii*) inter-assay comparative analysis, and *iii*) tryptic peptide analysis. The single assay analysis performs a full taxonomic and functional analysis of a single assay and corresponds to the default "metaproteomics analysis" as presented by the Unipept web application. The inter-assay comparative analysis on the other hand, provides the ability to explore similarities and differences between multiple assays. While the comparison of multiple assays was already possible with the Unipept web application, this was only available for a limited number of quite small assays due to strict memory constraints posed by web browsers. The tryptic peptide analysis, lastly, can be used to look up which proteins, taxa and functions are associated with a given peptide.

Unipept Desktop delivers these core functions through a concise user interface (Figure 2.1) that consists of three main parts: the sidebar, the project explorer, and the content view. The sidebar on the far left allows the user to navigate between the different analysis pipelines and functions of this application. Directly to the right of the sidebar is the project explorer that allows the user to switch between assays, and to modify the project. The project explorer is only shown when performing single assay or comparative analyses. Assays and studies can be renamed or deleted by right clicking them, after which a context menu opens. Lastly, the content view takes up most of the application's visual space and presents either analysis results or the settings page.

The Unipept Desktop Application also allows offline analysis of data through a choice of the API endpoint in the settings menu. This endpoint, which uses the Unipept API and by default connects to the online Unipept system, can be configured to call any service that supports the Unipept API. By setting up a local instance of the Unipept backend system, the user can thus ensure that all data remains locally. Setting up a local Unipept back-end is possible by cloning the open source Unipept repository on GitHub, but requires advanced technical knowledge. We plan to make

the installation process of these custom API endpoints even easier with future releases of Unipept.

Unipept Desktop is powered by the cross-platform Electron framework, which in itself is powered by Chromium browser technology. This means that the application is developed with web-centric technologies, such as the Vue frontend framework and TypeScript, and hence we were able to reuse large parts of the web app's codebase. The choice for the Electron platform was mostly driven by the extensive suite of different functionalities that can be integrated with minimum configuration efforts. Thanks to the Electron platform we can provide an automatic update mechanism, easily generate installation packages for all major platforms (Windows, macOS and Linux), and include automatic crash reporting, amongst others. Once installed, the Unipept Desktop application can thus update fully autonomously in the background, ensuring that users always have the latest functionality and bug fixes installed.

2.2.1 Project-centric analysis

The Unipept Desktop Application has full access to the local filesystem. Hence, it can store an arbitrary amount of data and does not need to worry about strict size limits; this in contrast to web applications that are only allowed to store up to a few megabytes using the local storage API. This allows us to improve upon the organization of data sets by introducing project-based data management capabilities. In accordance with the terminology introduced by the ISA-tab standard for experimental metadata annotation (Sansone *et al.*, 2012), we now refer to a data set derived from a sample as an "assay", while a study is a grouping of multiple, related assays, and a Unipept project represents a collection of such studies.

On the file system, a project is stored in a single folder that contains an SQLite database file, a subfolder for each study and one text file per assay, located in the subfolder of the corresponding study. This folder can be modified outside of the application, using the default file explorer

application of your operating system, thus providing maximum flexibility. All changes made to this project folder are automatically detected and imported by the application, granting users the ability to mass import assays and edit project properties with external applications. The application accepts simple text files with one peptide per line. In order to quantify peptide occurrence, a peptide can be included more than once in this file and the "filter duplicate peptides" option should be disabled for the analysis.

Because projects are folder-based, they can contain both the raw input data as well as the analysis results for an assay, making it practical for users to share projects with each other, for instance, in the form of compressed project folders. In addition, previously performed analyses do not need to be recomputed when the application is restarted, as opposed to analyses that were run on the Unipept website, which need to be recomputed every time the website is closed.

2.2.2 Comparative analysis

The Unipept Desktop Application provides both intra-assay and inter-assay comparative analyses that are rendered as heatmap visualizations. The intra-assay comparison can be started from the single assay analysis page by selecting the heatmap tab and provides a wizard to guide users through the set-up process of the comparison (Figure 2.2). Users are required to select two types of data sources (one for each axis of the heatmap) and indicate which items should be compared. Four different data sources are currently supported: NCBI taxa, GO terms (The Gene Ontology Consortium, 2019), EC numbers and InterPro entries (Finn *et al.*, 2017).

The inter-assay comparative analysis is designed to visualize differences and similarities in functional or taxonomic composition of multiple assays. Here too, users are presented with a wizard that is similar to the one found in the intra-assay comparison. For inter-assay comparisons, however, the horizontal axis of the heatmap is reserved for the set of selected assays, and users can therefore only select one collection of items that should be

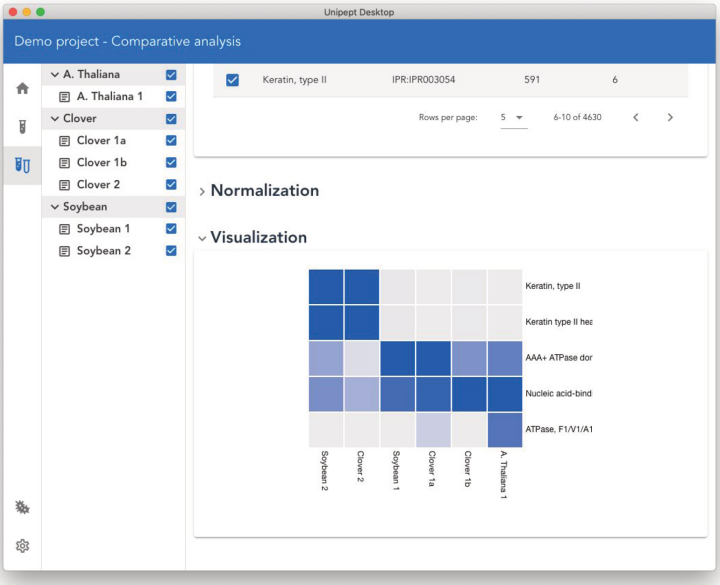

Figure 2.2: Screenshot of the inter-assay comparative analysis pipeline. Note that it is possible to select multiple assays from the project explorer. A heatmap is constructed from the set of items that were selected for comparison at the top of the page.

compared between the different assays.

Because the number of peptides can drastically differ between multiple assays, three different normalization techniques are provided to the user. The default setting normalizes the heatmap globally, i.e. the minimum and maximum values over the complete grid are computed and all grid values are normalized with respect to these values. The other two normalization techniques also normalize based on minimum and maximum values, but restricted within a row or column, respectively.

It is worth noting that, while the comparative analysis pipeline was originally designed for the Unipept Desktop Application, a slimmed-down version has meanwhile also been integrated into the Unipept web app.

With the advent of the Unipept Desktop Application, users now have a variety of ways in which they can use Unipept. A comparison between the various functionalities offered by these different services is provided in Table 2.1.

2.3 Conclusion

Unipept Desktop is a novel desktop application that extends upon the Unipept web application by eradicating the strict limitations posed by the web-based nature of this application to increase metaproteomics data analysis throughput. Moreover, the Unipept Desktop Application adds new features such as allowing users to structure their data in a hierarchical project-based system, to keep track of their analysis results, and to share or distribute these results very easily. Whereas the Unipept web application is limited to assays with up to 50 000 peptides, the Unipept Desktop Application supports assays containing one million peptides or more. For reference, the desktop app can analyze between 250 and 2000 peptides per second (without advanced missed cleavage handling enabled), depending on the type of assay that's being analyzed.

	desktop app	web app	CLI	API
visualizations	✔	✔	–	–
basic metaproteomics analysis pipeline	✔	✔	✖	✖
tryptic peptide analysis pipeline	✔	✔	✖	✖
comparative analysis	✔	–	✖	✖
metadata or projects	✔	✖	✖	✖
custom endpoint	✔	✖	✔	✖
store analysis results	✔	✖	–	✖
process large samples	✔	✖	✔	✔
no command line knowledge required	✔	✔	✖	✖
no installation required	✖	✔	✖	✔

Table 2.1: Comparison of the functionalities provided by the different Unipept services.

In a future release of the Unipept Desktop Application, we plan to provide support for the preparation of custom reference databases and further improve support for offline analysis. This will allow us to gradually evolve to a tool that is not only suitable for metaproteomics data analysis, but also for novel proteogenomics analysis techniques for complex environmental samples.

Our choice for the Electron framework proves to be very valuable as well, as a large portion of Unipept's codebase can thus be shared between the new desktop application and the existing web application. This in turn allows us to easily migrate (a slimmed-down version of) specific desktop features to the web app, and vice versa.

2.4 Availability

The source code for Unipept Desktop is open source and provided under the MIT license as a repository on GitHub: https://github.com/unipept/unipept-desktop. Pre-generated installers for Windows, macOS and Linux (AppImage format) can be downloaded from the release page of our GitHub repository. Installation instructions and documentation for the Unipept Desktop Application can be found on our website: https://unipept.ugent.be/desktop.

2.5 Acknowledgements

This work was supported by the Research Foundation—Flanders (FWO) [1164420N to P.V.; 12I5220N to B.M.; 1S90918N to T.V.D.B.; G042518N to L.M.].

Chapter 3

Unipept Desktop 2.0: construction of targeted reference protein databases for proteogenomics analyses

On February 10, 2023, version 2.0 of the Unipept Desktop application was released. A lot of work has gone into the development of this new version since it first introduced the concept of targeted protein reference databases. These targeted databases contain only those proteins that are associated with a list of organisms that is provided by the user. This not only increases the taxonomic and functional resolution of a metaproteomics analysis (by decreasing the chance of random protein matches), but also provides basic support for proteogenomics analysis.

I have taken the lead in the creation of the manuscript that I included in this chapter. The case study that's presented here could not have been written without the fantastic help of Dr. Alessandro Tanca and Dr. Marcello Abbondio of the University of Sassari, Italy. Both of them helped me validating the results produced of the latest iteration of the Unipept Desktop application and performed a lot of in-depth analyses to figure out where possible inconsistencies were coming from.

This chapter contains a verbatim copy of the manuscript by (Verschaffelt et al.*, 2023) as submitted to the bioRXiv preprint server.*

Abstract — Unipept Desktop 2.0 is the most recent iteration of the Unipept Desktop tool that adds support for the analysis of proteogenomics datasets. Unipept Desktop now supports the automatic construction of targeted protein reference databases that only contain proteins associated with a predetermined list of taxa. This improves both the taxonomic and functional resolution of a metaproteomic analysis and yields several technical advantages. By limiting the proteins present in a reference database, it is now also possible to perform (meta)proteogenomics analyses. Since the protein reference database now lives on the user's local machine, they have complete control over the database used during an analysis. Data does no longer need to be transmitted over the internet, decreasing the time required for an analysis and better safeguarding privacy sensitive data. As a proof of concept, we present a case study in which a human gut metaproteome dataset is analyzed with Unipept Desktop 2.0 using different targeted databases based on matched 16S rRNA gene sequencing data.

3.1 Introduction

The metaproteomics research discipline has undergone a big transition since the term was first introduced in 2004 (Wilmes and Bond, 2004). We have witnessed the evolution of metaproteomics from very small-scale experiments, in which three distinct proteins (Ram *et al.*, 2005) could be identified in an ecosystem, to a mature technology that is able to analyze more than 100 000 protein fragments (or peptides) from various environments.

Unipept (Gurdeep Singh *et al.*, 2019) was one of the first major tools in this promising new research field that could be used to analyze tryptic peptide-based metaproteomics samples. Originally starting as a web application, Unipept was quickly accompanied by an application programming interface (Mesuere *et al.*, 2016) (API) and command line interface (Verschaffelt *et*

al., 2020) (CLI) that respectively allow for embedding Unipept's analyses in other tools and analyzing larger samples directly from the command line. API usage metrics currently indicate that more than 500 000 requests are handled by Unipept on a monthly basis, acknowledging the importance of the tool in this field.

The advent of recent technological improvements in mass spectrometry and more powerful proteomics approaches have allowed metaproteomics to transition from small studies to large scale experiments (Wilmes *et al.*, 2015; Rechenberger *et al.*, 2019). Due to Unipept's inherent web-based nature, it was limited in the size of the samples that could be analyzed because of browser-imposed restrictions on available compute resources. This led to the development of the Unipept Desktop application (Verschaffelt, Van Den Bossche, Martens, *et al.*, 2021) in 2020.

Unipept Desktop (Verschaffelt, Van Den Bossche, Martens, *et al.*, 2021) paved the way for the analysis of large metaproteomics samples (containing 500k peptides or more) and completely overhauled the way these metaproteomics samples can be organized with the introduction of projects and studies. Projects can easily be shared with other researchers, who no longer need to reanalyze samples and wait for the results to become available, or can be archived for later use. The Unipept Desktop application also introduces a new inter- and intra-sample comparison pipeline which allows users to gain insight into the taxonomic and functional shift within and between multiple samples.

Over the last few years, interest in a new research area, proteogenomics, has been growing. Proteogenomics can be thought of as a logical next step in researching complex microbial ecosystems and combines information from both metagenomics and metaproteomics experiments in order to overcome a few key problems that arise when working with metaproteomics data in isolation (Schiebenhoefer *et al.*, 2019).

The first major issue that we need to consider is the ever-growing size

of the protein reference databases that are being used to match peptides with proteins. UniProtKB (The UniProt Consortium, 2021, 2019), a freely accessible database containing protein sequences, has seen a rapid increase in size over the last decade and has grown from approximately 19 million proteins in 2012 to 227 million proteins in 2022. Compared to the early days of Unipept, we are able to identify increasingly diverse species as a direct result of the increased size of the reference database, but this also comes with a few drawbacks. Each peptide that is presented to Unipept will be matched with all proteins in which this peptide occurs. All of these proteins are associated with a specific organism and such a peptide-based analysis thus results in a set of organisms from which this peptide could potentially originate. In order to increase insight of researchers into the taxonomic composition of a sample, Unipept summarizes all of this information and calculates the lowest common ancestor (LCA) of this set of organisms (i.e., NCBI taxa) for each peptide. If all of the matched organisms are evolutionarily close to each other, this works very well and the LCA of our matches will be of value. If, however, one or more of the matched organisms is very different from the others, the LCA will typically end up at the root or another very general taxon within the NCBI taxonomy (Figure 3.1).

Proteogenomics tries to overcome this problem by combining information from prior metagenomics experiments from the same environment with metaproteomics experiments. The metagenomics experiment is used to explore the taxonomic composition of an ecosystem, which subsequently guides the researcher to query only a subset of the reference database. In the case of shotgun metagenomics, DNA sequences identified by a metagenomics experiment can be used to build a customized protein reference database (Tanca *et al.*, 2016).

When using the Unipept Web application, analyses are always performed against the entire UniProtKB resource. We previously added the possibility to set up a local instance of Unipept and search against the database provided

by this endpoint as part of the Unipept Desktop application, but setting up such a custom endpoint is often experienced as a big technical hurdle by our target audience. Most researchers are thus still dependent on the reference database provided by Unipept, including the database update scheme that Unipept dictates. This creates a set of problems and drawbacks that need to be overcome in order to properly support proteogenomics data analyses. Since users do not have control over the database that is being used, they cannot provide potential metagenomics information (such as taxa identified in the ecosystem under study) and restrict the search space of the reference database.

To solve this problem, we introduce version 2.0 of the Unipept Desktop application, which marks the beginning of a new era for the analysis of proteogenomics datasets. Unipept Desktop now provides support for the automatic construction of targeted protein reference databases on the user's local machine. Such targeted databases are based on a filtered version of UniProtKB and only contain UniProtKB records that are associated with the taxa provided by the user (Figure 3.2).

In this article, we discuss how these targeted protein reference databases are constructed by the Unipept Desktop application and how they can be queried efficiently on a user's machine. We also present a case study, based on a human gut metaproteome dataset obtained from 28 celiac patients, in which we investigate to what extent the accuracy of the analysis results improves by only taking into account a subset of the UniProtKB reference database.

3.2 Unipept Desktop 2.0

In order to perform a proteogenomics experiment, researchers typically first perform a metagenomics experiment on the environment of interest, which then provides them with a set of organisms that are likely to be present in that environment. 16S rRNA gene sequencing and shotgun

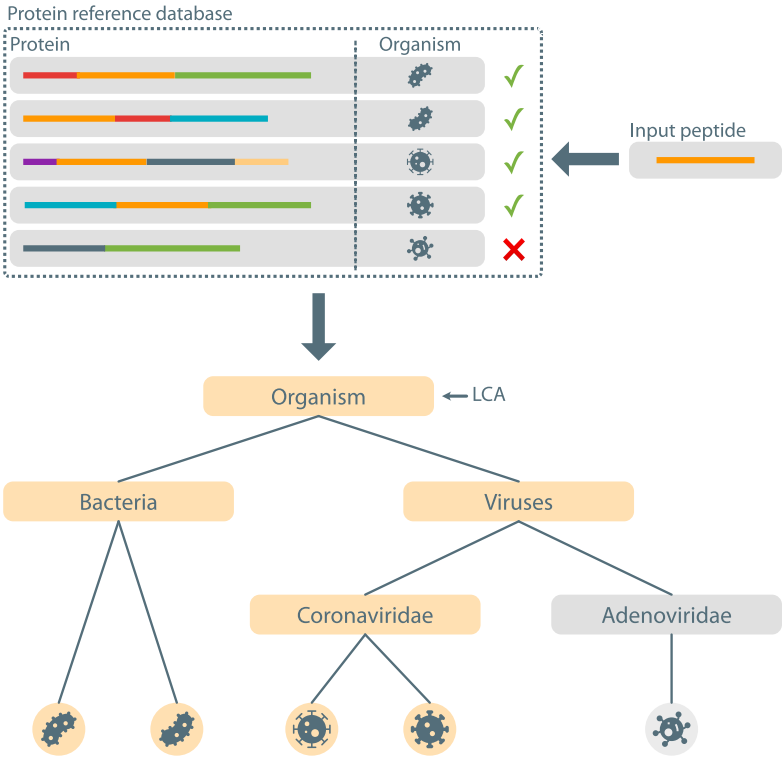

Figure 3.1: An example of how the lowest common ancestor (LCA) for a set of identified taxa is computed in Unipept Desktop, using an unfiltered protein reference database. A single input peptide is matched against proteins in a reference database. The taxa associated with all matched proteins are then summarized as the LCA, which is the most specific node in the taxonomy tree that is a parent of all matched taxa. An unfiltered protein reference database is used in the matching process. Since more proteins from a more diverse range of species are matched, the LCA ends up at the root of the taxonomy tree, providing little to no information. b) The reference database is restricted to viral proteins only and a much more specific LCA will be found (mapping onto the Coronaviridae family).

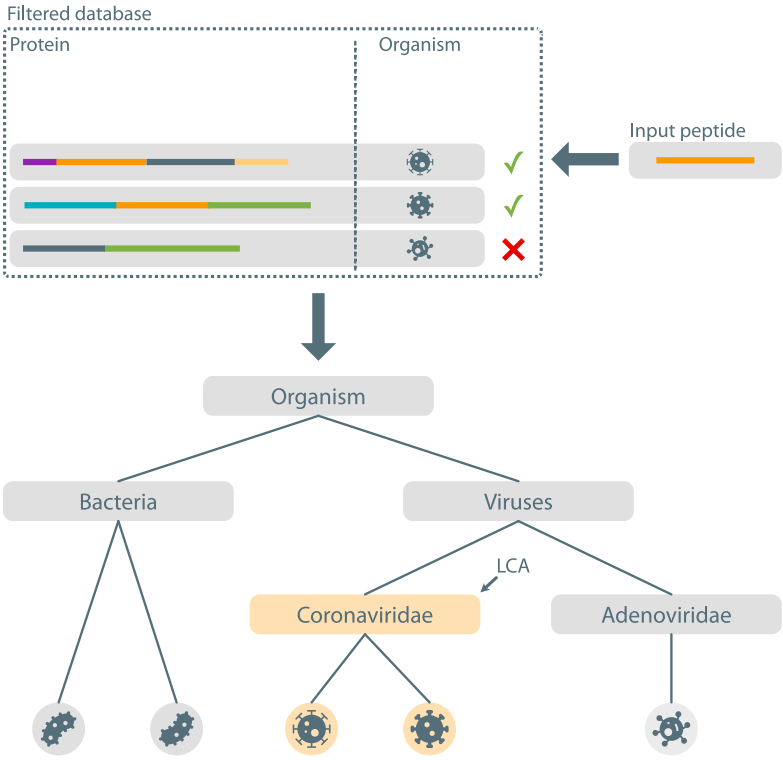

Figure 3.2: An example of how the lowest common ancestor (LCA) for a set of identified taxa is computed in Unipept Desktop, using a filtered protein reference database. The reference database is restricted to viral proteins only and a much more specific LCA will be found (mapping onto the Coronaviridae family). This example illustrate the importance of targeted protein reference databases when analyzing metaproteomics samples.

metagenomics are two widely used techniques in this research area that provide the required information, but other types of analysis can also be used. Sometimes the taxonomic composition of an environment can also be inferred from previous studies, if available.

Once the approximate taxonomic composition of a specific sample is known, a targeted protein reference database can be created that contains only those proteins that can be produced by the organisms of interest. This helps to drastically reduce the search space for a subsequent metaproteomics analysis (typically performed on the same environment) that is performed using the newly constructed targeted protein reference database.

Version 2.0 of the Unipept Desktop application is fully focused on the analysis of these proteogenomics datasets and therefore introduces an exciting new feature that allows users to construct targeted protein reference databases in two different ways.

To construct a targeted protein reference database with Unipept Desktop 2.0, researchers need some way of selecting UniProtKB proteins that can be present in the final result. The first selection method allows to specify which proteins are retained by providing a list of valid NCBI taxon identifiers. Unipept will then only include those proteins in the final database that are associated with a taxon that is present in this list, or that are associated with an (in)direct child of one of the provided taxon identifiers. Note that there's also the option to limit the database construction process to SwissProt (instead of SwissProt + TrEMBL) such that only manually-curated proteins are included. The second selection method requires the user to provide a set of UniProtKB reference proteome identifiers. Only the proteins that are found in these reference proteomes will then be selected for the construction of the reference database.

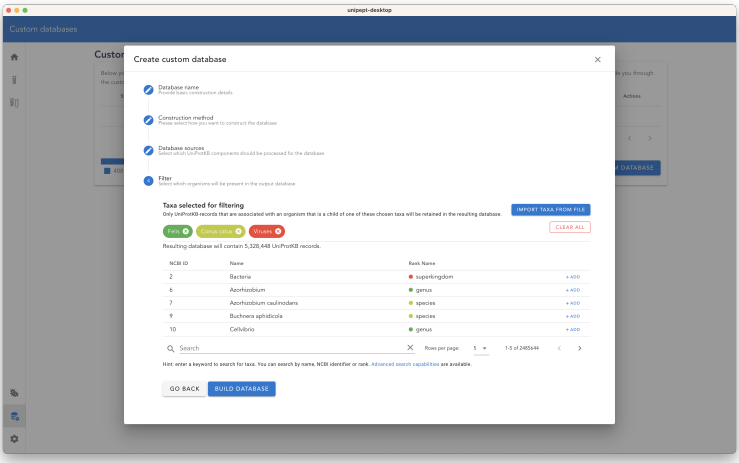

Figure 3.3: Database creation wizard in Unipept Desktop 2.0. This wizard guides the user through the process of building a targeted protein reference database. Researchers can select proteins in two different ways: by providing a set of taxon identifiers, or by providing a list of UniProtKB reference proteome IDs.

3.2.1 Construction of targeted protein reference databases

Protein reference databases that are required for the analysis of metaproteomic samples are typically very large. Depending on the number of proteins included, the size of these databases ranges from a few gigabytes to more than a terabyte. It is these huge size requirements that make it technically very hard to build targeted protein reference databases for multiple concurrent users on Unipept's servers.

Moving protein reference databases from Unipept's servers to a researcher's local machine opens up a world of new possibilities.

Firstly, researchers now have complete control over the database to be used for an analysis. They are no longer dependent on the update schedule dictated by Unipept, but can update their local reference database whenever they see fit. Previously, there was no way to roll back the database to the previous iteration of the UniProtKB resource after an update had been performed on Unipept's servers. This is important because researchers could no longer correctly compare metaproteomics analysis results for samples that were analyzed using a different database version.

Secondly, privacy sensitive data, which may be part of some metaproteomics experiments, is no longer sent over the internet to the Unipept servers for analysis. Some research institutes or applications do not allow sensitive data to be sent to remote services for analysis, but rather require it to be kept in-house to protect patient confidentiality and privacy.

Thirdly, in most cases, the runtime of analyses performed using a local database is improved because the data no longer needs to be transmitted over the Internet and the targeted protein reference databases are typically much smaller in size. The smaller the reference database, the faster the analyses can be performed.

Finally, the amount of false positive peptide matches that can occur is drastically reduced when compared to metaproteomics analyses against the complete UniProt database.

3.2.2 Implementation

3.2.2.1 Dependency management and portability

Unipept uses a custom format for storing protein reference databases, which includes a large amount of pre-computed data to speed up subsequent metaproteomics analyses. This database format is loaded into a relational database management system (RDBMS) such as MySQL. An RDBMS is a very specialized piece of software designed to query huge amounts of structured data as quickly as possible.

The Unipept Desktop application does not query this database directly, but instead relies on the Unipept API to do the hard work. The Unipept API provides a standardized set of HTTP REST endpoints that respond to queries from the Unipept Desktop (or Unipept Web) application with the desired information. This Unipept API in turn is a Ruby-on-Rails project that needs to be executed by a piece of software called a web server.

Both the installation and configuration of an RDBMS and a web server (to run the Unipept API) require a considerable amount of time, effort and technical skills, which is undesirable for users of the Unipept Desktop application. They need an application that is easy to install and that does not require a lot of user intervention to start and maintain.

To address these issues, we decided to encapsulate all required software dependencies in a Docker (Merkel) image. Docker is a free tool that allows to run predefined virtual containers (as defined by an image) on a variety of different operating systems. A Docker image contains a set of instructions to be executed by a virtual computing environment (controlled by Docker) and guarantees that these instructions will work deterministically on any supported system. By relying on Docker for the RDBMS and the web server, we have reduced the number of dependencies that are required by the Unipept Desktop to just one: Docker (which can be downloaded for free from its official website and supports all major operating systems).

Communication between the Unipept Desktop app and the software de-

pendencies that are managed by Docker is completely transparent to the user. We use a NodeJS package called Dockerode, which handles all communication between both parties.

3.2.2.2 Filtering UniProtKB by NCBI taxon identifiers

We described earlier how a targeted protein reference database can be constructed by selecting a list of NCBI taxon identifiers. In this case, Unipept selects only those proteins from the UniProtKB resource that are associated with one of these taxa (or children of these taxa) and includes them in the targeted reference database. In this section, we describe how this is implemented so that efficient filtering can be performed on all 227 million UniProtKB proteins.

The first step in the database construction process is downloading and processing all proteins from the UniProtKB database sources. For each source (SwissProt and TrEMBL), the application constructs a database index structure that can be easily queried and reused in the future. This index structure consists of several "chunks", each containing a set of different proteins. These chunks are compressed line-based `tsv`-files that contain all the necessary information (protein identifier, linked NCBI taxon identifiers, functional annotations, etc.). The protein information is sorted numerically by NCBI taxon identifier, and each chunk contains only those proteins that correspond to a known subset of the NCBI taxon identifier space. Figure 3.4 shows a schematic representation of how this index structure is constructed and the impact it has on the final targeted database construction process.

Downloading and constructing the reusable database index structure is a process that only needs to be performed once for each version of the UniProtKB resource. Subsequent targeted protein reference databases will reuse an existing index or will automatically rebuild it if the UniProtKB resource has been updated since the previous time the index was constructed.

To efficiently query the index structure, we first determine which chunks need to be queried. This is simply a matter of looking up which ranges

the various NCBI IDs provided fall into and only processing the chunks that correspond to those taxon ranges. All of these chunks are completely disjoint from each other and can be processed and filtered in parallel, maximizing the use of modern multi-core CPUs.

3.2.3 Case Study

To assess the strength of proteogenomics analyses in Unipept Desktop, we used Unipept Desktop 2.0 to perform the taxonomic annotation of a metaproteomic dataset obtained from 28 human fecal samples collected from celiac disease patients following a gluten-free diet and previously subjected to a 16S rRNA gene sequencing study (Bibbò *et al.*, 2020). Here, we re-analyzed and re-annotated the 16S rRNA gene sequencing data using a robust and up-to-date bioinformatics pipeline based on the amplicon sequence variant (ASV) approach and a newer database (Quast *et al.*, 2013) (as previously described (Palomba *et al.*, 2021)), to obtain accurate information about the set of bacterial taxa present in the environment under study. In parallel, the residues from the 28 fecal samples underwent protein extraction, filter-aided sample preparation and LC-MS/MS analysis, according to established procedures (Tanca *et al.*, 2022). Mass spectra were analyzed by Proteome Discoverer (version 2.4, Thermo Fisher Scientific) (Orsburn, 2021), using a publicly available collection of human gut metagenomes (Li *et al.*, 2014) as the sequence database, Sequest-HT as the search engine and Percolator for peptide validation (search parameters are detailed elsewhere). A total of 64 845 microbial peptides (of which 62 363 peptides remained after duplicate filtering) were identified (with 1% as FDR threshold) and used as input for Unipept Desktop annotation. We used the online NCBI Taxonomy Browser (Federhen, 2012) to convert between taxonomy IDs and taxa names where necessary.

The purpose of this case study was to demonstrate that the Unipept Desktop app is capable of analyzing a metaproteomic sample by matching the input peptides to only a specific subset of the proteins in the UniProtKB (The

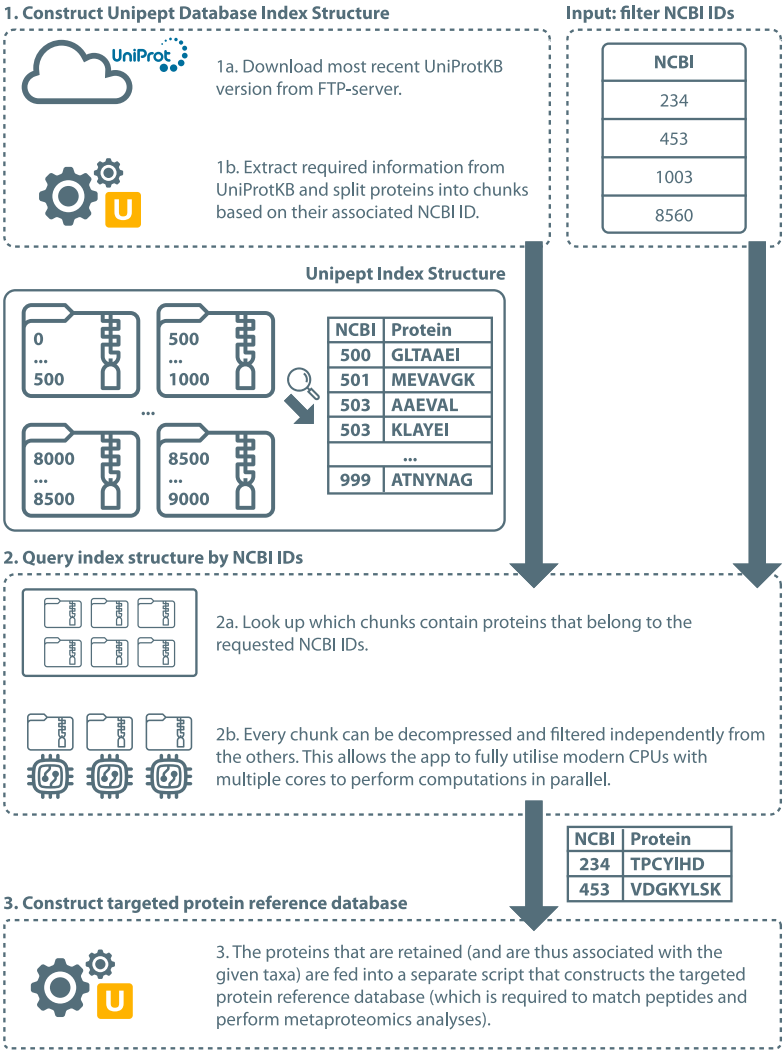

Figure 3.4: To efficiently filter the UniProtKB database, Unipept builds a custom index structure that can be easily filtered by taxon ID. This index structure only needs to be built the first time and will be reused for all subsequent database builds. The NCBI IDs, optionally provided by a user, are used to select which index chunks need to be queried. Each of these chunks is a compressed file containing protein information, which can then be processed in parallel by multiple CPU-cores.

UniProt Consortium, 2021) database. Secondly, we investigated the extent to which the taxonomic profile of a metaproteomic dataset differs when annotated against different protein reference databases.

As the 16S rRNA gene is only present in bacterial species, we restricted our analysis to bacteria. All metaproteomic analyses in this study are performed using Unipept Desktop v2.0.0-alpha.7 with the search settings "filter duplicates" and "equate I/L" enabled (the "advanced missed cleavage handling" setting was disabled).

The experiment started by processing all 227 million proteins in UniProtKB 2022.3 (both SwissProt and TrEMBL) and constructing an initial, unfiltered, protein reference database. During the analysis of our metaproteomic sample using Unipept Desktop and this general-purpose database, we were able to match 56 070 peptides (out of a total of 62 363 peptides, or 89.9%). Of these matches, 30 248 peptides (53.9%) were annotated with a taxon at the rank of "family" (or lower) and 13 659 peptides (24.4%) were annotated with a taxon at the rank of "species" (or lower).

We repeated this analysis using three other targeted protein reference databases, constructed by including only all taxa identified by the 16S rRNA gene sequencing analysis at the family, genus or species ranks, respectively. The higher a rank in the NCBI taxonomy, the less specific the constructed reference databases will be and the more proteins they will still contain. For each NCBI rank of interest, we counted the number of peptides annotated with a taxon of that rank (or lower) and compared these counts across the different protein reference databases. As can be seen in Figure 3.5, the total number of peptides matched decreases as the size of the protein reference databases decreases, but the number of taxon matches at a lower rank of the NCBI taxonomy increases for all three targeted protein reference databases.

Looking at the higher ranks, we see that the larger reference databases have an advantage and provide a valuable annotation for more peptides (53 243 (95.0%), 48 637 (86.7%) and 28 877 (51.5%) taxon matches at

the "Superkingdom" rank or lower for the "families", "genera" and "species" based databases, respectively). Note that, at this level, the targeted database constructed from a list of "families" already outperforms the unfiltered database (Figure 3.5).

It is important to note that there is a trade-off to be made between matches on a particular NCBI rank of interest and the input filter used to construct targeted protein reference databases. The more restrictive the input filter (assuming it is a good representation of the taxa in the environment under study), the fewer peptides will generally be matched, but the more likely it is that more detailed taxon matches will be found. This is easily explained. The more proteins there are in a reference database, the greater the chance that a purely random peptide match will occur by chance. These random matches are typically with proteins from organisms unrelated to the environment of interest, so the lowest common ancestor calculation ends up at a very high level in the NCBI taxonomy (Figure 3.1 and Figure 3.2).

Going down a few ranks in the NCBI taxonomy and counting the number of matches at the species level, we find 13 659 taxa (24.4%) when using the unfiltered database and 14 192 (25.3%), 14 593 (26.0%) and 13 960 matches (24.9%) for the "families", "genera" and "species" based databases, respectively. At this level, all three of the targeted reference databases outperform the unfiltered database while containing significantly fewer proteins. These significantly smaller protein reference databases require less storage space and fewer computing resources to work with, allowing the analyses to be performed on a simple laptop (rather than relying on the remote Unipept web servers). In most cases, the analysis is completed faster (especially when the missed cleavage handling option is enabled) and the end-user has full control over exactly which database is being used.

If we compare the targeted analysis with the analysis based on the unfiltered database, we see that 1 408 peptides that were previously annotated are now unmatched. As this is not an insignificant amount, it is important to investigate what the main differences between the two analyses at a

taxonomic level are.

If we look at the taxa that were matched using the full database, and not when using the targeted database, we see that most of them belong to the *Firmicutes phylum* (813 matches). Looking a little further, we see that the species *Evtepia gabavorous* (80 matches), *Odoribacter splanchnicus* (44 matches), and *Turicibacter sanguinis* (34 matches) are the most represented within the *Firmicutes*.

According to the NCBI taxonomy, *Evtepia gabavorous* belongs to the *Eubacteriales incertae sedis* "family", which is actually an unclassified taxon and is therefore not included in the 16S SILVA database. Secondly, *Odoribacter splanchnicus* belongs to the *Marinifilaceae* family according to SILVA, whereas it is assigned to the *Odoribacteraceae* family by the NCBI taxonomy (which is why this family is not present in the taxon list that was used to construct a protein reference database and is therefore not matched during the Unipept analysis). Finally, a similar explanation applies to *Turicibacter*, which belongs to *Erysipelotrichaceae* according to SILVA and to the *Turicibacteraceae* family according to NCBI. Therefore, this family is also missing from the filtered taxon list and its proteins are not included in the database construction.

Most of the other remaining *Firmicutes* assignments are mainly attributed to *Clostridiales bacterium* and *Firmicutes bacterium* (413 matches in total), two taxa for which the rank is unspecified in NCBI and which are therefore not taken into account when constructing the targeted database. A further 122 peptide matches using the full database were taxonomically annotated at the root level, which does not provide any valuable information other than the fact that the peptide was indeed matched.

After this detailed analysis, we can conclude that most of the mismatches when using a targeted reference database are due to taxonomic inconsistencies between the SILVA database (used for the 16S rRNA gene taxonomic annotation) and the NCBI taxonomy that is being used by Unipept, or even

to incomplete or provisional annotations within the NCBI taxonomy itself. This problem could be overcome by also using NCBI for the taxonomic classification of 16S data (instead of SILVA), but this choice is entirely up to the end-user, and is outside the scope of the analysis performed by Unipept Desktop.

All of these experiments have been conducted on a normal modern computer with a 6-core CPU (AMD Ryzen 3600X),16GiB of RAM, a SATA-6 SSD and a 100Mbps internet connection. On this machine, it took approximately 21 hours and 25 minutes to construct a custom database containing 46 million proteins or approximately 20 minutes for a targeted database containing 1 million proteins. All proteins from the UniProtKB resource are preprocessed the first time a targeted database is constructed and took approximately 5 hours and 30 minutes on this machine (this preprocessing step will automatically be performed when the UniProtKB resource updates). The final size of the database cache is 52GiB, and the 46 million and 1 million protein databases are respectively 255GiB and 5.76GiB in size.

3.2.4 Concluding Remarks

The newest iteration of the Unipept Desktop app builds upon the strength of the existing Unipept infrastructure to enable support for the analysis of proteogenomics samples. Leveraging taxonomic information of the environment under study (e.g. generated from a metagenomics experiment), it is possible to construct targeted protein reference databases that include only a (relevant) subset of proteins from the UniProtKB resource. These significantly smaller reference databases drastically improve the time and computational resources required to subsequently analyze metaproteomics samples, which ultimately makes it possible to perform these analyses on a local machine.

Since Unipept Desktop 2.0 makes it possible to perform metaproteomics analysis on a local machine, a range of new possibilities opens up. Privacy-

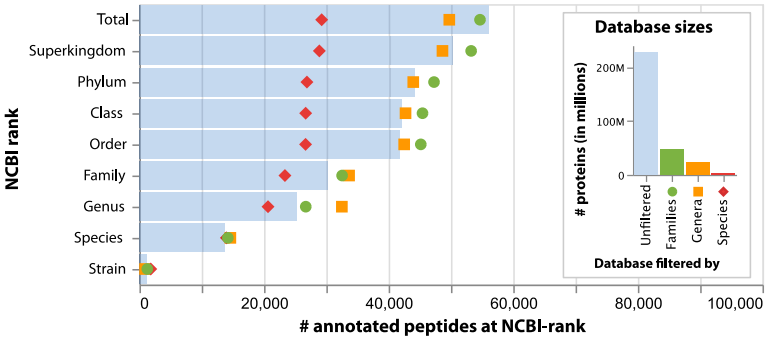

Figure 3.5: Four analyses have been performed on the same metaproteomic sample, but with a different underlying protein reference database. In the main visualization (left), we compare the number of peptides with a taxon match at a specific rank (or lower) in the NCBI taxonomy. In most cases the analyses performed by using a reference database constructed from families or genera performs as well as or better than the full UniProtKB database. However, these databases contain only a fraction of the proteins (right) and therefore require much less computational resources. At the species level, all protein reference databases behave almost identically. At higher ranks, we can see that the reference database constructed from a set of species is probably too restricted.

sensitive data no longer needs to be transmitted over the internet and users now control which reference database is used. We have shown that using targeted protein reference databases can even lead to a metaproteomics analysis with a higher taxonomic resolution (assuming that the selected taxa suits the environment under study). No scientific software package is ever completed and we can still think about future improvements that could be beneficial for the Unipept Desktop application. First of all, at this point it is not yet possible to construct targeted protein reference databases from UniProtKB versions other than the current one. This is a consequence of the fact that previous versions of the UniProtKB database are provided in a different file format for which a new parser needs to be implemented.

Secondly, since all targeted protein reference databases are always constructed by filtering the UniProtKB resource, only proteins that are included in UniProtKB can be matched using Unipept. This can be problematic for some research disciplines (such as protein research of ocean water) that are investigating proteins of organisms that are not well represented in UniProtKB. This problem could be overcome by allowing Unipept to construct protein reference databases from external sources (e.g., represented by FASTA or PEFF files). These additions are considered for future versions of the Unipept Desktop app.

3.2.5 Acknowledgements

This work has benefited from collaborations facilitated by the Metaproteomics Initiative (https://metaproteomics.org/) whose goals are to promote, improve, and standardize metaproteomics (Van Den Bossche, Kunath, *et al.*, 2021). This work has furthermore been supported by the Research Foundation - Flanders (FWO) [1164420N to P.V., 12I5220N to B.M.] and by the University of Sassari [Fondo di Ateneo per la Ricerca 2020 to A.T.]. We thank the Flemish Supercomputer Center (VSC) funded by the Research Foundation - Flanders (FWO) and the Flemish Government for providing the infrastructure to build the Unipept database and

to run the experiments from this manuscript. Part of this work was also supported by the Research Foundation - Flanders (FWO) for ELIXIR Belgium [I002819N].

Part II

The Unipept ecosystem

Unipept is not a single tool, but rather a collection of tools that support the analysis of metaproteomics and metagenomics datasets. In the previous part of this PhD thesis, I discussed the novel Unipept Desktop application and all of the improvements that have been made in that area. I have, however, also been part of a lot of other projects that are either a direct part of the Unipept ecosystem or very closely related to it.

In Chapter 4, I discuss the changes made to the Unipept CLI and API. As described in (Gurdeep Singh *et al.*, 2019), a functional analysis pipeline for metaproteomics datasets has been added to the Unipept web application in 2018. Initially, we supported the annotation and analysis of Enzyme Commission numbers (Webb, 1992) and Gene Ontology terms (Ashburner *et al.*, 2000). Together with Philippe Van Thienen (a master student that I guided in 2019), we expanded the functional analysis pipeline with support for InterPro annotations (Hunter *et al.*, 2009) and integrated our novel functional analysis with the Unipept CLI and API.

All of the visualizations that are incorporated in the Unipept web application are, in the first place, designed to visualize the results produced by Unipept. The visualizations are, however, powerful enough to be used for other types of data as well. In Chapter 5, I discuss how all of the visualizations were extracted into a separate JavaScript visualization library that can be used by third-party applications. A new visualization, the heatmap (including support for the visualization of a dendrogram), is also featured in this chapter.

MegaGO is a tool for the comparison of multiple sets of GO-terms and is discussed in Chapter 6. This tool is not a direct member of the Unipept ecosystem, but is nonetheless closely related to Unipept. For each metaproteomics analysis, Unipept reports a set of GO-terms that have been identified and which can afterwards be fed into the MegaGO application for comparison with other analysis results.

Finally, I also consider three side-projects that I contributed to in Chap-

ter 7. The first project, Pout2Prot, talks about the application of protein grouping on a specific type of output file (with the `.pout` extension, as generated by the Percolator software). The second project has not received a definitive name yet and consists of highlighting the metabolic pathways that are active in a complex microbial ecosystem. Lastly, I present a very special `HashMap` implementation for JavaScript that can be used by multiple processes simultaneously.

Chapter 4

Unipept CLI 2.0: adding support for visualisations and functional annotations

Release 4.0 of the Unipept Web application is the first release of Unipept that provides a functional analysis pipeline for metaproteomics datasets. This initial release provided support for Enzyme Commission numbers and Gene Ontology numbers. Together with Philippe Van Thienen (a master student that I guided during the first year of my PhD), we further expanded Unipept's functional analysis pipeline with support for InterPro annotations. In order to expose this new functionality to power users and third-party applications, we have also implemented support for this novel functional analysis pipeline in the Unipept CLI and API.

This chapter contains a verbatim copy of the technical note by (Verschaffelt et al.*, 2020) as submitted to Bioinformatics.*

Abstract — Unipept (Mesuere *et al.*, 2012) is a collection of tools developed for fast metaproteomics data analysis. The Unipept ecosystem consists of a web application, an application programming interface (API) as a web service (Mesuere *et al.*, 2016) and a command-line interface (CLI) (Mesuere *et al.*, 2018). The key strengths of Unipept are its speed, its ease-of-use and the extensive use of interactive data visualization in the analysis results. The Unipept database is derived from the UniProt (UniProt, 2019) KB and consists of tryptic peptides linked with taxonomic and functional annotations. Unipept initially launched with support for taxonomic analysis of metaproteomics data in 2012. Version 4.0 (Gurdeep Singh *et al.*, 2019) of the Unipept web application was launched in November 2018 and extended the web interface with support for functional annotations such as Gene Ontology (GO) terms (Ashburner *et al.*, 2000), Enzyme Commission (EC) numbers (Webb, 1992) and InterPro entries (Hunter *et al.*, 2009).

4.1 Introduction

The GO terms are organized into three different domains: 'cellular components', 'molecular functions' and 'biological processes'. Every GO-term is associated with exactly one domain and consists of a name, an identifier and an exact definition. The EC numbers can be used to classify enzymes, based on the chemical reactions that they catalyze. Every EC number consists of four numbers, separated by a dot, yielding a hierarchical classification system with progressively finer enzyme classifications. InterPro is a database that consists of predictive models collected from external databases that can be classified into five different categories. More information about functional annotation in metaproteomics can be found in the study by (Schiebenhoefer *et al.*, 2019).

For each input peptide, Unipept finds all functional annotations associated

with all of the UniProt entries in which the peptide occurs. All found functions are listed in order of decreasing number of peptides associated with this function.

In this article, we present several new additions to the Unipept API and CLI which allow third-party applications [such as Galaxy-P (Jagtap *et al.*, 2015)] to integrate the new functional analysis capabilities provided by Unipept.

4.2 Materials and methods

The Unipept API is a high-performance web service that responds in a textual format (JSON) to HTTP-requests from other applications or tools and allows to integrate the services provided by Unipept into other workflows. Unipept's CLI is a Ruby-based application and high-level entry point which allows users to actively query Unipept's database. Compared to the API, users do not need to compile API-requests manually but can rely on the CLI to automatically do so in a parallelized way. In addition, it supports multiple input and output formats such as FASTA and CSV.

The Unipept database and web application were recently expanded to include GO terms, EC numbers and InterPro entries. These new annotations are now also available from newly developed API-endpoints and CLI-functions, providing structured access to this functional information.

Most of the newly developed endpoints support batch retrieval of information for a list of peptides. In this case, the API returns a list of objects where each object in the response corresponds with information associated with one of the input peptides. Every API-endpoint is accompanied by an identically named CLI-function, which provides the user with the ability to import data from or export data to various specifically formatted files. In addition, version 2 of the Unipept CLI introduces the ability to produce interactive visualizations directly from the command line.

Among other information, the Unipept tryptic peptide analysis lists functional annotations associated with a given tryptic peptide. These data are aggregated because a peptide can occur in multiple proteins that each can have multiple functional annotations. For each annotation, we also return the amount of underlying proteins that match with this specific annotation.

Some applications require all known information for a list of tryptic peptides. The 'pept2funct' function is a combination of the preceding three endpoints and returns all functional annotations associated with the given tryptic peptide. 'peptinfo' on the other hand, returns all the available information for one or more tryptic peptides. All functional annotations for this peptide are part of the response, as well as the lowest common ancestor for this peptide. Both functions also support splitting the GO terms and InterPro entries over the respective domains, and naming information can optionally be retrieved.

The 'taxa2tree' function constructs a tree from a list of NCBI taxon ids. This tree is an aggregation of the lineages that correspond with the given taxa and can be exported as three distinct output formats: JSON, HTML and as a URL. The HTML and URL representation of a taxonomic tree both provide three interactive data visualizations, albeit with different possibilities. A generated HTML-string first needs to be stored in a file before it can be rendered by a browser and cannot be easily shared with other people but is easily editable. A URL on the other hand is simply a shareable link to an online service that hosts all interactive visualizations.

4.3 Conclusion

Version 2.0 of the Unipept API and CLI is a significant update that provides fast and easy access to the powerful analysis pipeline of Unipept. In addition to the existing taxonomic analysis, it now features multiple functional annotations which will enable users to gain new insights into complex ecosystems. These new features can easily be integrated into third-party

tools such as the MetaProteome Analyzer (Muth *et al.*, 2018). Galaxy-P, a highly used workflow integration system, is already successfully making use of the novel analysis functions that are introduced with this new release.

4.4 Funding

This work was supported by the Research Foundation—Flanders (FWO) [1164420N to P.V.; 12I5220N to B.M.; 1S90918N to T.V.D.B.; G042518N to L.M.; 12S9418N to C.D.T.].

Chapter 5

Unipept Visualizations: an interactive visualization library for biological data

Unipept's visualizations have been designed to be flexible and useable for data that is not generated by Unipept. In order to promote reuse of these visualizations by other parties, we extracted the visualizations from Unipept and exposed them as separate modules in a JavaScript package.[1] *Each visualization exposes an interface that data needs to adhere to in order for them to be visualized by the components provided in this package. A new visualization, the heatmap (with support for dendrograms), also makes it first appearance here.*

Together with the extraction of the visualization components, we have also updated the underlying code from JavaScript to TypeScript. I've had a lot of help from Dr. James Collier (VIB) and Dr. Alexander Botzki (VIB) for this. The application note that's included in this chapter provides a technical description about these visualizations, and how they are designed on the inside.

[1] https://www.npmjs.com/package/unipept-visualizations

This chapter contains a verbatim copy of the application note by (Verschaffelt et al.*, 2022) as submitted to Bioinformatics.*

Abstract — The Unipept Visualizations library is a JavaScript package to generate interactive visualizations of both hierarchical and non-hierarchical quantitative data. It provides four different visualizations: a sunburst, a treemap, a treeview and a heatmap. Every visualization is fully configurable, supports TypeScript and uses the excellent D3.js library. The Unipept Visualizations library is available for download on NPM: https://npmjs.com/unipept-visualizations. All source code is freely available from GitHub under the MIT license: https://github.com/unipept/unipept-visualizations.

5.1 Introduction

Unipept is an ecosystem of software tools for the analysis of metaproteomics datasets that consists of a web application (Gurdeep Singh *et al.*, 2019), a desktop application (Verschaffelt, Van Den Bossche, Martens, *et al.*, 2021), a command line interface (Verschaffelt *et al.*, 2020) and an application programming interface. It provides taxonomic and functional analysis pipelines for metaproteomics data and highly interactive data visualizations that help interpret the outcome of these analyses.

We developed custom visualizations used for Unipept from scratch because existing libraries, such as Krona (Ondov *et al.*, 2011), were lacking essential features or were hard to integrate. They were designed as generic tools to visualize hierarchical quantitative data and can therefore also be used to visualize data from nonproteomics origins. To facilitate reuse of these broadly usable components, we have isolated the visualizations from the main Unipept project and made them available as a standalone package that can easily be reused by other software tools. We released this package under the permissive MIT open-source license, so researchers from other disciplines are free to reuse these visualizations and connect them to their own data sources. Currently, our visualizations are already incorporated in

TRAPID 2.0, a web application for the analysis of transcriptomes ((Bucchini *et al.*, 2021) and UMGAP, the Unipept MetaGenomics Pipeline (Van der Jeugt *et al.*, 2022).

5.2 Visualizations

We currently provide four highly interactive data visualizations that are all designed for a specific purpose: a sunburst, a treeview, a treemap and a heatmap. The sunburst (Figure 5.1a), treeview (Figure 5.1d) and treemap (Figure 5.1b) can be used to visualize quantitative hierarchical data and are designed to depict the parent–child relationship of a hierarchy of nodes as clearly as possible, while still incorporating the strength of the relationship between, or the counts associated with, connected nodes. The heatmap (Figure 5.1c), conversely, is not suitable to visualize hierarchical information but displays a magnitude in two dimensions, including optional clustering and dendrogram rendering.

5.2.1 Quantitative hierarchical data visualizations

Hierarchical data occurs throughout a variety of bioinformatics disciplines. In the metaproteomics research area alone, many examples of hierarchical data exist, such as the hierarchical structure of the NCBI taxonomy (Schoch *et al.*, 2020), the hierarchy imposed by the enzyme commission numbers and the gene ontology terms (The Gene Ontology Consortium, 2019). In most cases, quantitative data are available for multiple nodes at many levels in the hierarchy. For example, Unipept assigns peptide counts to taxa that are scattered around the NCBI taxonomy, including identifications that are highly specific (near leaves of the tree) or lack deep taxonomic resolution (near the root of the tree). Being able to interactively zoom in and out on the hierarchical data enables exploratory analysis.

The three visualizations for hierarchical data provided by our package take input data in the same hierarchical format, making it trivial to switch be-

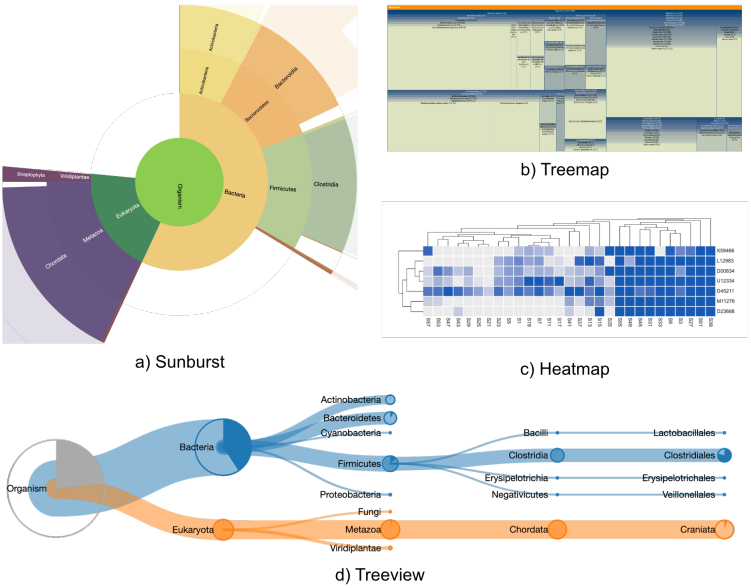

Figure 5.1: Overview of the visualizations currently provided by the Unipept Visualizations library. All examples were generated with default configuration settings, except for the heatmap for which the setting 'dendrogramEnabled' was set to 'true'.

tween the different types of visualization once the input data are formatted correctly.

5.2.2 Quantitative non-hierarchical data visualizations

A heatmap (Figure 5.1c) is a well-known visualization that consists of a two-dimensional grid of cells in which each cell is assigned a specific color from a scale corresponding to its magnitude. The heatmap implementation in our package provides this functionality in an extensively customizable form. Users can reorganize elements, change the color scheme and update label information, among other operations. All values are also automatically normalized to a $[0, 1]$-interval.

As neighboring rows and columns in the input data can have very distinct values, and as this can interfere with reasoning about the heatmap, it is important to group similar values. Our implementation achieves this through hierarchical clustering based on the UPGMA algorithm (Sokal and Michener, 1958). The produced grouping of rows and columns is further clarified by an optional dendrogram that can be plotted alongside each axis of the heatmap.

However, after clustering, it can still occur that two consecutive leaves in a dendrogram are quite dissimilar due to the $2^n - 1$ possible linear orderings that can be derived from a dendrogram (a dendrogram contains $n - 1$ flipping points for which both children can be switched). This can be addressed by reordering the leaves of the tree, as the orientation of the children of all n nodes in a dendrogram can be flipped without affecting the integrity of the dendrogram itself. Our heatmap implementation uses the Modular Leaf Ordering technique (Sakai *et al.*, 2014) to reorder all leaves of the dendrogram such that the distance between consecutive leaves is minimized. This technique is a heuristic that performs very well in comparison to the more resource-intensive Optimal Leaf Ordering (Bar-Joseph *et al.*, 2001) or Gruvaeus–Wainer algorithms (Gruvaeus and Wainer, 1972).

5.3 Implementation

The visualization package has been developed with D3 (Bostock *et al.*, 2011) and TypeScript (Bierman *et al.*, 2014) and every visualization is displayed in the web browser with one of two technologies: SVG or HTML5 canvas. SVG's are easy-to-use and are scalable by nature but often lack necessary performance for complex interactive visualizations. HTML5 canvas, in contrast, provides much better performance using a rasterized image.

Every visualization is presented as a single JavaScript class and provides a full set of configuration options to extend and configure the visualization. New versions of the package will automatically be published on NPM (https://npmjs.org) and GitHub (https://github.com/unipept/unipept-visualizations), so that any project depending on the package can always use the latest version.

We also provide an extensive set of documentation resources that ease the adoption process of our package, as well as a collection of live notebooks (see https://observablehq.com/collection/@unipept/unipept-visualizations). These notebooks provide interactive and editable examples that demonstrate the full potential and guide users through the different configuration options. The code and resources that make up the live notebooks can be modified online and provide a very convenient way to try out the package.

5.4 Funding

This work has been supported by the Research Foundation—Flanders (FWO) [1164420N to P.V.; 12I5220N to B.M.; G042518N to L.M.].

Chapter 6

MegaGO: a fast yet powerful approach to assess functional Gene Ontology similarity across meta-omics data sets

The MegaGO project was born at the 2020 EuBIC Developer's meeting in Nyborg, Denmark. This developer's meeting was the first conference that I joined as a PhD-student and consisted of little hackathon projects that were planned and executed in groups of about 6 to 8 people. Together with Prof. Dr. Ir. Bart Mesuere, Dr. Tim Van Den Bossche and Dr. Henning Schiebenhoefer, I hosted a hackathon project over there. During the months after the conference, we have further expanded this project with a command line tool and a web application.[1]

The manuscript discussed in this chapter has mainly been written by Dr. Tim Van Den Bossche, Dr. Henning Schiebenhoefer and myself. I was furthermore the lead developer of the web application and contributed a big portion of the back-end code that powers the analyses of our web app.

[1] https://megago.ugent.be/

This chapter contains a verbatim copy of the manuscript by (Verschaffelt, Van Den Bossche, Gabriel, et al.*, 2021) as submitted to Journal of Proteome Research.*

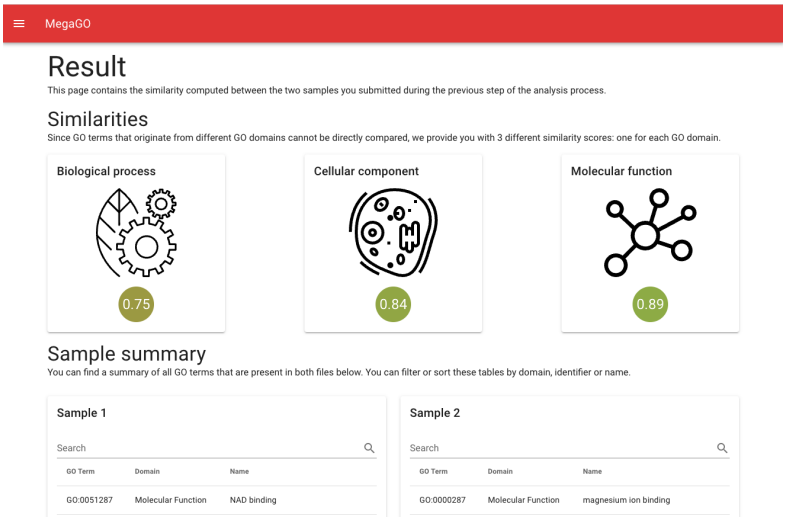

Figure 6.1: Screenshot of the MegaGO Web application. This application is accessible at https://megago.ugent.be.

Abstract — The study of microbiomes has gained in importance over the past few years and has led to the emergence of the fields of metagenomics, metatranscriptomics, and metaproteomics. While initially focused on the study of biodiversity within these communities, the emphasis has increasingly shifted to the study of (changes in) the complete set of functions available in these communities. A key tool to study this functional complement of a microbiome is Gene Ontology (GO) term analysis. However, comparing large sets of GO terms is not an easy task due to the deeply branched nature of GO, which limits the utility of exact term matching. To solve this problem, we here present MegaGO, a user-friendly tool that relies on semantic similarity between GO terms to compute the functional similarity between multiple data sets. MegaGO is high performing: Each set can

contain thousands of GO terms, and results are calculated in a matter of seconds. MegaGO is available as a web application at https://megago.ugent.be and is installable via pip as a standalone command line tool and reusable software library. All code is open source under the MIT license and is available at https://github.com/MEGA-GO/.

6.1 Introduction

Microorganisms often live together in a microbial community or microbiome where they create complex functional networks. These microbiomes are therefore commonly studied to reveal both their taxonomic composition as well as their functional repertoire. This is typically achieved by analyzing their gene content using shotgun metagenomics. Whereas this approach allows a detailed investigation of the genomes that are present in such multiorganism samples, it reveals only their functional potential rather than their currently active functions (Jansson and Baker, 2016). To uncover these active functions within a given sample, the characterization of the protein content is often essential (Lohmann *et al.*, 2020).

The growing focus on functional information as a complement to taxonomic information (Louca *et al.*, 2016) is derived from the observation that two taxonomically similar microbial communities could have vastly different functional capacities, whereas taxonomically quite distinct communities could have remarkably similar functions. Whereas the investigation of the active functions is thus increasingly seen as vital to a complete understanding of a microbiome, the identification and comparison of these detected functions remains one of the biggest challenges in the field (Schiebenhoefer *et al.*, 2019).

Several omics tools exist to describe functions in microbial samples, although these tools link functionality to different biological entities such as genes, transcripts, proteins, and peptides (Muth *et al.*, 2015, 2018; Van Den Bossche *et al.*, 2020; Verschaffelt *et al.*, 2020; Gurdeep Singh *et al.*,

2019; Riffle *et al.*, 2018; Schneider *et al.*, 2011; Schiebenhoefer *et al.*, 2020; Huerta-Cepas *et al.*, 2019; Huson *et al.*, 2007). However, most tools are capable of directly or indirectly reporting functional annotations as a set of Gene Ontology (The Gene Ontology Consortium, 2019) (GO) terms, regardless of the biological entities they are assigned to. In October 2020, there were 44264 of these terms in the complete GO tree. GO terms are organized into three independent domains: molecular function, biological process, and cellular component (Ashburner *et al.*, 2000). In each domain, terms are linked into a directed acyclic graph, an excerpt of which is shown in Figure 6.2. Figure 6.2 shows the Gene Ontology graph for all parent terms up to the root for the GO-term "translation" (GO:0006412). In this case, the root GO term "biological process" (GO:0008150) has multiple children, while the most specific term "translation", in contrast, has multiple parents. When comparing the two terms GO:0044267 and GO:0034645 (portrayed in light red), we find two different lowest common ancestors: GO:0044249 and GO:1901576 (dark red). Only one of these, however, can be the most informative common ancestor (MICA), that is, the common ancestor with the highest information content for the terms in light red. Because an IC of 1.52 is larger than 1.48, the GO:0044249 is the MICA. The terms GO:0043604 and GO:0006518 (in light blue) are more similar than the two terms we previously described and have only one lowest common ancestor, which is also automatically the MICA for these terms: GO:0043603 (in dark blue).

Whereas this highly branched graph structure of GO allows flexible annotation at various levels of detail, it also creates problems when the results from one data set are compared to those of another data set. Indeed, even though two terms may be closely linked in the GO tree and are therefore highly similar (e.g., as parent and child terms or as sibling terms), the typically employed exact term matching will treat these terms as wholly unrelated, as the actal GO terms (and their accession numbers) are not identical. This problem is illustrated in a study by (Sajulga *et al.*, 2020), where a multisample data set was analyzed using several metaproteomics

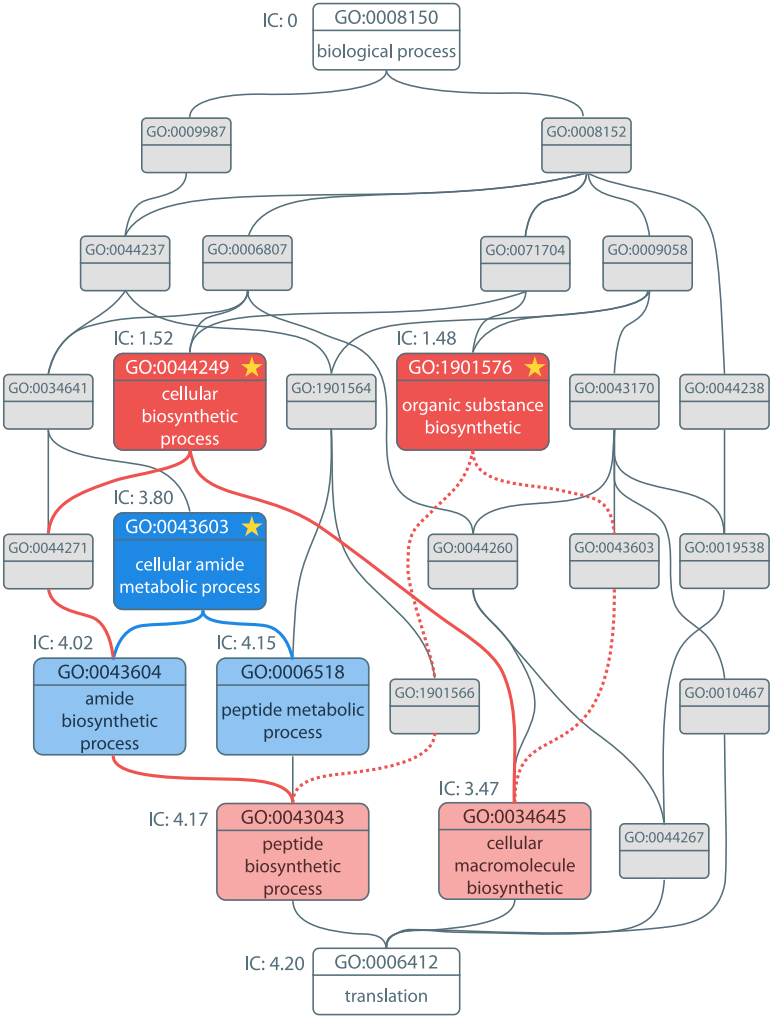

Figure 6.2: Excerpt of the biological process domain of the Gene Ontology showing all parent terms up to the root for "translation" (GO:0006412). IC, information content; *, most informative common ancestor.

tools. The resulting GO terms were then compared using exact matching. The overlap between the result sets was quantified using the Jaccard index and was found to be low. As previously explained, this low similarity is likely the result of the limitations of the exact term matching approach.

There is thus a clear need for a more sophisticated GO term comparison that takes into account the existing relationships in the full GO tree. However, most existing tools that provide such comparison are based on enrichment analyses (Huang *et al.*, 2009; Waardenberg *et al.*, 2015; Fruzangohar *et al.*, 2013). In such analyses, a list of genes is mapped to GO terms, which are then analyzed for ienriched biological phenomena. As a result, to the best of our knowledge, no tools allow the direct comparison of large functional data sets against each other, nor are these able to provide metrics to determine how functionally similar data sets are.

We therefore present MegaGO, a tool for comparing the functional similarity between large sets of GO terms. MegaGO calculates a pairwise similarity score between multiple sets of GO terms for each of the three GO domains and can do so in seconds, even on platforms with limited computational capabilities.

6.2 Implementation

To measure the similarity between sets of GO terms, we first need to measure the similarity of two individual terms. We compare two terms using the Lin semantic similarity metric, which can take on a value between 0 and 1 (Supplementary Formula 1a). The Lin semantic similarity is based on the ratio of the information content of the most informative common ancestor (MICA) to the average of the terms' individual information content.

The information content (Supplementary Formula 1b) is computed by estimating the terms' probability of occurrence (Supplementary Formula 1c), including that of all of their children. Term frequencies are estimated based on the manually curated SwissProt database (The UniProt Consortium,

2019). As a result, a high-level GO term such as "biological process" (through its many direct or indirect child terms) will be present in all data sets and thus carries little information. A more specific term such as "translation" (or any of its potential child terms) will occur less frequently and thus will be more informative (Figure 6.2). To finally calculate the similarity of two terms, we compare their information content with that of their shared ancestor that has the highest information content, the MICA. If the information content of the MICA is similar to the terms' individual information content, then the terms are deemed to be similar. The dissimilar terms "peptide biosynthetic process" and "cellular macromolecule biosynthetic" are situated further from their MICA "cellular biosynthetic process" than the similar terms "amide biosynthetic process" and "peptide metabolic process" with their respective MICA "cellular amide metabolic process" (Figure 6.2).

MegaGO, however, can compare not only two terms but also sets of GO terms. More specifically, two sets of GO terms can be compared via the web application, but an unlimited number of sets can be compared via the command line tool. Note that in these sets, duplicate GO terms will be removed so that each GO term will be equally important, regardless of how often it is provided by the user. To compare the sets of GO terms, pairwise term similarities are aggregated using the Best Matching Average (BMA, Supplementary Formula 2) (Schlicker *et al.*, 2006). For each GO term in the first input data set, the BMA finds the GO term with the highest Lin semantic similarity in the second data set and averages the values of these best matches. Moreover, MegaGO calculates the similarity for each of the three domains of the gene ontology (molecular function, biological process, and cellular component), as GO terms from distinct domains do not share parent terms. The general overview of MegaGO is shown in Figure 6.3.

MegaGO is implemented in Python, is installable as a Python package from PyPi, and can easily be invoked from the command line. The GOATOOLS (Klopfenstein *et al.*, 2018) library is used to read and process

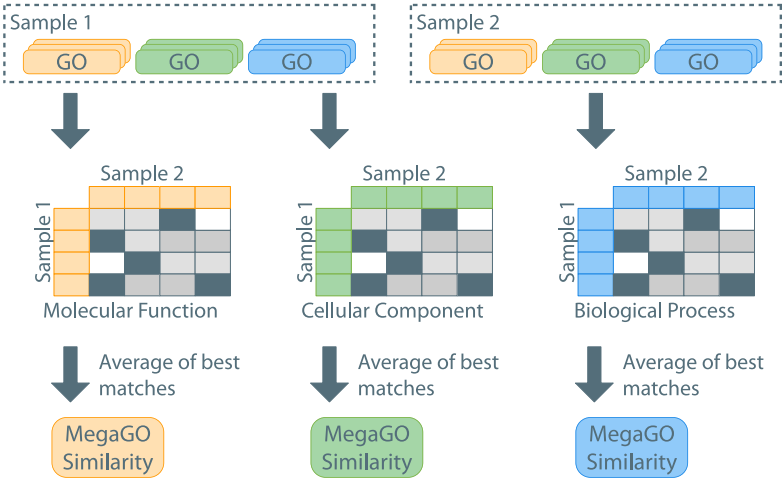

Figure 6.3: Overview of MegaGO workflow. The Gene Ontology (GO) terms of each sample set are separated into three GO domains: molecular function, cellular component, and biological process. Each term of each sample set is compared to every term in the other set that is from the same domain. The match with highest similarity for each term is then selected, and the average across all of these best matches is calculated.

the Gene Ontology and to compute the most informative common ancestor of two GO terms, which are both required to compute the information content value (Supplementary Formula 1, p(go)). GO term counts are recomputed with every update of SwissProt, and a new release is automatically published bimonthly to PyPi, which includes the new data set. Automated testing via GitHub Actions is in place to ensure correctness and reproducibility of the code. In addition, we also developed a user-friendly and easily accessible web application that is available on https://megago.ugent.be. The backend of the web application is developed with the Flask web framework for Python, and the frontend uses Vue. Our web application has been tested on Chromium-based browsers (Chrome, Edge, and Opera) as well as Mozilla Firefox and Safari. The MegaGO application is also available as a Docker-container on Docker Hub (https://hub.docker.com/repository/docker/pverscha/mega-go) and can be started with a single click and without additional configuration requirements. Our Docker container is automatically updated at every change to the underlying MegaGO code. All code is freely available under the permissive open source MIT license on https://github.com/MEGA-GO/. Documentation for our Python script can be found on our Web site: https://megago.ugent.be/help/cli. A guide on how to use the web application is also available: https://megago.ugent.be/help/web.

MegaGO is cross-platform and runs on Windows, macOS, and Linux systems. The system requirements are at least 4 GiB of memory and support for either Python 3.6 (or above) or the Docker runtime.

6.3 Validation

To validate MegaGO, we reprocessed the functional data from (Easterly *et al.*, 2019). This data set consists of 12 paired oral microbiome samples that were cultivated in bioreactors. Each sample was treated with and without sucrose pulsing, hereafter named ws and ns samples, respectively. Each sample contains mass-spectrometry-based proteomics measurements, and

all samples were annotated with 1718 GO terms on average. We calculated the pairwise similarity for each of the 300 sample combinations, which took less than 1 min for a single sample pair on the web version of MegaGO. This resulted in a MegaGO similarity score for each of the three GO domains for each sample combination. These similarities were then hierarchically clustered and visualized in a heatmap. All data and intermediate steps of our data analysis are available at https://github.com/MEGA-GO/manuscript-data-analysis/ and can be reproduced with the command line tool using the –heatmap option.

In the heatmap (Figure 6.4), we can observe that the two sample groups cluster together, except for 730ns and 733ns that are clustered in the ws sample group. These two samples were also identified as outliers in (Easterly *et al.*, 2019) and 733ns was originally also identified as both a taxonomic and functional outlier in (Rudney *et al.*, 2015). Similar results can be observed for the GO domain "molecular function" (Supplementary Figure 1). The MegaGO similarity-based clustering of "cellular component" GO terms (Supplementary Figure 2) has two additional samples clustered outside of their treatment group: 852ws in the ns cluster and 861ns in the ws group. Again, these patterns can also be found in previous analyses: 852ws is placed in direct proximity of the ns samples in the principal component analysis (PCA) of the HOMINGS analysis by Rudney et al., and 861ns is closest to 730ns and 733ns in PCA of Rudney et al.'s taxonomic analysis. Interestingly, subjects 730 and 852 were the only ones without active carious lesions, which could cause their divergence in the similarity analyses.

Results produced by MegaGO are thus in close agreement with prior analyses of the same data, showing that MegaGO offers a valid and very fast approach for comparing the functional composition of samples.

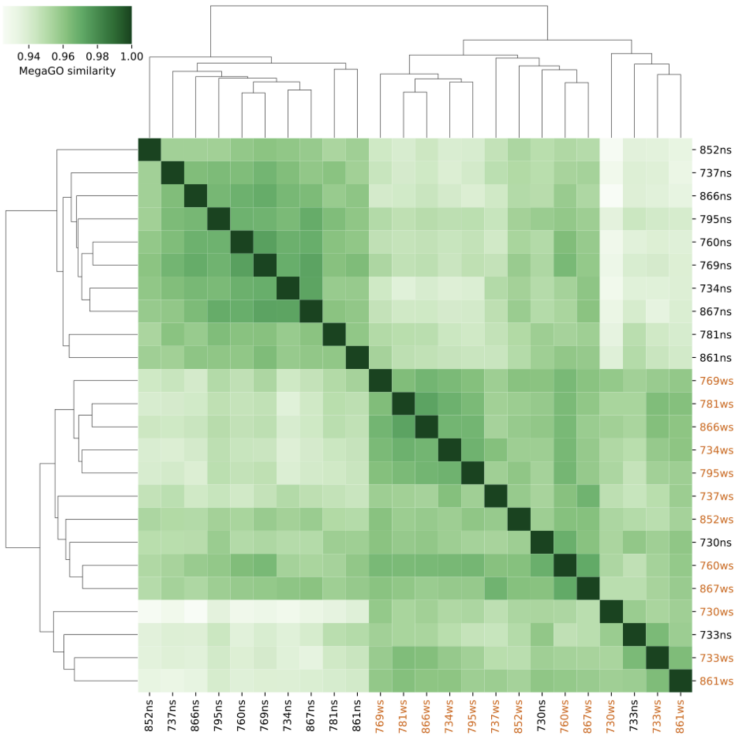

Figure 6.4: Hierarchically clustered heatmap comparing MegaGO similarities for the GO domain "biological process" for each of the samples from (Easterly *et al.*, 2019). Samples that are treated with sucrose pulsing are labeled as "ws" and are displayed in orange.

6.4 Conclusions

MegaGO enables the comparison of large sets of GO terms, allowing users to efficiently evaluate multiomics data sets containing thousands of terms. MegaGO separately calculates a similarity for each of the three GO domains (biological process, molecular function, and cellular component). In the current version of MegaGO, quantitative data are not taken into account, thus giving each GO term identical importance in the data set.

MegaGO is compatible with any upstream tool that can provide GO term lists for a data set. Moreover, MegaGO allows the comparison of functional annotations derived from DNA-, RNA-, and protein-based methods as well as combinations thereof.

6.5 Acknowledgements

We acknowledge the European Bioinformatics Community for Mass Spectrometry (EuBIC-MS). This project found its origin at the EuBIC Developers' 2020 meeting (Ashwood *et al.*, 2020) in Nyborg, Denmark. We thank Thilo Muth (Bundesanstalt für Materialforschung und -prüfung, Berlin, Germany) and Stephan Fuchs (Robert Koch Institute, Berlin, Germany) for their support. P.V., T.V.D.B., L.M., and B.M. acknowledge the Research Foundation - Flanders (FWO) (grants 1164420N, 1S90918N, G042518N, and 12I5220N). L.M. also acknowledges support from the European Union's Horizon 2020 Programme under grant agreement 823839 (H2020-INFRAIA-2018-1). H.S. and B.Y.R. acknowledge support by Deutsche Forschungsgemeinschaft (DFG; grant numbers RE3474/5-1 and RE3474/2-2) and the BMBF-funded de.NBI Cloud within the German Network for Bioinformatics Infrastructure (de.NBI; 031A537B, 031A533A, 031A538A, 031A533B, 031A535A, 031A537C, 031A534A, and 031A532B).

Chapter 7

Other projects

During the course of my career as a PhD student, I have also been working on a lot of different research projects for which I was either not the main contributor (but still provided a significant addition), or which are smaller side-projects that I wanted to experiment with. I have selected three of these projects and included them as sections in this chapter.

Pout2Prot is a project that I did together with Dr. Tim Van Den Bossche and Dr. Kay Schallert. Tim and Kay dedicated a lot of time on writing the manuscript, while I mainly focused on the implementation of a scalable web application that allows researchers to perform protein grouping on Percolator files.

The second side-project that I included in this chapter does not have a final name yet, but consists of a new approach at improving the interpretation of metabolic pathways and how specific proteins are involved in these metabolic processes. This project started at the EuBIC 2023 hackathon in Switzerland where I've worked on the tool with a whole team. Tibo Vande Moortele is the lead developer of the pathway visualization web application and together with him, we are currently in the process of trying to incorporate some of these features into the Unipept ecosystem.

Lastly, I am presenting a special type of `HashMap`*, implemented in JavaScript, that is tailored at web applications that are wanting to exploit the parallelism provided by modern multi-core CPUs. This project first started as an experiment and is currently completely managed by myself.*

7.1 Pout2Prot: An efficient tool to create protein (sub)groups from Percolator output files

This section contains a verbatim copy of the manuscript by (Schallert et al.*, 2022) as submitted to Journal of Proteome Research.*

Abstract — In metaproteomics, the study of the collective proteome of microbial communities, the protein inference problem is more challenging than in single-species proteomics. Indeed, a peptide sequence can be present not only in multiple proteins or protein isoforms of the same species, but also in homologous proteins from closely related species. To assign the taxonomy and functions of the microbial species, specialized tools have been developed, such as Prophane. This tool, however, is not directly compatible with post-processing tools such as Percolator. In this manuscript we therefore present Pout2Prot, which takes Percolator Output (.pout) files from multiple experiments and creates protein group and protein subgroup output files (.tsv) that can be used directly with Prophane. We investigated different grouping strategies and compared existing protein grouping tools to develop an advanced protein grouping algorithm that offers a variety of different approaches, allows grouping for multiple files, and uses a weighted spectral count for protein (sub)groups to reflect abundance. Pout2Prot is available as a web application at https://pout2prot.ugent.be and is installable via pip as a standalone command line tool and reusable software library. All code is open source under the Apache License 2.0 and is available at https://github.com/compomics/pout2prot.

7.1.1 Introduction

In metaproteomics, the study of the collective proteome of whole (microbial) ecosystems, it is important to learn about the taxonomy and functions represented in the community. For this purpose, tools such as Unipept (Verschaffelt, Van Den Bossche, Martens, *et al.*, 2021) and Prophane (Schiebenhoefer *et al.*, 2020) have been made available to specifically perform downstream

annotation of metaproteomic data, while other, more generic tools also provide connections to downstream annotation tools (Schiebenhoefer *et al.*, 2020; Van Den Bossche *et al.*, 2020; Muth *et al.*, 2015). These tools, however, work very differently: while Unipept relies on identified peptides without inferring the corresponding proteins (a peptide-centric approach), Prophane uses protein groups as input (a protein-centric approach). Recently, these two tools were compared in the first multilab comparison study in metaproteomics (CAMPI), (Van Den Bossche, Kunath, *et al.*, 2021) which indicated that the choice between these approaches is a matter of user preference.

The process of grouping proteins is unfortunately not as straightforward as it might first appear (Martens and Hermjakob, 2007; Uszkoreit *et al.*, 2015; Audain *et al.*, 2017; Nesvizhskii and Aebersold, 2005). Identified peptide sequences have to be assembled into a list of identified proteins, but when a peptide can be mapped to multiple proteins, this leads to the protein inference problem (Nesvizhskii and Aebersold, 2005).

In metaproteomics, this problem is exacerbated due to the presence of homologous proteins from multiple species in its necessarily large protein databases (Schiebenhoefer *et al.*, 2019). Protein grouping is therefore commonly used to generate a more manageable list of identified protein groups that can be used for further downstream analysis. However, different protein grouping algorithms can be chosen, leading to different lists of protein groups from a single set of identified peptides (Martens and Hermjakob, 2007).

In the past, many protein grouping methods have been developed, as reviewed in Audain et al., (Audain *et al.*, 2017) but these typically do not interface well with post-processing tools like Percolator, (Käll *et al.*, 2007) which are able to increase the number of peptide-to-spectrum matches (PSMs) due to a better separation of true and false matches (Bouwmeester *et al.*, 2020). Moreover, the common strategy used by these tools is the Occam's razor strategy, which is not always ideal (Van Den Bossche, Kunath,

et al., 2021).

We here therefore present a new tool, Pout2Prot, which provides users with two relevant protein inference options that are tailored toward metaproteomics use cases: Occam's razor and anti-Occam's razor. Occam's razor is based on the principle of maximum parsimony and provides the smallest set of proteins that explains all observed peptides. Here, however, proteins that are not matched by a unique peptide are discarded, and their associated taxonomy and functions, which might actually be present in the sample, are lost. This algorithm is for example used in the X!TandemPipeline (Langella *et al.*, 2017). On the other hand, anti-Occam's razor is based on the maximal explanatory set of proteins, where any protein that is matched by at least one identified peptide will be included in the reported protein list. This algorithm is used in, for example, MetaProteomeAnalyzer (MPA) (Muth *et al.*, 2015). Unfortunately, there is no simple way to determine a priori which algorithm will be optimal, as this can differ from sample to sample (Muth *et al.*, 2015). These strategies are visually represented in Figure 7.1.

Moreover, as proteins are grouped based on their identified peptides, carefully defined rules are required on when and how to group these proteins. There are two possible approaches here: the first approach consists of grouping all proteins that share one or more identified peptides (i.e., the shared peptide rule), while the second approach consists of only grouping proteins that share the same set (or subset) of identified peptides (i.e., the shared peptide set rule). These two approaches can also be interpreted as grouping at two different levels: the protein group level (based on the shared peptide rule) and the protein subgroup level (based on the shared peptide set rule). These two approaches are also visualized in Figure 7.1.

Pout2Prot implements all of these approaches: Occam's razor and anti-Occam's razor, and both of these at the protein group and protein subgroup level. During conceptualization and testing, we discovered challenges with the naive description of these algorithms. First, different protein subgroups can have the same peptide and therefore have the same spectrum assigned

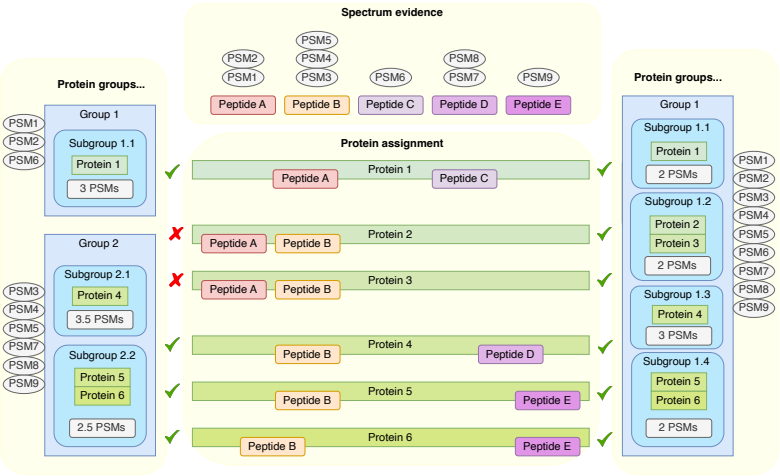

Figure 7.1: Protein grouping algorithms Occam's razor (left) and anti-Occam's razor (right). Groups can be based on shared peptide rule (protein groups) or on shared peptide set rule (protein subgroups). This figure also illustrates how PSMs are assigned to protein (sub)groups and shows the weighted PSM count for subgroups. When a PSM is assigned to multiple subgroups, it will be calculated as one divided by the number of subgroups, which can result in fractional PSM counts.

to them, leading to distorted spectrum counts. Second, when removing proteins using Occam's razor or when assigning subgroups using anti-Occam's razor, "undecidable" cases can occur as illustrated in Figure 7.2. In these undecidable cases, the naive approach might produce inconsistent results when the algorithm is run multiple times.

In this manuscript, we describe a new command line tool and web application that can convert .pout files from different experiments into two files containing protein groups and subgroups either as .tsv for direct use with Prophane or as human readable .csv files. Furthermore, we include a file converter that turns Proteome Discoverer output files into the .pout file format. Thus, Pout2Prot enables Percolator (or Proteome Discoverer users) to use Prophane for downstream functional and taxonomic analysis.

7.1.2 Implementation

Pout2Prot is implemented in Python and installable as a Python package from PyPI. It can then be invoked from the command line. We also provide a user-friendly and easily accessible web application of our tool (https://pout2prot.ugent.be). The transpiler Transcrypt (https://www.transcrypt.org/) was used to convert our Python package into JavaScript-compatible code and reuse it in our web application. Protein grouping analysis is efficient and can, consequently, be performed entirely on the user's local machine. Moreover, the web application processes all data locally, so that no data is sent to our servers. This safeguards user data and allows researchers to analyze confidential information more safely.

The detailed implementation of the protein grouping algorithms is visualized in the Supporting Information (Figure S1 and S2) and consists of four sub-algorithms: the creation of protein groups, the removal of proteins using the rule of maximum parsimony, and a subgroup algorithm each for Occam's razor and anti-Occam's razor.

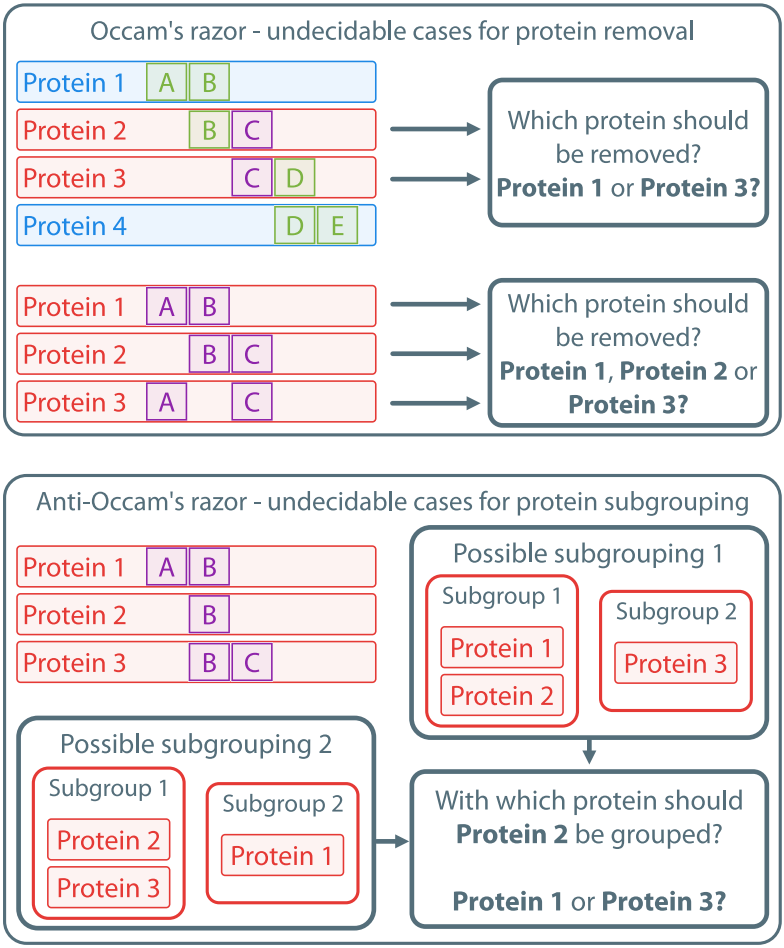

Figure 7.2: Illustration of undecidable cases. Undecidable cases are situations where peptides and proteins are matched in such a way that the naive interpretation of the algorithm cannot make a clear decision. Specifically, this occurs in Occam's razor when one of two or more proteins can be removed to explain the remaining peptides (top), and this occurs in anti-Occam's razor when a protein can be put into a subgroup with two or more other proteins that cannot be subgrouped together (bottom).

7.1.3 Evaluation

Pout2Prot converts .pout files to protein (sub)group files that can be immediately imported in Prophane for further downstream analysis. This Prophane input file consists of four tab-separated fields: sample category, sample name, protein accessions, and spectrum count. The sample category allows users to divide their experiment in different categories (e.g., "control" and "disease"). If no sample categories are provided, these values will be identical to the sample name, which results in individual quantification by Prophane. The sample name is identical to the name of the .pout file, so each protein (sub)group can be traced back to its origin file. The protein accessions will contain the proteins present in the protein (sub)group, based on the chosen strategy. Finally, the spectrum count contains the weighted spectrum count from all PSMs present in that protein (sub)group, with PSMs present in multiple subgroups counted as fractional values in each subgroup.

7.1.3.1 Qualitative comparison to other tools

To develop a protein grouping algorithm and to truly compare different protein grouping tools, the behavior of the algorithm must be validated against a set of well-defined data, where differences between expected and observed behavior (i.e., the composition of the groups) can be clearly distinguished. During the development of Pout2Prot, it quickly became clear that multiple algorithms can solve certain test cases, but fail at others. This also led to the discovery of the undecidable cases outlined in Figure 7.2. Therefore, we created 14 test cases (Supporting Information, Figures S3–S16) that capture all possible pitfalls of protein grouping algorithms, and solved those cases by using both Occam's razor and anti-Occam's razor at the protein group and subgroup level. To resolve the issue of undecidability, we propose that no choice should be made at all. For undecidable cases for protein removal (Occam's razor), no protein should be removed, and for undecidable cases of protein subgroups (anti-Occam's razor), the protein in question should

remain in its own subgroup.

Figure 7.3 shows the result of a comparison between five protein grouping tools: PIA, Fido (integrated into Percolator), MetaProteomeAnalyzer (MPA), X!TandemPipeline, and Pout2Prot. To run tests with each tool, appropriate input files that reflect the test cases were created manually, and these are all available on the Pout2Prot GitHub repository. If a test case did not produce the expected output, it was investigated more closely to ensure this was not the result of differences between, or potential errors in, these input files. For undecidable cases, it was verified that the random choice behavior could be observed (i.e., multiple analyses, different results). For anti-Occam's razor subgrouping Cases 3 and 10, a difference in behavior was observed for PIA and Fido that can be attributed to a different conception of what a protein group is. Specifically, if a protein's peptide set is a strict subset of another protein's peptide set, PIA and Fido will not group these two proteins, while MPA and Pout2Prot will. Of all the tests that could be run, one resulted in an error: the algorithm for X!TandemPipeline for Case 13. In this case, only one of the six proteins was put into a single group, which leads to a situation where one of the three peptides was not explained by the resulting groups.

While we tried to make a fair comparison, it should be noted that PIA also offers and even recommends another option that falls in between Occam's razor and anti-Occam's razor. This method called SpectrumExtractor uses spectrum level information to determine which proteins should be removed or grouped together. Furthermore, Fido offers an option similar to Occam's razor that operates at the level of the protein database. Percolator and other tools (e.g., Triqler (The and Käll, 2019)) assign probabilities to proteins instead of making a binary choice for each protein. In contrast, Pout2Prot is based on the binary model in which a peptide or protein is either identified or not. This choice is influenced by the fact that a probabilistic approach makes the assignment of taxonomies and functions in metaproteomics very difficult.

Tool	Occam Grouping	Occam Subgrouping	Anti-Occam Grouping	Anti-Occam Subgrouping
PIA		case 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14 successful case 12 undecidable		case 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14 successful case 3, 10 different approach
FIDO (Percolator)				case 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14 successful case 3, 10 different approach
MPA			all successful	case 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12 successful case 8, 13, 14 undecidable
X!Tandem Pipeline	case 1, 2, 3, 4, 5, 6, 8, 10, 11, 14 successful case 7, 9, 12 undecidable case 13 incorrect	case 1, 2, 3, 4, 5, 6, 8, 10, 11, 14 successful case 7, 9, 12 undecidable case 13 incorrect		
Pout2Prot	all successful	all successful	all successful	all successful

Figure 7.3: Comparison of the outcome of test cases for five protein grouping tools. The 14 test cases were run with the PIA, Fido (Percolator), MetaProteomeAnalyzer (MPA), X!TandemPipeline, and Pout2Prot. Test cases producing the expected outcome are marked as "successful" (green). Otherwise, these are either categorized as "undecidable" (yellow) if a random choice was made in case of undecidability, "incorrect" (red) if the result cannot be explained logically, and as "different approach" for PIA and Fido, because the anti-Occam protein subgrouping approach used here follows different rules (blue). If a tool does not implement a certain grouping method it is marked as "not implemented" (grey).

7.1.3.2 Performance evaluation

To evaluate the performance of Pout2Prot, we tested it on a metaproteomics data set, derived from the six selected SIHUMIx (Schäpe *et al.*, 2019) data sets used in the Critical Assessment of Metaproteome Investigation (CAMPI) study (Van Den Bossche, Kunath, *et al.*, 2021). Here, we used the X!Tandem (Craig and Beavis, 2004) files available on PRIDE (Perez-Riverol *et al.*, 2019) (PXD023217) to (i) convert these files to Percolator Input (.pin) files with tandem2pin, (ii) process the .pin files with Percolator resulting in Percolator Output (.pout) files, and (iii) convert these .pout files to protein (sub)grouping files with Pout2Prot, once using Occam's razor, once using anti-Occam's razor.

Interestingly, the identification rate (the number of identified spectra divided by the total number of spectra measured) at 1% False Discovery Rate (FDR) increases on average by 7% when using Percolator (Figure 7.4, blue bars (X!Tandem) vs red bars (Percolator)). It is important to notice that Pout2Prot takes into account the PSM FDR, not the protein FDR. As expected and described before, the semisupervised machine learning algorithm Percolator is able to increase the number of PSMs due to the better separation of true and false matches (Käll *et al.*, 2007; Bouwmeester *et al.*, 2020). More interestingly, we examined the effect of Percolator on the number of protein groups and subgroups. To establish the number of protein (sub)groups before Percolator analysis, we reanalyzed the publicly available raw files of the selected data sets with MPA, also using X!Tandem with identical search settings. Note here that MPA is only able to group proteins according to the anti-Occam's strategy, so only those numbers were compared in the section below.

In Figure 7.5, we observe that after Percolator analysis, the number of protein groups per sample increased by 18.5% on average (blue vs red bars) and the number of protein subgroups per sample increased by 25.3% on average (yellow vs green bars). The total number of groups and subgroups across all samples increased more drastically (by 34.7% and 39.9%, respec-

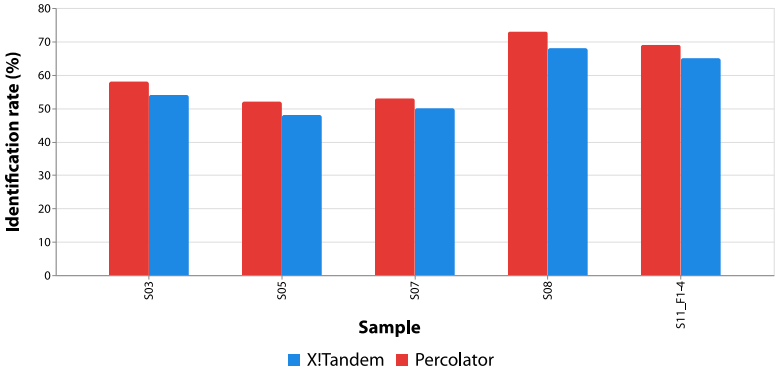

Figure 7.4: Identification rates per sample for X!Tandem and Percolator analyses. Here, the identification rate was defined as the number of identified spectra divided by the total number of spectra measured. S03, S05, S07, S08, S11_F1–4, and S14_Rep1 refer to the six SIHUMIx samples.

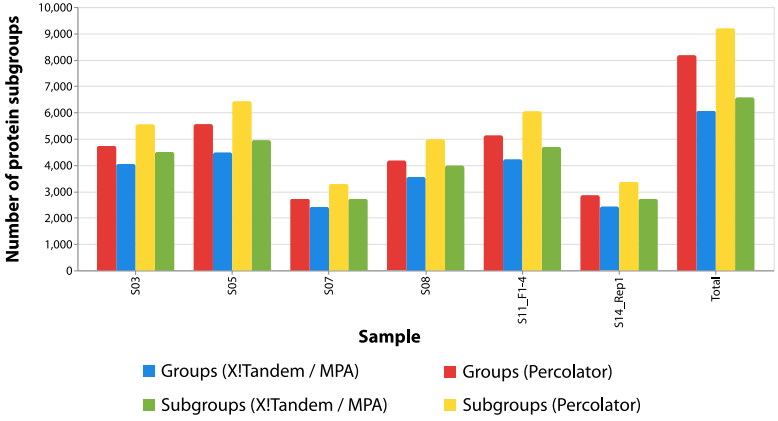

Figure 7.5: Number of protein (sub)groups compared between X!Tandem and Percolator for the anti-Occam's razor strategy, and number of protein (sub)groups using Percolator for the Occam's razor strategy. S03, S05, S07, S08, S11_F1–4, and S14_Rep1 refer to the six SIHUMIx samples.

tively) in comparison to the averages per sample. All raw data is available in Supporting Information (Tables S1 and S2).

Furthermore, we also investigated the effect on the number of protein (sub)groups of combining different fractions at different places in the workflow. We combined (i) the Mascot Generic Format (.mgf) files before the X!Tandem search, (ii) before the Percolator search, and (iii) before Pout2Prot protein inference. Since the range for the number of protein (sub)groups constitute a 2–3% difference, the point in the workflow where the different files are combined, is of minimal impact (Supporting Information, Table S3). For completeness, an example result file for taxonomic and functional analysis after processing of Pout2Prot output in Prophane can be found in Supporting Information, Figures S17 and S18). In addition, the time for a Pout2Prot analysis (Occam's razor) for the complete SIHUMIx experiment via the web service was less than 5s.

7.1.4 Conclusion

Pout2Prot enables the conversion of Percolator output (.pout) files to protein group and protein subgroup files, based on either the Occam's razor or anti-Occam's razor strategy, and therefore closes an important gap in the bioinformatic workflow of metaproteomics data analysis. Moreover, Pout2Prot also allows the user to create protein (sub)groups across experiments. The output of Pout2Prot can be imported directly into Prophane, which in turn allows users to perform downstream taxonomic and functional analysis of metaproteomics samples.

7.1.5 Acknowledgements

This work has benefited from collaborations facilitated by the Metaproteomics Initiative (https://metaproteomics.org/) whose goals are to promote, improve, and standardize metaproteomics (Van Den Bossche, Arntzen, *et al.*, 2021). TVDB, PV, LM, and BM would like to acknowledge the Research Foundation - Flanders (FWO) [grants 1S90918N, 1164420N,

G042518N, and 12I5220N]. LM also acknowledges support from the European Union's Horizon 2020 Programme under Grant Agreement 823839 [H2020-INFRAIA-2018-1]. KS and DB would like to acknowledge the German Federal Ministry of Education and Research (BMBF) of the project "MetaProteomanalyzer Service" within the German Network for Bioinformatics Infrastructure (de.NBI) [031L103]. The authors declare no conflict of interest.

7.2 Highlighting taxonomic diversity of metaproteomic samples in metabolic pathways

7.2.1 Introduction

Since version 4.0 of the Unipept Web application, a functional analysis pipeline for metaproteomics samples was added. This pipeline initially supported EC-numbers and GO-terms, but was soon expanded to include support for InterPro entries. Over the last few years, interest in the functional analysis of metaproteomics samples has been increasing and researchers are more and more trying to gain a deeper understanding of what is happening in an ecosystem of interest.

In addition to the ontologies that are supported by Unipept thus far, the Kyoto Encyclopedia of Genes and Genomes (KEGG) (Kanehisa and Goto, 2000) provides a database with a big collection of metabolic pathways. Metabolic pathways are a series of chemical reactions that occur within a living organism to maintain life. These pathways allow for the synthesis and breakdown of molecules necessary for energy production, growth, and maintenance of cellular processes.

The KEGG resource provides a collection of maps that visualize the reactions that are taking place in a specific metabolic pathway, which enzymes are involved in this reaction and what the start and end product of each reaction is. Because proteins play a major role in these metabolic reactions, we have developed a tool that highlights which parts of a pathway have been identified to be active in a protein sample.

7.2.2 Implementation

The pathway visualization library that we decided to build counts on the pathway maps that our provided by the publicly accessible KEGG `API`. This KEGG `API` allows third-party applications to integrate KEGG pathway maps into their own software applications and to highlight specific nodes of

these maps in one or more colours. A machine-readable format for each of the pathway map images is also provided, but we soon found out that these are typically incomplete. Since the KEGG service overlays each pathway map image with a collection of HTML elements (including their size and positioning), we were able to extract the required information from there.

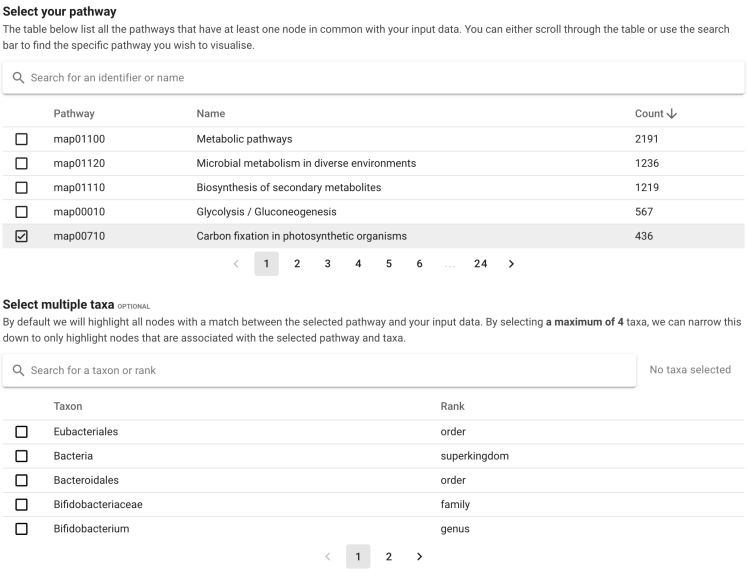

Figure 7.6: Once a peptide or protein input sample has been analysed, the web application displays a list of all metabolic pathway maps that are applicable for the provided input and requests the user to select a map of interest.

Our web application is implemented in TypeScript and uses the Vue and Vuetify frameworks for the development of a concise graphical user interface. It accepts a wide range of different input formats containing either peptide or protein related information and quickly analyses each of the input files to detect in which metabolic pathway reactions each of the input peptides or proteins are involved. Afterwards, the application will show a list of all pathway maps that are applicable to the input sample and highlights all identified metabolic reactions on the map. Figure 7.6 shows the

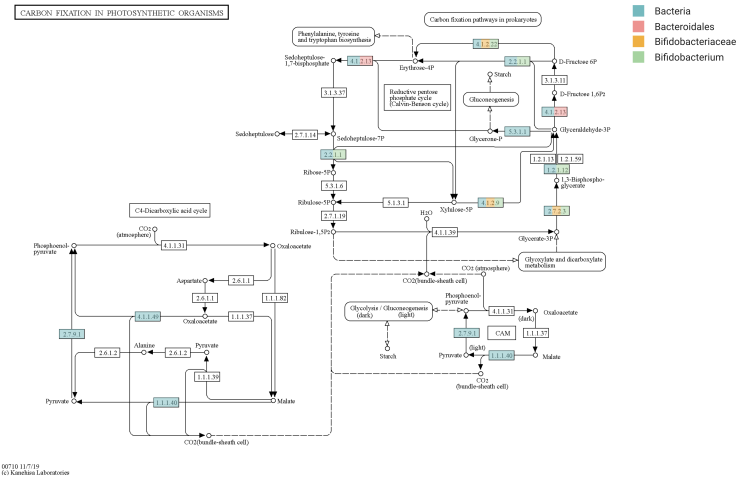

Figure 7.7: After selecting a pathway map and a series of organisms, the final results are visualised by the web application.

pathway selection tool of the web application and Figure 7.7 shows an image of the final visualization result that is produced by the application.

7.3 Efficiently exploiting parallelism in modern web applications

7.3.1 Introduction

JavaScript is one of the most popular programming languages at this point in time. According to a report of GitHub, it was the number one most used programming language in 2022.[1] In recent years, web applications have become a viable alternative for desktop applications and are increasingly favored by software developers. Because of the increased functionality that is provided by these web applications, they have also grown in complexity and started to adopt some of the programming paradigms that are traditionally used by desktop applications.

In order to efficiently process large amounts of data, software developers try to split up hard tasks into smaller tasks that can be processed in parallel by the different cores in modern-day CPUs. Since web applications are almost exclusively relying on JavaScript, this programming language of the web has adopted support for multithreading by implementing the "Web Workers" construct.

A "Web Worker" is a JavaScript script that is executed by the browser using a background thread. Web applications can provide them with a collection of input data, instruct it to process the data and receive the results when done, all without occupying the browser's "main thread".

In order to understand what the "main thread" is in JavaScript, you need to realise that JavaScript is an event-driven programming language. Every single operation that is performed by a piece of JavaScript code will be sent to a queue that is systematically queried and emptied by the "Event loop". The main JavaScript thread is constantly checking this event queue for new tasks that need to be taken care of and executes them one-by-one in a specific order. Because only one thread is available to process the tasks

[1] See https://octoverse.github.com/2022/top-programming-languages

pushed onto this queue, a long-running task blocks the execution of other tasks and can cause the web browser to "hang" on a specific operation. It will only continue to process interactions of the user with a web app's user interface once this long-running task is completed (which is not user-friendly).

In the past, this was never really an issue since JavaScript was typically only used for providing some simple interactivity to a web page, but since the advent of complex web apps, this is becoming a major hurdle. The amount of data that needs to be handled by modern-day web applications has seen a tremendous growth and can no longer be efficiently processed with a single thread.

7.3.2 Web Workers to the rescue

An initial proposal to add a "Web Workers" construct to the JavaScript programming language was first suggested in the ECMAScript 5 standard and has been formally adopted by all major browsers at this point. A Web Worker can be defined as a task that receives some data as input, processes in the background and notifies the main thread when it is done. These workers are typically managed by a separate browser process and can thus be executed in parallel to the tasks of the main JavaScript process.

Since each web worker runs in a separate process, they don't share the same memory space. A reference to an object or piece of data that lives in one thread cannot be simply transferred to a web worker. Instead, every object that needs to be sent between a web worker and the main JavaScript thread needs to be serialized on the sender's side and reconstructed on the receiver's side.

The "structured clone algorithm" is a mechanism in JavaScript that allows for the deep copying of complex objects in order to transmit or store them in a serialized format (see Figure 7.8). If an object is transferred between contexts in JavaScript, this algorithm will be used. This works very well for

simple and small objects, but becomes slow very soon when large chunks of data need to be transferred and partially negates some important benefits of using Web Workers.

To make matters even worse, either one of the serialization or deserialization of an object is always performed by the main JavaScript thread, causing the application to hang again (which is one of the problems we are trying to overcome).

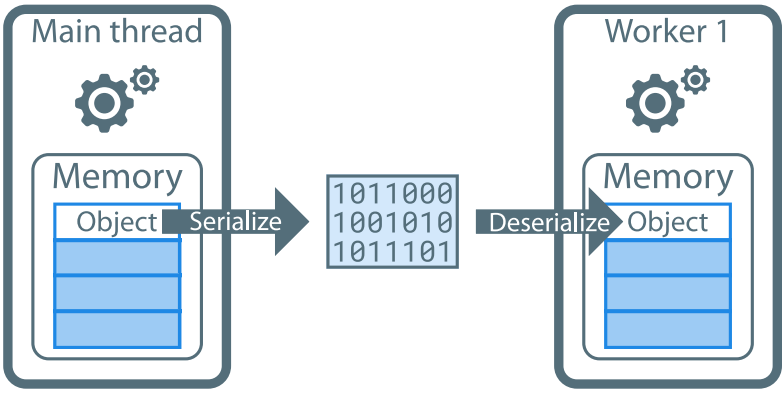

Figure 7.8: If a normal JavaScript Object is being sent from one worker (or the main thread) to another, it first needs to be serialized on the sending side using the "structured clone algorithm". On the receiving end, it will then be completely deserialized again.

7.3.2.1 Near zero-cost copy of ArrayBuffers

Since a few years, JavaScript exposes a new type of object called an `ArrayBuffer`. An `ArrayBuffer` is a built-in object that represents a fixed-length raw binary data buffer. This means that it allows a software developer to store a sequence of bytes that can be accessed and manipulated in a low-level way. Such an `ArrayBuffer` is similar to a normal JavaScript array in that it is a collection of values, but the values in this `ArrayBuffer` are binary data instead of rich values such as numbers or strings.

Since the `ArrayBuffer` is just a series of binary values, it can also be thought of as a block of memory. Because of its very simple structure, an `ArrayBuffer` does not need to be copied between different Web Workers, but instead only "ownership" of this memory block needs to be transferred (see Figure 7.9). The thread or Web Worker that currently has the "ownership" of an ArrayBuffer is the only one that is allowed to make changes to, or read from the block of memory at that point. Transferring ownership is almost instantly.

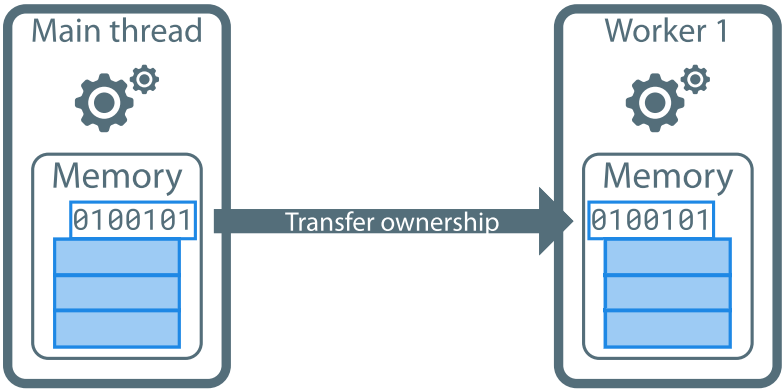

Figure 7.9: When sending an ArrayBuffer from one worker to another, it's ownership will be transferred. This means that no data needs to be copied which makes this operation a lot faster than for normal JavaScript Objects since the "structured clone algorithm" is not involved.

7.3.2.2 Sharing data between Web Workers

Next to `ArrayBuffers`, modern JavaScript specifications also describe a new concept called a `SharedArrayBuffer`. Contrary to `ArrayBuffers`, `SharedArrayBuffers` no longer need to be "transferred" between Web Workers. Instead, a `SharedArrayBuffer` object provides a "view" on a contiguous block of memory and can be "transmitted" to other workers in which case a new `SharedArrayBuffer` object will be created on the receiving side which is simply a view onto the same block of memory (see Figure 7.10).

The shared data block referenced by the two `SharedArrayBuffer` objects is the same block of data, and a side change made to this block of memory in one worker, will also become visible to the other worker.

If we compare `ArrayBuffers` to `SharedArrayBuffers` when it comes to the transfer of information from one Web Worker to another, we can conclude that `SharedArrayBuffers` allow multiple Web Workers to read and write to the same block of memory at the same time. This hidden feature of JavaScript is something that we decided to exploit in order to speed up the Unipept Web and Unipept Desktop applications. How we were able to do so is explained in depth in the next section.

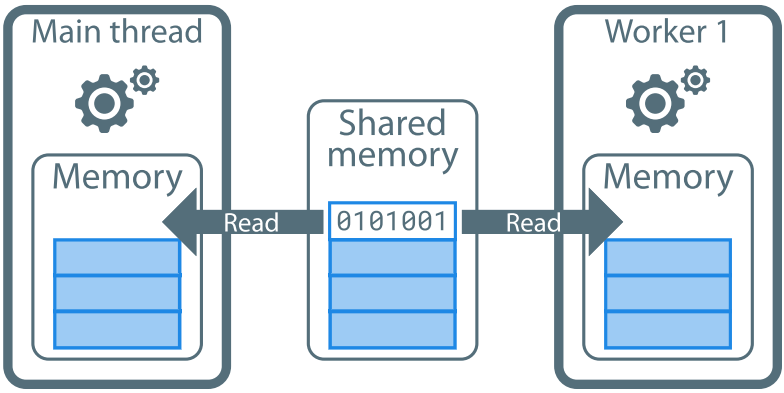

Figure 7.10: SharedArrayBuffers point to a specific block of memory that can be modified and used by different workers at the same time. This allows applications to split intensive operations and let them be executed by different workers in parallel which can then all read and write from the same HashMap.

7.3.3 A shared-memory HashMap in JavaScript

7.3.3.1 General structure of the HashMap

At this point, it is clear that there are constructs that allow the communication of large data sets between different Web Workers. For most structured data, however, it is not trivial to encode it as a stream of raw bits and

bytes. In order to accommodate for this issue, we decided to implement a `HashMap` that allows arbitrary data and objects to be encoded as a stream of bits in an `ArrayBuffer` or `SharedArrayBuffer` that can then easily be transferred between threads.

Our `HashMap` implements the interface that is provided by JavaScript and is thus fully compatible and interchangeable with pieces of code that use the default `Map` implementation of JavaScript. It follows the idea of most `HashMaps` that are already implemented in other programming languages such as Java. In short, there is one block of memory that can keep track of n pointers (referred to as the "index table" from now on). Since every element in a `HashMap` has both a key and a value, we hash the value of the key and use this hash to determine at what position in the "index block" a point to the corresponding value can be found.

For a `HashMap`, a hash function typically needs to generate hashes as fast as possible that are distributed evenly. We chose the Fowler-Noll-Vo hash function (Fowler *et al.*), which can be computed really fast on modern CPUs and produces hashes that are evenly distributed. Another advantage of this hashing algorithm is that there already exists a good implementation of it for JavaScript.[2]

Each generated hash is represented as a large number. In order to map this number onto a specific position in the index table, we simply compute the remainder of the hash when divided by n (the size of the index table). So, in order to retrieve the value that's associated with a specific key, we first compute the hash of the key, then find out its remainder when divided by n and look at the pointer in the index table at this position. Figure 7.11 shows a detailed example of how a lookup for the key "cat" is performed. Note that it might happen that multiple keys are mapped onto the same position in the index table, which is why every value object always keeps track of the original key and a pointer to the next item that is mapped onto the same index. When retrieving a value for a key, this linked list of value

[2] See https://www.npmjs.com/package/fnv-plus

objects needs to be processed until the key is found, or until the list ends. See this article on Wikipedia for more information on how a `HashMap` works internally: https://en.wikipedia.org/wiki/Hash_table.

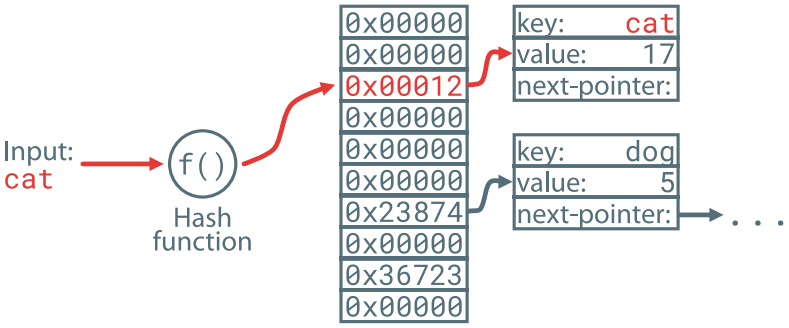

Figure 7.11: Looking up a value in the HashMap consists of several steps. First the key is hashed, then the position of this hash in the index table is computed. At last, the pointer at this computed position in the index table can be used to retrieve the value associated with the key.

7.3.3.2 Internal memory lay-out

My `HashMap` implementation requires the reservation of two blocks of memory:

- **index table:** This block of memory keeps track of all pointers to data objects that keep track of the key and value for a `HashMap` entry and also a pointer to the next data objects. Each of these data objects live in the second reserved block of memory (the "data block"). Some extra bytes are also reserved as part of this memory block at the beginning for internal housekeeping of the `HashMap`.
- **data block:** This block of memory keeps track of all data objects that actually store the values that the user puts into the `HashMap`.

Each of the values that are provided by the user need to somehow be translated into bytes before we can store them in a raw block of memory.

For some of the most common data types in JavaScript (e.g. String, Number,

etc), a default serialization implementation is provided. This is not the way most `HashMaps` are implemented in other programming languages. Normally, the values themselves are not serialized and stored as part of the `HashMap`, but rather a pointer to each of the values is kept, decreasing the amount of memory required. Because this is a high-level implementation of a `HashMap`, we don't have access to the raw object pointers that are used internally by the JavaScript engine and we have to reside to this workaround.

7.3.3.3 Serialization of complex objects

Since serialization of objects can be very slow in some cases, the `ShareableHashMap` allows the user to provide a custom serialization (and deserialization) function. These allow for some really nice optimizations and can circumvent the need to convert objects to strings and bytes altogether, leading to a significant speed up. By cleverly exploiting the structure of some objects, we can encode objects as streams of bytes and directly extract those bytes that are associated with a specific property of the object.

7.3.4 Case study: keeping track of peptides in Unipept

In order to demonstrate the power of this specific `HashMap` implementation in JavaScript, we will be looking at its specific use in the Unipept Web and Desktop application. For each metaproteomic analysis, both Unipept applications need to keep track of the taxa and functions associated with each peptide that was provided by the user. This information is queried from Unipept's API, but since this querying process takes a long amount of time, it is performed by a Web Worker in the background and once done, a big `Map` containing the peptide/results mapping is passed onto the main thread.

During all of this time, the deserialization of all information is performed by the main thread and the application is thus unresponsive to all interactions of the user.

Every `(key, value)` pair in this mapping has a specific structure. The keys will always be peptides (i.e. strings) and the values are JSON-objects that keep track of some annotations for this peptide. Since most properties in this object have a fixed length, or the property value lengths are known beforehand, we can encode these objects as streams of bytes in an ArrayBuffer. See Figure 7.12 for an example of how the information that's tracked by Unipept can be respresented by a stream of bytes. Using this information, it is no longer required to serialize this object to a string-based representation (such as JSON).

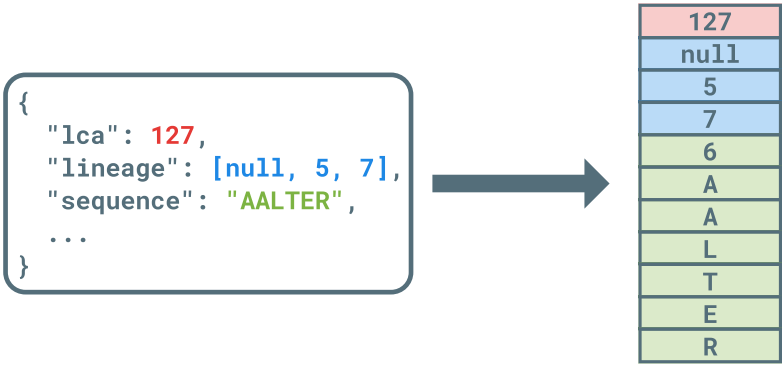

Figure 7.12: Example of how a complex object (in the case of Unipept) can be encoded as a simple stream of bytes that directly fits into an ArrayBuffer. We know that the lowest common ancestor is always an unsigned integer, so we can store it in the first 4 bytes of a block of memory. The lineage in this example is a numeric array that always contains 3 unsigned integers, thus the next three places in the ArrayBuffer are reserved for this array. By continuing this strategy, each of the properties can directly be encoded in the memory block and can be recovered very efficiently.

In the case of Unipept specifically, we were able to reduce a memory transfer of 30 seconds back to only a few milliseconds because of this custom `HashMap` implementation. Not only did an analysis take 30 seconds longer before this improvement, the Unipept Web and Unipept Desktop application where completely unresponsive to user interaction during this timespan.

7.3.5 Conclusion and remarks

Based on the results from the case study, it is fair to conclude that this `HashMap` is not suitable for all projects, but can be of very high value in a specific environment (as is the case with Unipept). In order to counteract the effects of some serious vulnerabilities that were dedected in x86 CPUs (i.e. the Spectre (Kocher *et al.*, 2019) and Meltdown (Lipp *et al.*, 2018) attacks), most major browsers have taken serious precautions to counteract these attacks and blocked the use of `SharedArrayBuffers` in most cases. Only websites that pack a specific set of HTTP headers into their HTTP responses[3] are allowed to use SharedArrayBuffers.

This means that consumers of our `HashMap` implementation either need to resort to regular `ArrayBuffers` if they don't need multiple Web Workers to manipulate the `HashMap` simultaneously or that they need to properly configure their servers in order to take care of the required Cross-Origin Isolation headers.

[3] See https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/SharedArrayBuffer#security_requirements

Chapter 8

Future work

No scientific work is ever complete and there remain a lot of challenges in the field of metaproteomics and proteogenomics that still need to be solved. In this section, I discuss some issues and challenges that currently arise and what needs to be overcome in order to solve these.

8.1 Modelling the inherent ambiguities in the Unipept matching system

8.1.1 Current situation

In a process prior to database construction, peptide sequences are reconstructed by spectral search engines. During the construction of the protein reference database, Unipept aggregates the functional and taxonomic annotations of proteins by grouping them by exact matching of peptide sequences. Since a single mass spectrum can, however, be explained by different peptide sequences, a search engine sometimes needs to pick and output the most probable peptide sequence that explains this spectrum. This inherent ambiguity is ignored during the Unipept database construction process by only considering peptide sequence similarity when grouping and summarising the functional and taxonomic annotations of peptides.

Right now, all proteins that are fed into Unipept during database construction will be in-silico tryptically digested into peptides. Then, a similarity function is used to compare peptides with each other, and the protein-level annotations of peptides that are found to be equal, according to this similarity function, are grouped and summarised. Currently, two peptides are considered to be equal when their peptide sequences are exactly the same. In this case, we rely on the search engines that are matching experimental mass spectra with peptide sequences in order to get a reliable result. In practice, it is common for multiple peptide sequences to correspond with the same mass spectrum, causing ambiguity in the spectral matching process. But, since Unipept only looks at sequence identity, part of this spectral ambiguity is (potentially unjustified) ignored.

8.1.2 Proposed work plan

Instead of grouping together different peptides by only looking at the sequence similarity, we can predict the mass spectrum of each peptide in the Unipept protein reference database using a tool such as MS2PIP (Degroeve

and Martens, 2013). MS2PIP employs the XGBoost machine learning algorithm in order to predict MS2 signal peak intensities from peptide sequences and has proven to produce very reliable results. Tryptic peptides that were identical when using the sequence-based similarity metric, will also be identical with this new metric. But important to note is that the spectrum-based similarity metric will be "less strict" than sequence-based similarity, meaning that peptides that were different under the sequence-based similarity can now be considered identical. Consequently, this also means that resolution of the taxonomic and functional profile for a metaproteomics sample will go down. By now taking into account the spectral ambiguity that was previously masked by Unipept's analysis, we can design an experiment to investigate to what extent this ambiguity proves to be an issue.

8.1.2.1 Update the Unipept database construction process

The construction process of the Unipept database is currently not designed to work with different similarity metrics when it comes to grouping peptides. A first, big change, will have to be made to this construction process in order to allow it to accept arbitrary similarity metrics. This will, in turn, allow us to implement the spectrum based similarity metric (as well as variants) and easily plug them into the database construction process. No other major changes will have to be made to the finalised Unipept database, the underlying database structure will be more or less the same.

8.1.2.2 Perform a first experiment

In order to test the hypothesis that we proposed at the start of this section, we will have to perform an experiment in which we compare the end result of a metaproteomics analysis using the updated spectrum-based similarity approach for Unipept against using the traditional sequence-based similarity approach. For this experiment, we can analyse the SIHUMI sample using each of the approaches and compare the end results. SIHUMI is a sample that was artificially constructed for the renowned CAMPI study (Van

Den Bossche, Kunath, *et al.*, 2021) and for which the exact taxonomic composition is known. We expect to find that the spectrum-similarity based Unipept leads to a lower, but more realistic, taxonomic and functional resolution for the provided sample.

8.1.2.3 Predict retention times

Instead of only looking at the predicted MS2 peak intensities of a peptide sequence using MS2PIP, we can go one step further and also predict retention times and take these into account and expand the similarity metric that was developed during the previous steps. Retention times can be predicted using the DeepLC tool (Bouwmeester *et al.*, 2021) and will cause some peptides, that are similar when we compare them solely by spectra, to be different if we also take into account this predicted retention time. Since the Unipept database construction process has already been updated to be compatible with hot-pluggable similarity metrics at this point, we only need to implement a new similarity metric and rebuild the Unipept database.

8.1.2.4 Perform a second experiment

Finally, we can augment the experiment designed earlier to compare the results between the updated spectrum-based similarity approach for Unipept against using the traditional sequence-based similarity approach with the analysis using the spectrum-similarity based Unipept that also takes into account retention times for the tryptic peptides. For this comparison, we expect the taxonomic and functional resolution of the end result to have increased when comparing it to the spectrum-based similarity of before.

8.2 Identification and analysis of arbitrary peptides, including variants

8.2.1 Current situation

Unipept requires that all input peptides are tryptic in order to be able to match them with peptides in its reference database. However, researchers are transitioning to experiment with datasets that contain other peptide formats that Unipept currently can not use for downstream analysis. In order to accomodate this change, we could design a new index structure for Unipept based on bidirectional FM-indices and search schemes which will no longer require the input peptides to adhere to a fixed format.

Over the last 10 years, a lot of research has gone into the development and improvement of efficient data structures for sequence alignment. One such highly-used data structure that offers excellent performance is the FM-index (Ferragina and Manzini, 2000). An FM-index is produced by computing the Burrows-Wheeler transform of a specific string and allows to look up if (and where) a pattern occurs in the preprocessed text in a very efficient manner. By adjusting the FM-index and its accompanying query algorithms, we are not limited to matching exact strings but we can also detect if a specific sequence (with up to a certain number of k mismatches) is present in a longer string.

A big advantage of these FM-indices over the approach that Unipept currently follows for matching input peptides with proteins in the protein reference database is that the FM-index allows us to match arbitrary peptide sequences with proteins (instead of only directly matching tryptic peptides as we do today). This opens up the possibility to go and analyse semi-tryptic peptides using Unipept, or even matching tryptic peptides with missed cleavages. At this point, it is already possible to analyse peptides with missed cleavages, but this drastically slows down the analysis since a lot of Unipept's precomputed aggregations are not available in this case.

In order to efficiently match a peptide (with up to k mismatches) with a protein, an FM-index by itself does not suffice and we need to look for improved data structures. This is where search schemes come into play. A search scheme is a strategy that describes how a bi-directional FM-index can be queried such that patterns with up to k mismatches can efficiently be matched with a long string (such as a protein). Search schemes were first proposed by (Lam et al.) and were further generalised by (Kucherov *et al.*, 2014).

8.2.2 Proposed work plan

8.2.2.1 Design and implement a new index structure for Unipept

The first step that should be performed in order to allow Unipept to match peptides of arbitrary format, is to design a new index structure for our database, using FM-indices, and to implement this new index structure. Each protein in the protein reference database, will be added to a generalised FM-index that will eventually contain all proteins from the reference database and which allows us to match any kind of peptide. We can use the Rust programming language since it is designed with performance and parallelization in mind, and it already provides a very good, open-source implementation of the FM-index data structure.[1] These changes will allow us to match arbitrary peptides and peptides with missed cleavages using Unipept.

8.2.2.2 Implement a bi-directional FM-index

A second step consists of updating the FM-index data structure that was used during the previous step such that it supports matching patterns in two directions (backwards *and* forwards). This so-called bi-directional FM-index is extensively described in (Lam *et al.*) and is required for efficiently performing approximate pattern matching using search schemes. We can improve and expand the existing, open-source Rust FM-index

[1] https://docs.rs/fm-index/latest/fm_index/

implementation from the previous step such that it allows searching in two directions. By contributing to this open-source project we do not need to start from scratch, and we can share our improvements with other researchers around the globe.

8.2.2.3 Implement and validate search scheme prototypes

A lot of different proposals for search schemes already exist at this point. During this step, we can take a look at a selection of search schemes such as the Pigeon H.S. (Fletcher and Patty, 1996), 01*0 seeds (Vroland *et al.*, 2016), schemes proposed by Kucherov (Kucherov *et al.*, 2014), and Man_{best} (Pockrandt, 2019). All of these search schemes will have to be benchmarked for performance and applicability for our needs. The search scheme that comes out as best from this comparison can then be tweaked and refined further.

8.2.2.4 Integrate the best search scheme into Unipept

Finally, the search scheme that was selected during the previous step needs to be integrated into the Unipept database index structure. By doing so, Unipept will effectively support matching peptides with up to k mismatches with the proteins from a reference database.

8.3 Towards a meta-, multi-omics Unipept

8.3.1 Current situation

Over the last few years, we have been working very hard to build the Unipept Desktop application which provides a first step in the integration of metagenomics information in metaproteomics experiments. By first performing a metagenomics experiment on a sample, researchers are able to derive the taxonomic profile of the ecosystem under study. This taxonomic profile can then be used in a subsequent step as a guide for constructing a targeted protein reference database (that only contains proteins that

are associated with the taxa that were detected during the metagenomics experiment).

As a possible future addition to Unipept, I propose to further integrate data from different "omics" sources such as transcriptomics and metagenomics into Unipept. Building on the individual strength of these techniques, an aggregated view enables researchers to gain a much deeper insight into and understanding of what exactly is taking place in a complex ecosystem. By augmenting Unipept with support for both metagenomics and metatranscriptomics analyses, it has the potential to become the "go-to" tool for all analyses related to the "meta-omics" research disciplines. The ultimate goal of this addition would be to transform Unipept into the first tool that provides a complete global overview of multi-disciplinary "meta-omics" experiments.

8.3.2 Proposed work plan

8.3.2.1 Allow Unipept to directly load metagenomics reads

By improving Unipept with the capability of loading metagenomics reads directly into the software, we can allow users to construct a fully custom protein reference database from these reads. At this time, a targeted reference database is always constructed by extracting and filtering proteins from the UniProtKB resource. This works very well when the organisms under study have been analysed before and their proteomic profile is available in the UniProtKB resource. By providing support for the construction of protein reference databases from metagenomic reads instead, we can also allow organisms that are not present in the UniProtKB resource to be analysed.

8.3.2.2 Design and implement new visualizations for metagenomics experiments

Unipept provides a lot of valuable and interactive data visualisations that increase the insight of researchers into the taxonomic and functional profile of a metaproteomics ecosystem. These visualisations are implemented in a

very generic way that allows them to be applicable to other situations as well. I propose to expand Unipept with the ability to visualise the taxonomic profile determined by a metagenomics experiment, and the potential functional profile that is determined by performing a metatranscriptomics experiment. This will increase the insight of users into the ecosystem that they are currently investigating.

8.4 Differential analysis of metaproteomics data

8.4.1 Current situation

Dedicated data analysis methods for differential metaproteomics are currently lacking. Different tools for the analysis of metaproteomics data exist at this point, but none of them provide a statistically sound framework for the **comparative** analysis of metaproteomics data.

By devising a framework that allows for filtering and normalisation, we can take advantage of the parallel data structures in metaproteomics and we can extent beyond two-group comparisons by building on the general linear model (GLM) framework. This approach provides researchers with the tools to assess differential abundance in both taxa, as well as in functional annotations between conditions. Moreover, we could also devise methods to detect shifts in the population or the use of functions induced by treatment or biological processes such as diseases.

Ideally, this comparative analysis pipeline can be integrated and embedded into user-friendly workflows for Unipept.

8.4.2 Proposed work plan

8.4.2.1 Develop and validate normalisation and filtering strategies for metaproteomics applications

A Unipept analysis will map peptides to taxa and functions resulting in a count table, but these counts can be largely driven by library size, i.e.,

the total number of peptides in a sample. Hence, the data need to be normalised, which will be done by calculating normalisation offsets for the GLMs that will be developed in the next step. Besides library size, we also have to evaluate the use of other features, e.g., average peptide length, for more advanced normalisation using a CQN approach (Hansen *et al.*, 2012).

8.4.2.2 Establish a generalized linear model (GLM)

The second step in my proposed work plan consists of establishing a generalised linear model (GLM) framework for count regression in metaproteomics. Development will start from well-established tools for modelling bulk RNA sequencing (RNA-seq) data. A preliminary edgeR (Robinson *et al.*, 2010) analysis that we've performed shows that metaproteomics counts exhibit a similar mean-dispersion relation as RNA-seq data. Hence, negative binomial (NB) count regression and existing empirical Bayes methods to stabilise the dispersion estimation seem to be promising for metaproteomics data analysis applications. The GLM framework also naturally extends the analysis beyond two group comparisons and enables for normalisation. Indeed, experiments with more complex designs can be accommodated for by including factors with multiple levels and even continuous covariates in the model, and normalisation offsets can be included in the linear predictor. Finally, the challenge to detect shifts in communities, e.g., at a certain taxonomic level, shows many similarities with assessing differential transcript usage in bulk RNA-seq experiments, where researchers try to discover shifts in the relative abundance of isoforms within a gene. Again, we will have to port and tailor these tools towards metaproteomics.

8.4.2.3 Perform statistical inference

Within the GLM framework, likelihood ratio tests, score tests and Wald tests can be conducted. We can evaluate their performance within an empirical Bayes context and in terms of computational efficiency. Next, false discovery rate (FDR) methods should be adopted and more advanced FDR

methods will be developed for metaproteomics applications. Specifically, we can build upon stage-wise testing procedures that have proven their merits for transcript level analysis and gene-set enrichment analysis (Van den Berge *et al.*, 2017).

8.4.2.4 Develop a user-friendly workflow and integrate into Unipept

The last step that I can propose for this workplan is to integrate all of the methods and strategies that where developed during the previous tasks into a user-friendly workflow for the Unipept web application. For each of the steps we will have to consider whether to implement the step in JavaScript and run it directly in the browser, or to offload it to an R instance running server-side. The latter will probably require less implementation work because of the available R packages but comes at the cost of less flexibility and interactivity. During the implementation phase, it is extremely important to gather feedback from end-users to iterate on the proposed solution.

Figure attributions

Some of the figures used in this document originate from sources that I would like to attribute here.

Figure 1.1

- Book icon is designed by Vector Place of The Noun Project.
- DNA Strand icon is designed by Symbolon of The Noun Project.
- Pizza Slice icon is designed by Aidan Stonehouse of The Noun Project.

Figure 1.2

- Cell icon is designed by Léa Lortal of The Noun Project.
- Chromosome icon is designed by Mette Galaxy of The Noun Project.
- DNA icon is designed by Irene Hoffman of The Noun Project.

Figure 1.3

- DNA icon is designed by Irene Hoffman of The Noun Project.

Figure 1.4

- Organism icon is designed by Nithinan Tatah of The Noun Project.

Figure 1.5

- Spectrometer icon is designed by M. Oki Orlando of The Noun Project.
- Scissors icon is designed by Bmijnlieff of The Noun Project.

Figure 1.8

- File icon is designed by Khoiriyah of The Noun Project.
- Application Code icon is designed by Damian Hetman of The Noun Project.

Figure 1.9

- Bacteria icon is designed by Tanvir Islam of The Noun Project.
- Virus icon is designed by Farwadi Sofi of The Noun Project.
- Virus icon is designed by Cetha Studio of The Noun Project.

Figure 1.10

- Bacteria icon is designed by Tanvir Islam of The Noun Project.
- Virus icon is designed by Farwadi Sofi of The Noun Project.
- Virus icon is designed by Cetha Studio of The Noun Project.

Figure 1.11

- Bacteria icon is designed by Tanvir Islam of The Noun Project.
- Virus icon is designed by Farwadi Sofi of The Noun Project.
- Virus icon is designed by Cetha Studio of The Noun Project.

Figure 3.1

- Bacteria icon is designed by Tanvir Islam of The Noun Project.
- Virus icon is designed by Farwadi Sofi of The Noun Project.
- Virus icon is designed by Cetha Studio of The Noun Project.

Figure 3.2

- Bacteria icon is designed by Tanvir Islam of The Noun Project.
- Virus icon is designed by Farwadi Sofi of The Noun Project.
- Virus icon is designed by Cetha Studio of The Noun Project.

Figure 3.4

- Cloud icon is designed by Ria Fitriana of The Noun Project.
- CPU icon is designed by Iconbysonny of The Noun Project.
- Magnifying Glass icon is designed by Larea of The Noun Project.
- ZIP Folder icon is designed by Nack_Thanakorn of The Noun Project.
- Cog icon is designed by Creative Stall of The Noun Project.

Figure 7.8

- Cog icon is designed by Creative Stall of The Noun Project.

Figure 7.9

- Cog icon is designed by Creative Stall of The Noun Project.

Figure 7.10

- Cog icon is designed by Creative Stall of The Noun Project.

References

Ashburner,M. *et al.* (2000) Gene Ontology: Tool for the unification of biology. *Nature Genetics*, **25**, 25–29.

Ashwood,C. *et al.* (2020) Proceedings of the EuBIC-MS 2020 Developers' Meeting. *EuPA Open Proteomics*, **24**, 1–6.

Audain,E. *et al.* (2017) In-depth analysis of protein inference algorithms using multiple search engines and well-defined metrics. *Journal of Proteomics*, **150**, 170–182.

Bar-Joseph,Z. *et al.* (2001) Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, **17**, S22–S29.

Benchmark and improving methods in metaproteomics informatics (2022).

Bibbò,S. *et al.* (2020) Fecal Microbiota Signatures in Celiac Disease Patients With Poly-Autoimmunity. *Frontiers in Cellular and Infection Microbiology*, **10**, 349.

Bierman,G. *et al.* (2014) Understanding TypeScript. In, Jones,R. (ed), *ECOOP 2014 Object-Oriented Programming*, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 257–281.

Bittremieux,W. (2020) Spectrum_utils: A Python Package for Mass Spectrometry Data Processing and Visualization. *Analytical Chemistry*, **92**, 659–661.

Bostock,M. *et al.* (2011) D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, **17**, 2301–2309.

Bouwmeester,R. *et al.* (2021) DeepLC can predict retention times for peptides that carry as-yet unseen modifications. *Nature Methods*, **18**, 1363–1369.

Bouwmeester,R. *et al.* (2020) The Age of Data-Driven Proteomics: How

Machine Learning Enables Novel Workflows. *PROTEOMICS*, **20**, 1900351.

Bucchini,F. *et al.* (2021) TRAPID 2.0: A web application for taxonomic and functional analysis of de novo transcriptomes. *Nucleic Acids Research*, **49**, e101.

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

Crick,F. (1970) Central Dogma of Molecular Biology. *Nature*, **227**, 561–563.

Degroeve,S. and Martens,L. (2013) MS2PIP: A tool for MS/MS peak intensity prediction. *Bioinformatics*, **29**, 3199–3203.

Easterly,C.W. *et al.* (2019) metaQuantome: An Integrated, Quantitative Metaproteomics Approach Reveals Connections Between Taxonomy and Protein Function in Complex Microbiomes *. *Molecular & Cellular Proteomics*, **18**, S82–S91.

Federhen,S. (2012) The NCBI Taxonomy database. *Nucleic Acids Research*, **40**, D136–D143.

Ferragina,P. and Manzini,G. (2000) Opportunistic data structures with applications. In, *Proceedings 41st Annual Symposium on Foundations of Computer Science*., pp. 390–398.

Finn,R.D. *et al.* (2017) InterPro in 2017beyond protein family and domain annotations. *Nucleic Acids Research*, **45**, D190–D199.

Fletcher,P. and Patty,C.W. (1996) Foundations of Higher Mathematics PWS Publishing Company.

Fowler,G. *et al.* The FNV Non-Cryptographic Hash Algorithm Internet Engineering Task Force.

Fruzangohar,M. *et al.* (2013) Comparative GO: A Web Application for

Comparative Gene Ontology and Gene Ontology-Based Gene Selection in Bacteria. *PLOS ONE*, **8**, e58759.

Gruvaeus,G. and Wainer,H. (1972) Two Additions to Hierarchical Cluster Analysis. *British Journal of Mathematical and Statistical Psychology*, **25**, 200–206.

Gurdeep Singh,R. *et al.* (2019) Unipept 4.0: Functional Analysis of Metaproteome Data. *Journal of Proteome Research*, **18**, 606–615.

Hansen,K.D. *et al.* (2012) Removing technical variability in RNA-seq data using conditional quantile normalization. *Biostatistics*, **13**, 204–216.

Herbst,F.-A. *et al.* (2016) Enhancing metaproteomicsThe value of models and defined environmental microbial systems. *PROTEOMICS*, **16**, 783–798.

Huang,D.W. *et al.* (2009) Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nature Protocols*, **4**, 44–57.

Huerta-Cepas,J. *et al.* (2019) eggNOG 5.0: A hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. *Nucleic Acids Research*, **47**, D309–D314.

Hunter,S. *et al.* (2009) InterPro: The integrative protein signature database. *Nucleic Acids Research*, **37**, D211–D215.

Huson,D.H. *et al.* (2007) MEGAN analysis of metagenomic data. *Genome Research*, **17**, 377–386.

Jagtap,P.D. *et al.* (2015) Metaproteomic analysis using the Galaxy framework. *PROTEOMICS*, **15**, 3553–3565.

Jansson,J.K. and Baker,E.S. (2016) A multi-omic future for microbiome studies. *Nature Microbiology*, **1**, 1–3.

Kanehisa,M. and Goto,S. (2000) KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, **28**, 27–30.

Käll,L. *et al.* (2007) Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature Methods*, **4**, 923–925.

Klopfenstein,D.V. *et al.* (2018) GOATOOLS: A Python library for Gene Ontology analyses. *Scientific Reports*, **8**, 10872.

Kocher,P. *et al.* (2019) Spectre attacks: Exploiting speculative execution. In, *40th IEEE symposium on security and privacy (S&P'19)*.

Kucherov,G. *et al.* (2014) Approximate String Matching Using a Bidirectional Index. In, Kulikov,A.S. *et al.* (eds), *Combinatorial Pattern Matching*, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 222–231.

Lam,T.W. *et al.* High Throughput Short Read Alignment via Bi-directional BWT.

Langella,O. *et al.* (2017) X!TandemPipeline: A Tool to Manage Sequence Redundancy for Protein Inference and Phosphosite Identification. *Journal of Proteome Research*, **16**, 494–503.

Li,J. *et al.* (2014) An integrated catalog of reference genes in the human gut microbiome. *Nature Biotechnology*, **32**, 834–841.

Lipp,M. *et al.* (2018) Meltdown: Reading kernel memory from user space. In, *27th USENIX security symposium (USENIX security 18)*.

Lohmann,P. *et al.* (2020) Function is what counts: How microbial community complexity affects species, proteome and pathway coverage in metaproteomics. *Expert Review of Proteomics*, **17**, 163–173.

Louca,S. *et al.* (2016) Decoupling function and taxonomy in the global ocean microbiome. *Science*, **353**, 1272–1277.

Martens,L. and Hermjakob,H. (2007) Proteomics data validation: Why all must provide data. *Molecular BioSystems*, **3**, 518–522.

Merkel,D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. 5.

Mesuere,B. *et al.* (2018) High-throughput metaproteomics data analysis with Unipept: A tutorial. *Journal of Proteomics*, **171**, 11–22.

Mesuere,B. *et al.* (2012) Unipept: Tryptic Peptide-Based Biodiversity Analysis of Metaproteome Samples. *Journal of Proteome Research*, **11**, 5773–5780.

Mesuere,B. *et al.* (2016) Unipept web services for metaproteomics analysis. *Bioinformatics*, **32**, 1746–1748.

Muth,T. *et al.* (2018) MPA Portable: A Stand-Alone Software Package for Analyzing Metaproteome Samples on the Go. *Analytical Chemistry*, **90**, 685–689.

Muth,T. *et al.* (2015) The MetaProteomeAnalyzer: A Powerful Open-Source Software Suite for Metaproteomics Data Analysis and Interpretation. *Journal of Proteome Research*, **14**, 1557–1565.

Nesvizhskii,A.I. and Aebersold,R. (2005) Interpretation of Shotgun Proteomic Data. *Molecular & Cellular Proteomics*, **4**, 1419–1440.

Ondov,B.D. *et al.* (2011) Interactive metagenomic visualization in a Web browser. *BMC Bioinformatics*, **12**, 385.

Orsburn,B.C. (2021) Proteome DiscovererA Community Enhanced Data Processing Suite for Protein Informatics. *Proteomes*, **9**, 15.

Palomba,A. *et al.* (2021) Time-restricted feeding induces Lactobacillus- and Akkermansia-specific functional changes in the rat fecal microbiota. *NPJ biofilms and microbiomes*, **7**, 85.

Perez-Riverol,Y. *et al.* (2019) The PRIDE database and related tools and resources in 2019: Improving support for quantification data. *Nucleic Acids Research*, **47**, D442–D450.

Pockrandt,C.M. (2019) Approximate String Matching: Improving Data Structures and Algorithms.

Quast,C. *et al.* (2013) The SILVA ribosomal RNA gene database project:

Improved data processing and web-based tools. *Nucleic Acids Research*, **41**, D590–D596.

Ram,R.J. *et al.* (2005) Community Proteomics of a Natural Microbial Biofilm. *Science*, **308**, 1915–1920.

Rechenberger,J. *et al.* (2019) Challenges in Clinical Metaproteomics Highlighted by the Analysis of Acute Leukemia Patients with Gut Colonization by Multidrug-Resistant Enterobacteriaceae. *Proteomes*, **7**, 2.

Riffle,M. *et al.* (2018) MetaGOmics: A Web-Based Tool for Peptide-Centric Functional and Taxonomic Analysis of Metaproteomics Data. *Proteomes*, **6**, 2.

Robinson,M.D. *et al.* (2010) edgeR: A Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, **26**, 139–140.

Rodríguez-Valera,F. (2004) Environmental genomics, the big picture? *FEMS Microbiology Letters*, **231**, 153–158.

Rudney,J.D. *et al.* (2015) Protein relative abundance patterns associated with sucrose-induced dysbiosis are conserved across taxonomically diverse oral microcosm biofilm models of dental caries. *Microbiome*, **3**, 69.

Sajulga,R. *et al.* (2020) Survey of metaproteomics software tools for functional microbiome analysis. *PLOS ONE*, **15**, e0241503.

Sakai,R. *et al.* (2014) Dendsort: Modular leaf ordering methods for dendrogram representations in R. *F1000Research*, **3**, 177.

Sansone,S.-A. *et al.* (2012) Toward interoperable bioscience data. *Nature Genetics*, **44**, 121–126.

Schallert,K. *et al.* (2022) Pout2Prot: An Efficient Tool to Create Protein (Sub)groups from Percolator Output Files. *Journal of Proteome Research*,

21, 1175–1180.

Schäpe,S.S. *et al.* (2019) The Simplified Human Intestinal Microbiota (SIHUMIx) Shows High Structural and Functional Resistance against Changing Transit Times in In Vitro Bioreactors. *Microorganisms*, **7**, 641.

Schiebenhoefer,H. *et al.* (2020) A complete and flexible workflow for metaproteomics data analysis based on MetaProteomeAnalyzer and Prophane. *Nature Protocols*, **15**, 3212–3239.

Schiebenhoefer,H. *et al.* (2019) Challenges and promise at the interface of metaproteomics and genomics: An overview of recent progress in metaproteogenomic data analysis. *Expert Review of Proteomics*, **16**, 375–390.

Schlicker,A. *et al.* (2006) A new measure for functional similarity of gene products based on Gene Ontology. *BMC Bioinformatics*, **7**, 302.

Schneider,T. *et al.* (2011) Structure and function of the symbiosis partners of the lung lichen (Lobaria pulmonaria L. Hoffm.) analyzed by metaproteomics. *PROTEOMICS*, **11**, 2752–2756.

Schoch,C.L. *et al.* (2020) NCBI Taxonomy: A comprehensive update on curation, resources and tools. *Database*, **2020**, baaa062.

Sokal,R.R. and Michener,C.D. (1958) A Statistical Method for Evaluating Systematic Relationships University of Kansas.

Tanca,A. *et al.* (2022) Metaproteomic Profile of the Colonic Luminal Microbiota From Patients With Colon Cancer. *Frontiers in Microbiology*, **13**.

Tanca,A. *et al.* (2016) The impact of sequence database choice on metaproteomic results in gut microbiota studies. *Microbiome*, **4**, 51.

The Gene Ontology Consortium (2019) The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Research*, **47**, D330–D338.

The,M. and Käll,L. (2019) Integrated Identification and Quantification Error Probabilities for Shotgun Proteomics * [S]. *Molecular & Cellular Proteomics*, **18**, 561–570.

The UniProt Consortium (2019) UniProt: A worldwide hub of protein knowledge. *Nucleic Acids Research*, **47**, D506–D515.

The UniProt Consortium (2021) UniProt: The universal protein knowledgebase in 2021. *Nucleic Acids Research*, **49**, D480–D489.

Uszkoreit,J. *et al.* (2015) PIA: An Intuitive Protein Inference Engine with a Web-Based User Interface. *Journal of Proteome Research*, **14**, 2988–2997.

Van den Berge,K. *et al.* (2017) stageR: A general stage-wise method for controlling the gene-level false discovery rate in differential expression and differential transcript usage. *Genome Biology*, **18**, 151.

Van Den Bossche,T. *et al.* (2020) Connecting MetaProteomeAnalyzer and PeptideShaker to Unipept for Seamless End-to-End Metaproteomics Data Analysis. *Journal of Proteome Research*, **19**, 3562–3566.

Van Den Bossche,T., Kunath,B.J., *et al.* (2021) Critical Assessment of MetaProteome Investigation (CAMPI): A multi-laboratory comparison of established workflows. *Nature Communications*, **12**, 7305.

Van Den Bossche,T., Arntzen,M.Ø., *et al.* (2021) The Metaproteomics Initiative: A coordinated approach for propelling the functional characterization of microbiomes. *Microbiome*, **9**, 243.

Van der Jeugt,F. *et al.* (2022) UMGAP: The Unipept MetaGenomics Analysis Pipeline. *BMC Genomics*, **23**, 433.

Verschaffelt,P., Van Den Bossche,T., Gabriel,W., *et al.* (2021) MegaGO: A Fast Yet Powerful Approach to Assess Functional Gene Ontology Similarity across Meta-Omics Data Sets. *Journal of Proteome Research*, **20**, 2083–2088.

Verschaffelt,P. *et al.* (2020) Unipept CLI 2.0: Adding support for visualiza-

tions and functional annotations. *Bioinformatics*, **36**, 4220–4221.

Verschaffelt,P. *et al.* (2023) Unipept Desktop 2.0: Construction of targeted reference protein databases for proteogenomics analyses. 2023.02.09.527820.

Verschaffelt,P., Van Den Bossche,T., Martens,L., *et al.* (2021) Unipept Desktop: A Faster, More Powerful Metaproteomics Results Analysis Tool. *Journal of Proteome Research*, **20**, 2005–2009.

Verschaffelt,P. *et al.* (2022) Unipept Visualizations: An interactive visualization library for biological data. *Bioinformatics*, **38**, 562–563.

Vroland,C. *et al.* (2016) Approximate search of short patterns with high error rates using the 01*0 lossless seeds. *Journal of Discrete Algorithms*, **37**, 3–16.

Waardenberg,A.J. *et al.* (2015) CompGO: An R package for comparing and visualizing Gene Ontology enrichment differences between DNA binding experiments. *BMC Bioinformatics*, **16**, 275.

Webb,E.C. (1992) Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes. *Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes.*

Wilmes,P. *et al.* (2015) A decade of metaproteomics: Where we stand and what the future holds. *PROTEOMICS*, **15**, 3409–3417.

Wilmes,P. and Bond,P.L. (2004) The application of two-dimensional polyacrylamide gel electrophoresis and downstream analyses to a mixed community of prokaryotic microorganisms. *Environmental Microbiology*, **6**, 911–920.

Yates,J.R. (2019) Recent technical advances in proteomics. *F1000Research*,

8, F1000 Faculty Rev–351.

Zhang,X. and Figeys,D. (2019) Perspective and Guidelines for Metaproteomics in Microbiome Studies. *Journal of Proteome Research*, **18**, 2370–2380.