

École polytechnique de Louvain

Clustering and analyzing large mass spectrometry-based proteomics data

Author: **Kilian MAES**

Supervisors: **Pierre DUPONT, Laurent GATTO**

Readers: **Vincent BRANDERS, Xavier GILLARD**

Academic year 2020–2021

Master [120] in Computer Science and Engineering

Acknowledgments

This master thesis was made possible thanks to multiple people to whom I would like to express my gratitude.

I want to thank Professor Laurent Gatto for his availability, his benevolence, and his close support all year long. With his expertise in biology, he helped me understand the principles I needed to carry out this thesis. The inclusion in his research team allowed me to keep social contact during this pandemic.

I would also like to thank Professor Pierre Dupont. His rigor and his high requirement pushed me further to improve this master thesis. At the same time, his feedbacks on my algorithmic analysis of Falcon allowed me to propose a clear description of the algorithms.

Then, I would like to thank Wout Bittremieux, the author of the software discussed throughout this document, for his valuable feedback and constructive discussions. This work would not have been possible if he had not released his code as open-source software.

Last but not least, I would like to thank Matthieu Meert for his proofreading and the discussions we had when I was hesitating between several ways of proceeding.

Abstract

Shotgun proteomics is a method widely used to identify proteins present in biological samples. This is essential to understand how cells work, the mechanisms of certain diseases, etc. During shotgun proteomics experiments, a large number of spectra are generated. Their analysis, typically using database search, identifies many proteins present in the samples. However, a large proportion of these spectra remain unidentified. Previous works have shown that clustering the spectra can help to identify new proteins. However, the increasing amount of data requires efficient clustering algorithms. This work focuses on Falcon, a clustering software recently released. First, we analyze its algorithms, and we study its scalability. Then, we use it to cluster a large dataset, and we examine whether useful information can be extracted from the clusters. Finally, we discuss several areas of improvement for Falcon.

Contents

1	Introduction	7
1.1	Proteomics	7
1.2	Motivation for Clustering	11
1.3	A survey	12
1.4	Discussion	15
1.5	Objectives	15
2	Data set	17
2.1	The Draft of the Human Proteome	17
2.2	The spectra	17
3	Falcon : Algorithmic description	22
3.1	Problem statement and constraints	22
3.2	Algorithm	23
3.3	Implementation	33
3.4	Time complexity of FALCON	36
3.5	Why does FALCON seem to be a good choice?	40
4	Results	41
4.1	Parameters selection	41
4.2	Clusters overview	44
4.3	Unidentified clusters reanalysis	46
5	Areas for improvement	56
5.1	Dimension reduction	56
5.2	DBSCAN concurents	57
5.3	Avoiding the postprocessing step	64
6	Conclusion	66
	Appendices	67
A	Reproducibility	68
A.1	Setup	68
A.2	Draft of the Human Proteome	68
A.3	Source code	68
A.4	Parameters used for the clustering	69

A.5 Libraries	70
-------------------------	----

Notation	Description	Order of magnitude
S	The input space of spectra	$ S \approx 20 * 10^6$
$s = \{P_1, P_2, \dots\}$	A spectrum	-
$P_i = (m/z_i, I_i)$	A peak of intensity I at position m/z	-
X	The space of vectors corresponding to spectra	$ X = S $
$v(s) : S \rightarrow X^d$	The vectorization function	$d \approx 30000$
$\mathbf{x}$	A vector belonging to X	-
x_i	The component i of vector $\mathbf{x}$	-
$\phi(\mathbf{x}) : X \rightarrow X'$	The dimensionality reduction function	-
X'	The space of spectra of reduced dimension	$ X' = S $
$\mathbf{x}'$	A vector of reduced dimension, $\mathbf{x}' = \phi(\mathbf{x})$	-
$d'(\mathbf{x}')$	The number of components of $\mathbf{x}'$	$d' = 800$
$h(j)$	The non-cryptographic function used for the dimensionality reduction	-
$p_z(s)$	The precursor charge for spectrum s	$p_z \in \{1, 2 \dots 7\}$
$p_{m/z}(s)$	The precursor m/z ratio for spectrum s	$p_{m/z} \in [200, 1500]$
$p_m(s)$	The precursor mass for spectrum s	$p_m = p_z * p_{m/z}$
$B_{z,[a,b[} = B$	A bucket containing the spectra $\{s \in S p_z(s) = z, p_m(s) \in [a, b[\}$	$ B = 20000$
$\mathbf{B}$	A set of buckets	-
n	The number of spectra (all charges)	$n \approx 20 * 10^6$
n_z	The number of spectra s s.t. $p_z(s) = z$	$n_{z=2} \approx 10 * 10^6$
n_B	The number of spectra in a bucket B	$1 \leq n_B \leq 25000$
tol^{p_m}	The tolerance for the mass constraint	0.02 Da
$\text{dist}(\mathbf{x}_1, \mathbf{x}_2)$	A distance function	-
$\theta(\mathbf{x}_1, \mathbf{x}_2)$	The angle separating $\mathbf{x}_1$ and $\mathbf{x}_2$	$\theta \in [0, 180^\circ]$
$\text{dist}_\theta(\mathbf{x}_1, \mathbf{x}_2)$	The cosine distance ($= 1 - \cos(\theta(\mathbf{x}_1, \mathbf{x}_2))$)	$\text{dist}_\theta \in [0, 2]$
$D^{n_z \times n_z}$	The distance matrix for spectra of charge z	-
k	The number of centroids used by k-means	-
$\mathbf{C}$	A set of clusters of spectra	-
C_i	The cluster i , $C_i \in \mathbf{C}$	-
$h(\mathbf{C})$	The homogeneity metric	$h \in [0, 1]$
$c(\mathbf{C})$	The completeness metric	$c \in [0, 1]$

Table 1: Common notations used in this document.

Chapter 1

Introduction

This work aims at clustering data obtained during shotgun proteomics experiments. By detecting similar elements in a dataset (which is the goal of clustering), valuable results can be produced. In this chapter, we first introduce the utility of shotgun proteomics. Then, we present a survey of previous works aiming at clustering mass spectrometry data. Finally, we present the objectives pursued in this work.

1.1 Proteomics

Proteomics is the science that studies proteins, large molecules that are essential for the cells. Proteins participate in the metabolism; they ensure the tissues structure, are involved in the immune system, etc. In the human organism, the nucleus of each cell contains several DNA molecules carrying the entire genome, which contains the information to produce all the proteins needed by the cells. In practice, a cell only needs a subset of these proteins, among others depending on its type: a nerve cell will not express the same set of proteins as an endothelial cell.

1.1.1 Proteomics vs Genomics

The complete sequence of the **human genome** is known since the early 2000s, thanks to the efforts of the Human Genome Project [1]. It consists of a 2.91-billion base pairs consensus sequence of the DNA contained in the human cells (each cell contains the entire genome). The knowledge of the human genome allowed the scientists to intensively study and map the genes (subsequences of the genome).

The knowledge of the human genome is of paramount importance for biologists because the DNA contains the information needed to synthesize all the proteins that could potentially be expressed. However, not all the protein-coding genes are expressed at the same time, in the same cells, even though all the cells contain the whole genome. The **proteome** of a cell is the set of proteins expressed by the cell at a given time. Contrary to the genome, the proteome is dynamic and can vary from one cell to another, depending notably on the tissue.

Moreover, only studying the genome would be very simplistic because several protein **isoforms** (variants) can be produced from most genes, by mechanisms including alternative splicing (different messenger RNA's can be produced from the

same gene) and posttranslational modifications (PTMs) of the protein. The human genome contains around 20 000 genes, but many more proteins can be produced (there is currently no precise estimation of the human proteome size, which could be comprised between several tens of thousand and several billion of protein isoforms [2]).

1.1.2 Shotgun Proteomics

A shotgun proteomics experiment is one of the methods widely used to study the proteome (see Figure 1.1). It aims at identifying and quantifying the proteins present in a cell lysate, which is a solution containing the proteins of interest, obtained by lysing the cell membranes. The cell lysate, which can contain thousands of proteins, undergoes several stages [3]. First, a particular protein is used to digest (cut) the proteins present in the mixture in several smaller fragments, referred to as **peptides**. Secondly, the mixture is processed in a chromatography column, which separates the peptides according to the charge. Thirdly, the peptides are ionized before entering in a mass spectrometer, which will select some of these ionized peptides, based on their ratio mass over charge (m/z), to access to the second round of selection that will generate a **spectrum** corresponding to the peptide [4]. The ionized peptide from which the spectrum is generated is called the **precursor**.

Spectra

At the end of the experiment, the researcher gets the spectra and the associated metadata such as the m/z ratio of the precursor ($p_{m/z}$) and its charge (p_z), the retention time (the time needed by the peptide to pass through the chromatography column), etc. A spectrum consists in a set of pairs ($m/z, I$), called the peaks (an example is shown in Figure 1.2). The m/z value defines the peak position, and I is the peak intensity (visually, the peak height). The distance separating the peaks (their difference of m/z) identifies the amino acids¹ and thus the peptide sequence. In other words, a spectrum is a fingerprint of a peptide.

When enough peptides generated from the same protein are identified, it is usually possible to identify the corresponding protein. However, when identifying several peptides fails, a protein can remain unidentified, which can be problematic for understanding many biological processes.

Identification

Researchers use two main methods to identify a peptide sequence based on its spectrum [3]. On the one hand, *de novo* sequencing aims at "manually" retrieving the amino acid masses based on the distance of successive peaks. On the other hand, the **database search** is a more automated way that retrieves the sequence by matching the spectrum to the theoretical spectra of known sequences (i.e., the software predicts what the spectrum of a given sequence will look like).

¹The amino acids are the building blocks of peptides and proteins.

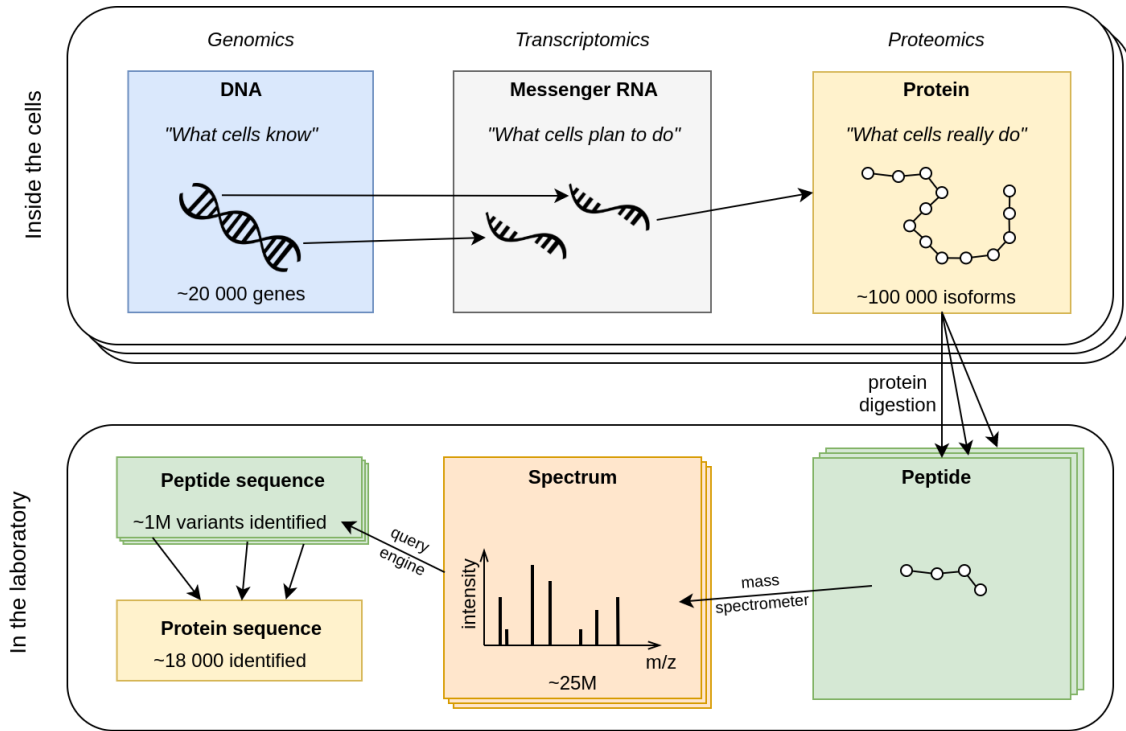


Figure 1.1: Each human cell contains the same genome (carried by DNA molecules). In each cell, a subset of the genome is transcribed in messenger RNA, which is translated into proteins. This subset varies depending notably on the cell type. Proteomics aims at studying which proteins are expressed in the cells. During shotgun proteomics experiments, the proteins are extracted from the cells, digested, and processed by a mass spectrometer which generates spectra. Query engines can identify, based on the spectra, the amino acid sequences of many peptides and the corresponding proteins. The values indicated in the lower rectangle are associated to the Draft of the Human Proteome, the dataset presented in Chapter 2: it contains 25M spectra. Performing a database search on these spectra made possible to identify around 1M peptides variants (including peptides with posttranslational modifications), originating from 18 000 human proteins.

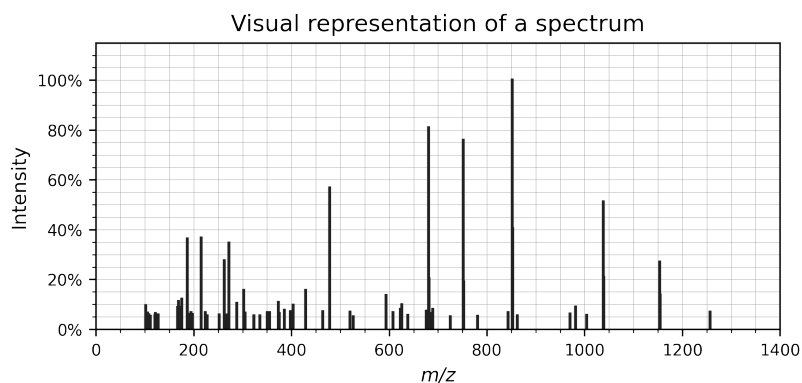


Figure 1.2: Visual representation of the peaks $(m/z, I)$ of a spectrum s . The metadata associated with this spectrum includes the precursor charge $p_z(s) = +2$, and the precursor m/z ($p_{m/z}(s) = 626.7679$). The query engine identified the peptide sequence corresponding to this spectrum as 'VDGETASDSESR', where each letter corresponds to one of the 20 amino acids (V = valine, D = aspartic acid, etc.).

Limitations

Ideally, each spectrum would correspond to a peptide, and it would be possible to identify its sequence (label) without ambiguity. In practice, this is not the case. A large proportion of spectra (typically 75% according to Griss et al. [5]) cannot be identified with high confidence when analyzing spectra using database search. Some of these spectra are of poor quality, making it difficult to distinguish the signal peaks from the noise [6]. Others are high-quality spectra corresponding to peptides having PTMs (Post Translational Modifications) or amino acids variants not contained in the database used for the search [7].

PostTranslational Modifications

PTMs are changes made to the proteins after they were synthesized. They can alter the protein function (e.g., activate/deactivate a protein and the downstream signaling cascades). PTMs can be detected by mass-spectrometry, because the spectrum of a peptide is modified depending on the modifications the peptide undergoes (i.e., the spectrum acquired from the peptide with a particular PTM is not the same as the spectrum acquired from the same peptide without the PTM). However, when using database search to identify spectra, it is impracticable to search for all the existing types of PTMs (e.g., phosphorylation, acetylation) on a large spectra dataset: given that a peptide could have several PTMs, and a theoretical spectrum has to be generated during the search for each combination, the search space would increase exponentially in the number of accounted PTMs. Thus, researchers usually only specify the PTMs they expect to find in their samples.

1.2 Motivation for Clustering

Since 2000, a dozen studies (see next section) have focused on the clustering of data obtained from tandem mass spectrometry. These studies have shown several purposes for clustering, including:

1. Speeding up the database search for identification: when several spectra are identical, only one of them needs to be submitted for identification;
2. Increasing the number of identified peptides and proteins.
 - In some cases, it is possible to combine several low-quality spectra in one spectrum with an improved signal-to-noise ratio, which can lead to the identification of the spectrum;
 - Clusters containing very similar spectra (see Figure 1.3) tend to indicate that the spectra correspond to the same peptide. In some cases, no spectrum in the cluster could be identified with the "basic" database search, but applying more advanced methods makes it possible to identify the spectrum sequences.
3. Controlling the quality of identifications. The determination of the sequence is based on a confidence level and can be wrong in some cases. Very similar spectra should logically be associated to the same sequence, since the identification is supposed to be deterministic.

The next section focuses on previous work related to the clustering of tandem mass spectrometry data.

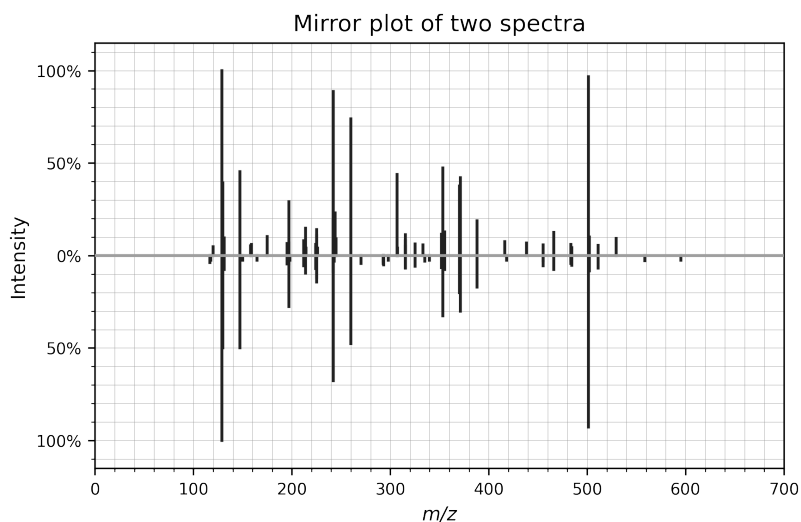


Figure 1.3: Mirror plot of two very similar spectra: they share most of their peaks and the peak intensities tend to be proportional. These spectra are likely to correspond to peptides having the same sequence.

1.3 A survey

Since 2003, many different clustering software and algorithms have been released. Most of them are briefly presented in this section, as well as interesting results obtained by the authors when applying them on proteomics datasets of spectra.

1.3.1 NoDupe

In 2003, Tabb et al. [8] observe that "duplicated" spectra are acquired during proteomic experiments, even though the control software of the spectrometers is programmed to avoid capturing several times the same peptide. Given that the time needed to identify a peptide is a bottleneck during the processing of the acquired spectra, they want to remove duplicated spectra to save time.

To measure the similarity between two spectra, they use the **cosine similarity**, which measures the proportionality of the spectra: the spectra are represented as vectors in a space with as many dimensions as m/z values corresponding to peaks in the two spectra. The component for each dimension corresponds to the peak intensity I . Two spectra are very similar if the angle formed by the corresponding vectors is small, and vice-versa. For each spectrum, they count how many other spectra are similar, and they remove the "duplicated spectra".

Tabb et al. conclude that up to 28% of the spectra are duplicated (highly similar to at least one other spectrum). By removing the duplicated spectra, they can speed up the identification process (fewer spectra have to be submitted to the query engine) but at the cost of losing some peptide identifications (around 4%). For further work, they propose to combine the spectra of the same cluster to improve the accuracy of the identification of the peptides.

1.3.2 Pep-Miner

In 2004, Beer et al. [9] propose a slightly different approach. As in the previous study, they use the cosine similarity to assess the similarity between two spectra, but their approach differs in two ways. First, they use a more sophisticated clustering algorithm, designed by Ben-Dor et al. [10], which represents the clustered data as a graph to identify the cliques and form the clusters. Secondly, they do not simply remove the duplicated spectra. Instead, they build a **consensus spectrum** based on all the spectra belonging to a cluster by summing and averaging them.

Beer et al. concluded that clustering could improve the identification of the peptides. By building a consensus spectrum from spectra belonging to the same cluster, the signal-to-noise ratio can be improved, and the score (related to the statistical significance) of the identification can be increased.

1.3.3 MS2Grouper

In 2005, Tabb et al. [11] use the same approach as in the Pep-Miner algorithm: they use the cosine similarity and build a consensus spectrum for each cluster. The main difference in their approach is that they use a different clustering algorithm: they build a graph where the vertices represent spectra, and the edges represent a

high similarity. Using the Bron-Kerbosch algorithm [12], they identify the maximal cliques of the graph, which are used to form the clusters. Finally, the clusters are extended by adding the highly similar spectra to all but one spectrum belonging to a cluster.

By replacing the spectra belonging to the same cluster by a consensus spectrum, they reduce by up to 20% the number of identifications to perform. The total number of peptides identified is lower when MS2Grouper is used (compared to individual identification of each spectrum), but some peptides are only identified when the clustering is performed.

1.3.4 PepMerger

In 2007, Flikka et al. [13] release a software able to cluster spectra. It proposes several similarity measures, clustering options, etc. When testing their software on experimental data, their results show that more peptides are identified without clustering, but as in the previous algorithm (MS2Grouper), many peptides are only identified when the clustering is performed to build a consensus spectrum for each cluster.

1.3.5 MS-Clustering

In 2008, another software is released by Frank et al. [14]. The cosine similarity is used to measure the likeness of two spectra, and an incremental hierarchical algorithm clusters the spectra. Some heuristics are used to reduce the number of comparisons to perform: for example, two spectra are compared only if they share at least one of their five strongest peaks, which avoids around 95% of the spectra comparisons.

Frank et al. conclude that the benefit of their algorithm for peptide identification depends on the size of the dataset. For large datasets (> 10 million spectra), the clustering can lead to a tenfold reduction of the number of spectra to submit to the query engine for identification, and it tends to improve the number of peptides and proteins identification. For smaller datasets, clustering can lead to the loss of some peptides identifications.

1.3.6 CAMS-RS

In 2012, Saeed et al. [15] [16] propose a clustering algorithm using a novel similarity metric. Based on the fact that spectra generated from the same peptide tend to have peaks at the same m/z values, they decompose each spectrum into sets of consecutive peaks. Then, they compare all the subsets from the first spectrum to all the subsets of the second spectrum. The scoring function is increased each time the peaks of the two subsets being compared are at the same position (the same m/z). In short, instead of simultaneously comparing all the peaks of the two spectra, CAMS-RS compares subsets of the two spectra.

1.3.7 PRIDE Cluster

In 2013, Griss et al. [17] use a clustering algorithm based on MS-Clustering [14] to cluster the large amount of spectra (several tens of million) available in the PRIDE Archive [18]. They create a new resource publicly available, PRIDE Cluster², that contains the result of the clustering.

1.3.8 MaRaCluster

In 2016, The et al. [19] develop MaRaCluster, a new clustering software using a fragment rarity metric to assess the similarity of the spectra. First, the background distribution is calculated, i.e., the probability of having a peak at a given m/z value in a spectrum. The similarity is evaluated based on the fact that spectra sharing peaks at an m/z value with low frequency are likely to correspond to the same peptide. Hierarchical clustering with complete linkage is used to group similar spectra.

1.3.9 Locality Sensitive Hashing

In 2006, Dutta et al. [20] propose to apply the principle of the LSH, Locality Sensitive Hashing, to speed up the MS/MS database search. The idea is to compute a hash of the spectra to compare. In theory, similar spectra should have the same hash. Therefore, all spectra having different hashes do not need to be compared, which results in better performances. They propose to use the LSH for clustering.

In 2018, Wang et al. [21] develop msCRUSH, a clustering algorithm using the LSH to reduce the number of comparisons between spectra. They conclude that using their algorithm results in a 10-fold decrease of the time needed to perform the clustering, compared to the PRIDE cluster algorithm [17]. Moreover, they notice that the number of peptide identifications is increased by a few percent (1% to 3%).

1.3.10 Spectra-cluster

The PRIDE Cluster algorithm [17] briefly presented earlier was used in 2013 by Griss et al. to cluster all the spectra available in the PRIDE Archive and make the result available in a new resource, PRIDE Cluster. Because new spectra are added to the PRIDE Archive continuously, the clustering must be redone periodically to keep the resource up-to-date.

Faced with the fact that the amount of data tends to grow exponentially, Griss et al. decide to design an improved version of the PRIDE Cluster algorithm, which can be executed on a cluster of computers (to decrease the time needed for the clustering). In 2016, Griss et al. use their improved version of PRIDE Cluster, called Spectra-cluster, to cluster all the spectra present in the PRIDE Archive (256 million of spectra at the moment of the study) [5] [22]. Around 75% of these spectra are unidentified. This very large quantity of spectra was clustered in parallel on a cluster of computers..

The new Spectra-cluster algorithm of Griss et al. has two main differences compared to its ancestor. First, the distance between spectra is not anymore

²<https://www.ebi.ac.uk/pride/cluster/>

computed with the cosine similarity. Instead, a probabilistic scoring function evaluates the chances that the number of matched peaks in the two spectra occurs randomly. Secondly, the algorithm is adapted for the MapReduce programming model, which is well suited for cloud computing: the clustering of the PRIDE Archive spectra described in the paper [5] is performed on a cluster of 30 nodes with the Apache Hadoop framework.

The motivations of Griss et al. for the clustering are the following:

- Identify wrongly annotated spectra. When a cluster contains spectra with different identifications, some are wrongly clustered or wrongly annotated.
- Infer the annotation of an unidentified spectrum from the other members of the cluster.
- Analyze clusters containing only unidentified spectra.

By paying particular attention to the unidentified clusters, the authors identified 20% of the spectra belonging to unidentified clusters. To achieve that, they use several methods: *de novo* identification, open modification searches³, etc. Applying these methods to the entire set of spectra would have been too costly and would have resulted in many false positives. However, by isolating some consistent unidentified clusters, it becomes possible to make further analysis.

1.4 Discussion

All the works presented in the previous section have shown benefits to cluster data obtained during shotgun proteomics experiments, either by reducing the time required for the identification, by identifying new peptides thanks to consensus spectra or by making possible to perform further analysis on clusters of interest. Unfortunately, little attention has been paid to the analysis of the clustering algorithms (e.g., their time complexity).

The ever-increasing amount of data requires the development of high-performance algorithms with good time complexities. At the same time, it remains obviously essential to produce a quality clustering, which tends to group the spectra corresponding to the same peptides in the same clusters (otherwise, clustering would not make it possible to obtain useful results).

1.5 Objectives

In this work, we analyze FALCON [23], a clustering software whose authors claim it presents good scalability, better time performance than many of its predecessors while producing larger (but very pure) clusters.

In Chapter 2, we present the dataset we aim to cluster, the Draft of the Human Proteome. We see that its large size justifies the need to use efficient algorithms. In Chapter 3, we analyze the algorithms used in FALCON. This allows us to determine

³A database search with parameters such that PTMs can be detected.

its time complexity. In Chapter 4, we apply FALCON to the Draft of the Human Proteome dataset, and we analyze the results. In particular, we can identify the peptide sequences corresponding to clusters containing previously unidentified spectra. Finally, in Chapter 5, we explore several areas of improvement for FALCON.

Chapter 2

Data set

This work aims at clustering a dataset obtained during a large-scale shotgun proteomics experiment. In this chapter, we briefly present this dataset, the Draft of the Human Proteome¹. The large size of the Draft of the Human Proteome notably justifies the use of an efficient clustering algorithm, which is presented in Chapter 3.

2.1 The Draft of the Human Proteome

In 2014, a "draft" of the human proteome obtained by two research groups is published [24] [25]: they took samples from several individuals, and they used mass spectrometry (according to the method briefly described in the introduction) to acquire the spectra. As shown in Table 2.1, the samples were taken either from adults or fetuses. Except for samples from blood tissue, all samples were taken during an autopsy on several individuals. Many tissues are represented (17 adult tissues and 7 fetal tissues, shown in Table 2.1): the esophagus, retina, heart, etc. Moreover, 6 samples are composed of primary hematopoietic cells (monocytes, B cells...) obtained by routine plateletpheresis (extracted from the blood of several individuals). All donors were presumed healthy (they had no known disease), and the solid tissues were histologically normal. By covering a wide range of human tissues, Kim et al. and Wilhelm et al. drew a (non-exhaustive) map of the human proteome: by submitting the spectra they acquired to query engines, it becomes possible to identify more than 440 000 peptides, which can be remapped to more than 18 000 proteins expressed in human cells.

2.2 The spectra

Kim et al. [24] released the spectra and their metadata openly, including the identifications (the "labels", corresponding to the peptide sequence associated with a spectrum in the case it could be determined).

¹This is the same dataset used by the Bittremieux et al. [23] to benchmark their clustering tool, which is presented in Chapter 3. We also used the same identification data (i.e., the spectra labels).

2.2.1 Metadata

The data set contains nearly 25 million spectra. Each spectrum is associated with the following metadata: the precursor charge p_z , the precursor m/z $p_{m/z}$, the label (or NA if the peptide sequence could not be identified with high confidence), the mass spectrometer used to acquire the spectra, and the sample it comes from. All the spectra from the same sample share common metadata:

- The tissue the sample comes from (e.g., liver, heart);
- The stage of the development of the individual the sample comes from (adult or fetal).

2.2.2 Precursor charge

Figure 2.1 shows the distribution of precursor charges among the spectra. The majority of the spectra have a precursor charge equal to 2. The precursor charge comprises between 1 and 15, but the number of spectra for precursor charges larger than 7 tends to be low (less than 10 000 spectra per charge) and cannot be observed in the figure.

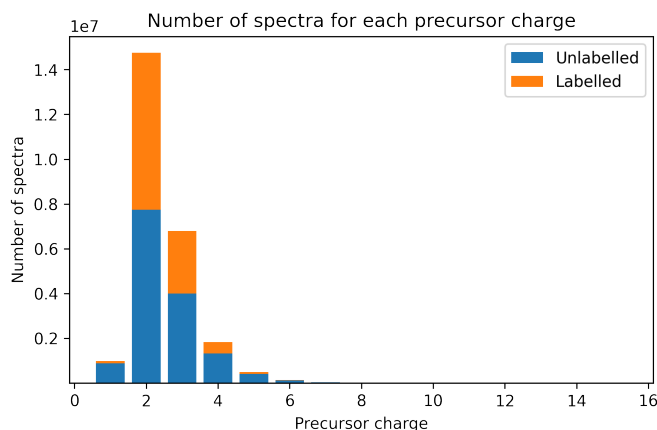


Figure 2.1: The number of spectra per precursor charge is not distributed uniformly. A label is available for less than half of the spectra.

2.2.3 Labels

We used the identification data from the MassIVE reanalysis (RMSV000000091.3) of the Draft of the Human Proteome. This reanalysis was performed with the MS-GF+ query engine [26]. In Figure 2.1, it can be observed that less than half of the spectra have labels. These labels, also called sequences or identifications, correspond to the peptide sequence in amino acids. For example, a peptide with sequence DMYQIMDEIK would consist of the following amino acids: aspartic acid (D), methionine (M), tyrosine (Y), etc. Ideally, submitting a spectrum to a search engine allows the sequence of the associated peptide to be identified (the sequence

is the label of the spectrum). When a spectrum is submitted to the query engine, it is compared to the theoretical spectrum generated from a list of known peptide sequences. When the similarity between the submitted spectrum and a theoretical spectrum is sufficient, it is a "match", and the peptide sequence used to generate the theoretical spectrum is returned: this sequence is the spectrum label.

To get reliable identifications, the authors performed the identification against a decoy database generated from random permutations of the known peptide sequences (a widely used method to ensure identification reliability). The permuted sequences do not correspond to any existing peptide while retaining the characteristics of a typical human peptide sequence. Thus, identifying a spectrum against the decoy database should not result in a match (otherwise, it is a false positive). In practice, it is impossible to avoid all the false positives. Still, it is possible to bound their number: the identification is made with respect to a FDR (False Discovery Rate) of 1%. It means that at most 1% of the spectra identified against the decoy database will result in a match. This FDR ensures reliable identifications but does not exclude the possibility to get a controlled proportion of false-positive hits².

2.2.4 Precursor m/z

For each precursor charge, it is also interesting to look at the distribution of the precursor m/z 's (Figure 2.2), since it will determine the time complexity of the clustering algorithm presented in the next chapter. Each bin of the histogram covers a 1 m/z interval. For precursor charges equal to 2, 3, and 4, the precursor m/z 's tend to follow a distribution with a "maximum" approximately between 500 and 750 m/z . For the other charges, the number of spectra is insufficient to observe the precursor m/z 's distribution clearly. However, it should be noted that the number of spectra can vary a lot from one bin to another, as is the case for precursor charge 6.

²False positive hits must be avoided as much as possible since they lead the researcher to think a peptide is present in the sample when it is not.

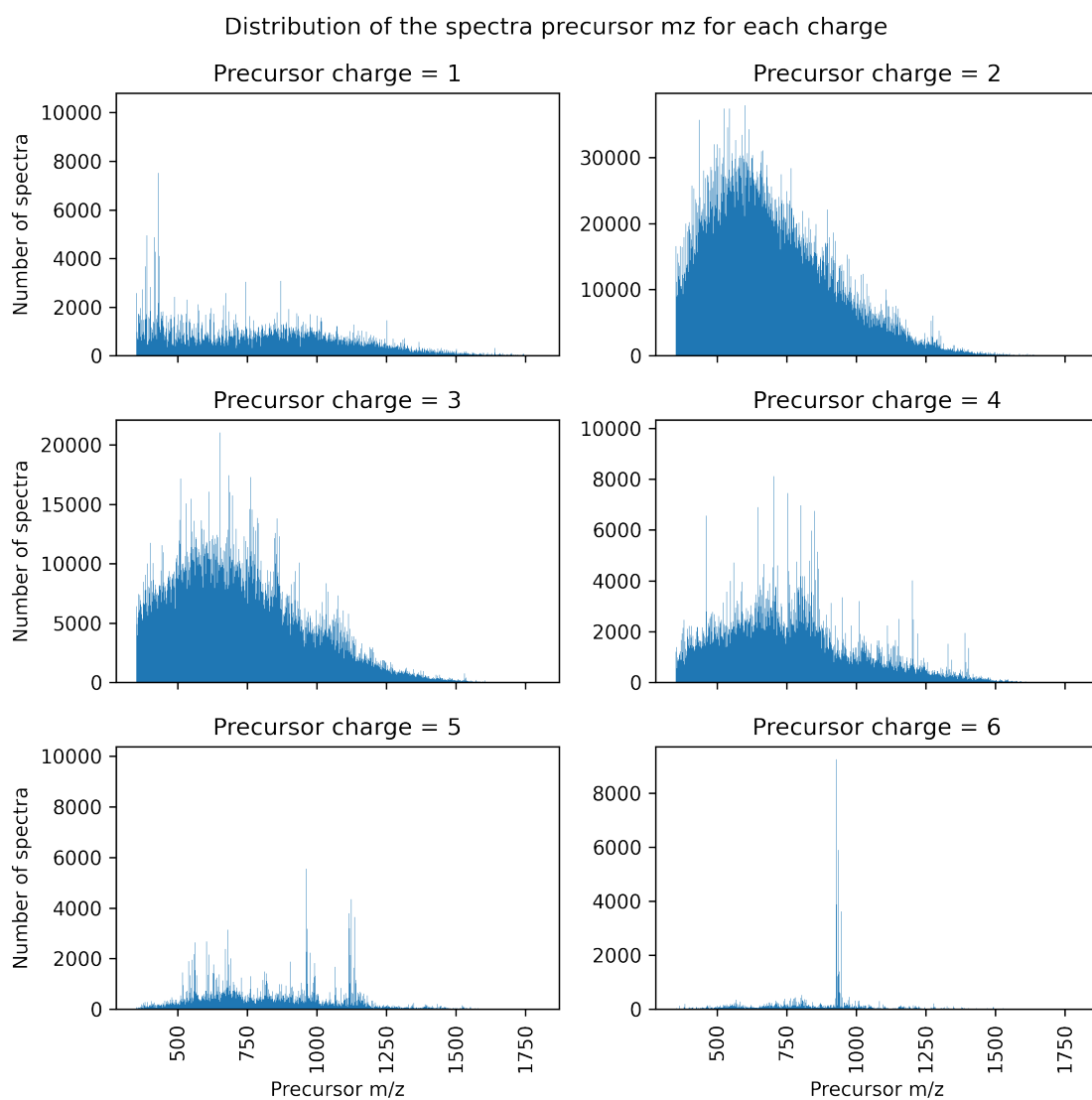


Figure 2.2: The precursor m/z 's are not uniformly distributed for each charge. Note that the y-axis is not common to all the subplots, it had to be adapted for each precursor charge subplot because the number of spectra can vary significantly depending on the precursor charge.

Stade of develop- ment	Type of tissue	Tissue/cells	# sp./tot	% sp. labelled
Adult	Blood	Bcells	3.98 %	36.30 %
		CD4 ⁺ T cells	2.82 %	48.08 %
		CD8 ⁺ T cells	4.26 %	49.19 %
		Monocytes	2.65 %	45.33 %
		NK cells	4.73 %	33.18 %
		Platelets	3.41 %	38.07 %
	Solid tissue	Adrenal gland	2.58 %	47.88 %
		Colon	2.82 %	41.76 %
		Esophagus	2.73 %	30.53 %
		Frontal cortex	4.56 %	39.43 %
		Gallbladder	3.10 %	38.59 %
		Heart	5.72 %	34.07 %
		Kidney	2.81 %	41.53 %
		Liver	4.37 %	38.72 %
		Lung	2.79 %	38.79 %
		Ovary	2.86 %	55.46 %
		Pancreas	2.05 %	45.88 %
		Prostate	3.28 %	38.07 %
		Rectum	2.74 %	43.89 %
		Retina	3.54 %	47.55 %
		Spinal cord	2.91 %	41.14 %
		Testis	2.76 %	54.57 %
		Urinary bladder	4.72 %	37.97 %
Fetal		Brain	2.31 %	42.04 %
		Gut	4.42 %	33.32 %
		Heart	4.44 %	52.25 %
		Liver	3.95 %	46.32 %
		Ovary	2.56 %	55.31 %
		Placenta	1.95 %	36.31 %
		Testis	2.19 %	45.38 %

Table 2.1: Many different samples from different tissues were used to generate the draft of the human proteome. The fourth column indicates the proportion of spectra acquired with the tissue/cells (for the total number of spectra, 25 M), and the last column indicates the proportion of spectra that could be identified (they have a label).

Chapter 3

Falcon : Algorithmic description

Clustering large datasets requires efficient algorithms. FALCON is a recently released software that uses advanced algorithmic solutions to cluster in reasonable time large datasets of spectra acquired by tandem mass spectrometry (MS/MS). It was developed by Bittremieux et al. [23]. To be efficient, it uses dimensionality reduction, fast nearest neighbors searching, and takes advantage of the background knowledge available about the use of the MS/MS for peptides identification (several constraints presented below can be used to decrease the search space). The central clustering algorithm used by FALCON is DBSCAN.

This chapter is structured as follows. In Section 3.1, we explain the constraints used in FALCON. In Section 3.2, we describe the different algorithms used by FALCON, with their pseudocodes. The actual Python implementation (which libraries are used) of FALCON is shortly presented in Section 3.3. The time complexity of FALCON is analyzed in Section 3.4. The reasons why this algorithm seems to be a good choice for our dataset are detailed in Section 3.5.

3.1 Problem statement and constraints

The authors of FALCON have designed it to solve a constrained clustering problem with two constraints:

1. Spectra with precursors of different charges should not be compared: if the same peptide is ionized and captured with different (precursor) charges, the peaks will not be at the same positions. Thus, it makes sense only to compare spectra having the same precursor charge. This constraint is the **precursor charge constraint** : if $p_z(s_1) \neq p_z(s_2)$, s_1 and s_2 cannot be in the same cluster.
2. Two spectra are likely to correspond to the same peptide only if their precursors have a very similar mass. Thus, we want to avoid grouping spectra having dissimilar precursor masses in the same clusters. This constraint is referred to as the **precursor mass constraint**. Two spectra s_1 and s_2 respect the precursor mass constraint iff

$$|p_m(s_1) - p_m(s_2)| < \text{tol}^{p_m}$$

When constraint 1 is respected, which is always the case in FALCON, then the precursor mass constraint can also be expressed with the spectra precursor m/z because

$$p_m(s) = p_z(s) * p_{m/z}(s)$$

so the precursor mass and the precursor m/z of the spectra are proportional when the precursor charge is the same.

Potential neighbors When two spectra respect these two constraints, they are said to be **potential neighbors**. It is necessary to compute the distance separating them and compare it to the ϵ threshold used in DBSCAN to determine whether they are neighbors or not. By opposition, two spectra that violate at least one of these constraints will not be considered as potential neighbors. In the distance matrix, the distance separating spectra violating the constraint is deemed to be maximal ($= 1$ with the cosine distance used by FALCON) during the clustering by DBSCAN.

Also, note that constrained clustering is often referred to as "semi-supervised clustering" [27].

3.2 Algorithm

In this section, we describe the FALCON algorithm. We ignore the implementation (i.e., the external libraries used) to focus on the algorithms themselves.

3.2.1 Overview of the whole algorithm

Pseudocode 1 contains the main steps of FALCON. To keep it concise, it does not contain a detailed description of all the algorithms used, which are available in the following subsections.

In this pseudocode, we can see that FALCON makes sure that buckets only contain spectra with precursor masses in the same interval. This is explained in the next subsection.

3.2.2 Buckets creation

As it is illustrated in Pseudocode 2, the input spectra are assigned to several buckets. Each bucket $B_{z,[a,b[}$ contains spectra having the same precursor charge z , whose precursor masses (and thus m/z)¹ belong to the same interval $[a, b[$ ². By processing the buckets separately, we will always respect the precursor charge constraint. By grouping the spectra having precursor masses in the same mass interval, most of the pairs of spectra respecting the precursor mass constraint will be put in the same bucket if

¹ $p_m(s) = p_{m/z}(s) * p_z(s)$

²By default, each bucket covers an interval of size 1 m/z (remember that the m/z is proportional to the mass).

Pseudocode 1 Main steps of FALCON. The highlighted lines represent the generic steps of the clustering, that could be used for a different clustering application.

Input: S , a set of spectra

Input: $k(n_B)$, the functions used to determine the number of partitions used in a bucket of size n_B

Input: m , the number of partitions to explore during the ANN (approximate nearest neighbors) search

Input: w_{all} , the number of ANN to find for each spectrum $\triangleright$ Ignoring the p_m constraint

Input: w , the number of ANN respecting the constraint to store in the distance matrix

Input: ϵ , the radius used by DBSCAN.

Input: min_{pts} , the density threshold used by DBSCAN.

```

1: B  $\leftarrow$  A set of buckets  $B_{z,[a,b[}$   $\triangleright$  A bucket is a set of spectra
2: for all spectra  $s \in S$  do
3:    $s \leftarrow$  Process  $s$ 
4:   Assign  $s$  to a bucket from B based on its metadata ( $p_z$  and  $p_{m/z}$ )
5: end for
6: for all precursor charges  $p_z$  do
7:   B $_z \leftarrow$  The set of buckets  $B_{z,[a,b[}$  of charge  $z = p_z$ 
8:    $D \leftarrow$  Empty distance matrix  $\triangleright$   $D$  is stored efficiently as a sparse matrix
9:   for all buckets  $B \in \mathbf{B}_z$  do
10:     $X \leftarrow$  Convert the spectra in  $B$  to vectors
11:     $X' \leftarrow$  Reduce the dimension of  $X$ 
12:    Partition  $X'$  in  $k(n_B)$  Voronoï cells using k-means
13:    for all  $\mathbf{x}' \in B$  do
14:      Find the  $w_{\text{all}}$  ANN's of  $\mathbf{x}'$  by exploring the  $m$  nearest partitions
15:      Remove the ANN's violating the precursor mass constraint
16:      Store the  $w$  smaller distances  $\text{dist}_\theta(\mathbf{x}', \text{ANN})$  in  $D$ 
17:    end for
18:  end for
19:  C  $\leftarrow$  DBSCAN( $D, \epsilon, \text{min}_{\text{pts}}$ )  $\triangleright$  C is a set of clusters
20:  Split the clusters violating the precursor mass constraint
21:  Export C
22: end for

```

$$\text{tol}^{p_m} \ll b - a$$

for spectra having similar precursor masses. Thus, we can restrict the search of neighbors within each bucket at the cost of losing a small proportion of potential neighbors (typically 1%). If a spectrum precursor mass is very close to the interval limit, it could have potential neighbors in the adjacent bucket.

Pseudocode 2 Buckets creation

Input: S , a set of spectra and their associated metadata p_z and $p_{m/z}$
B $\leftarrow$ A set of empty buckets $B_{z,[a,b[}$ (= sets of spectra). $\triangleright$ Each bucket contains spectra with the same precursor charge and with precursor masses in the same interval $[a, b[$
for all spectra $s \in S$ **do**
 $s \leftarrow$ Process the spectrum (peaks normalization, remove small peaks...)
 if s does not satisfy some conditions **then**
 continue
 end if
 $B_{z,[a,b[} \leftarrow B_{z,[a,b[} \cup \{s\}$ s.t. $z = p_z(s)$ and $a \leq p_m(s) < b$
end for
return B

Pseudocode 2 shows that some spectra are dropped if the peaks do not satisfy some conditions. These conditions include:

- The minimum number of peaks that the spectrum must have (by default, 5).
- The minimum m/z range between the two most distant peaks (by default, $250/p_z$).

These spectra are considered as being of low quality, and FALCON removes them. The preprocessing of the spectra includes the possibility to remove the small peaks (which are likely to be noise) and to scale the peaks to "de-emphasize overly dominant peaks" (for example, replace the intensity of each peak by its root square).

3.2.3 Conversion to vectors

Initially, each spectrum s_j consists in a set of pairs $\{(m/z_1, I_1), (m/z_2, I_2), \dots\}$ where m/z is the peak "position" (according to the x-axis) and I is the peak intensity (the "height" of the peak, on the y-axis). All the peaks whose m/z is in the range $[m/z_{\text{start}}, m/z_{\text{end}}[$ are mapped to the vector components:

$$x_i = \sum_{f(m/z_j)=i} I_j$$

where f is a function that linearly maps the m/z of the peaks to the corresponding component of the high dimensional vector $\mathbf{x} \in X$:

- The number of components is d ($\approx 30\,000$ by default, such that maximum one peak is assigned to each component);
- Each component x_i covers an interval of length $L = \frac{m/z_{\text{end}} - m/z_{\text{start}}}{d}$;

For example, the first component of $\mathbf{x}$ is the sum of the intensities of all the peaks (usually, maximum one peak) whose m/z are in the range $[m/z_{\text{start}}, m/z_{\text{start}} + L[$.

3.2.4 Dimensionality reduction

The spectra can be represented as very high dimensional (HD) vectors ($\approx 30\,000$ dimensions by default in FALCON). This is too high for efficient similarity searching during the pairwise distance matrix construction. The original vectors are converted to vectors of lower dimension (LD), 800 dimensions by default.

To perform the dimensionality reduction, FALCON uses a method called the *hashing-trick* described by Weinberger et al. [28]: the HD vectors are hashed to get LD vectors. More precisely, each component x'_i of the LD vector $\mathbf{x}'$ is the sum of several components x_j of the HD vector $\mathbf{x}$. Contrary to what is generally done during random projections, each component x_j is mapped to one and only one x'_i . The correspondence between $\mathbf{x}$ components and $\mathbf{x}'$ components is determined by a non-cryptographic function $h(j)$:

$$x'_i = \sum_{j:h(j)=i} x_j$$

The dimensionality reduction algorithm is illustrated on Pseudocode 3. According to the author, the method tends to conserve the Euclidean norm if d' , the number of components of $\mathbf{x}'$, is large enough: the Euclidean distance between two HD vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ should be approximately equal to the Euclidean distance between the corresponding LD vectors $\mathbf{x}'_1$ and $\mathbf{x}'_2$. To also conserve the angles separating the vectors (FALCON uses the cosine distance as distance function), the HD vectors are normalized (such that their norm is equal to 1) before the conversion.

Pseudocode 3 Dimensionality reduction

Input: d' , the number of dimensions desired for the low dimension (LD) vector

Input: $\mathbf{x} = \langle x_1, \dots, x_d \rangle$, a high dimension (HD) vector with $d' \ll d$

Input: $h : \mathbb{N} \rightarrow \{1, \dots, d'\}$, a non-cryptographic hash function

$x \leftarrow \text{Normalize } \mathbf{x}$

$\mathbf{x}' \leftarrow$ A null vector with m components

for all components x_j of $\mathbf{x}$ **do**

$i \leftarrow h(j)$

$x'_i \leftarrow x'_i + x_j$

end for

return $\mathbf{x}'$, a LD vector with d' components

3.2.5 DBSCAN

DBSCAN (Density-Based Algorithm for Discovering Clusters) is a clustering algorithm described in 1996 by Ester et al. [29]. This algorithm underlying idea is that clusters are high-density regions of the space separated by low-density areas. Its goal is to locate the core-points, i.e., points located inside the high-density areas, and to use them to expand clusters. To fully understand this algorithm, we need these definitions:

ϵ -neighborhood The ϵ -neighborhood, N_ϵ , of a point $\mathbf{x}_1$ is the set of points, including itself, that are at a distance smaller than ϵ , which is called the "radius". It is not required for the distance measure to be Euclidean. More formally, if X is the space of points,

$$N_\epsilon(\mathbf{x}_1) = \{\mathbf{x} \in X \mid \text{dist}(\mathbf{x}_1, \mathbf{x}) \leq \epsilon\}$$

Core point and border point The clusters are made of two types of points: the core points are inside the cluster, and the border points are at the border of the cluster. To be considered as a core point, a point must be surrounded by at least min_{pts} other points (the density threshold) located at a distance smaller than ϵ . It is the core point condition:

$$\mathbf{x} \text{ is a core point} \iff |N_\epsilon(\mathbf{x})| \geq \text{min}_{\text{pts}}$$

Density reachability If $\mathbf{x}_1$ is a core point, and $\mathbf{x}_2$ belongs to the ϵ -neighborhood of $\mathbf{x}_1$, $\mathbf{x}_2$ is directly density reachable from $\mathbf{x}_1$ (however, $\mathbf{x}_1$ is not always directly density reachable from $\mathbf{x}_2$, depending if $\mathbf{x}_2$ is a border point or not).

$$\mathbf{x}_2 \text{ is directly density reachable from } \mathbf{x}_1 \iff \mathbf{x}_2 \in N_\epsilon(\mathbf{x}_1)$$

If there exists a chain $\mathbf{x}_1, \dots, \mathbf{x}_{n-1}, \mathbf{x}_n$ such that $\mathbf{x}_{i+1}$ is directly reachable from $\mathbf{x}_i$, $\mathbf{x}_n$ is said to be **density reachable** from $\mathbf{x}_1$.

Density connectivity Two point $\mathbf{x}_1$ and $\mathbf{x}_2$ are density connected if they are both density reachable from another point $\mathbf{x}_3$.

Clusters and noise By definition, all the points belonging to a cluster must be density connected, and all the points density-connected must be in the same cluster. Points that do not belong to any cluster are said to be **noise**.

Based on these definitions, it is easy to understand how to build DBSCAN clusters: for each point p , we compute its ϵ -neighborhood, which tells us if p is a core point. If it is a core point, the ϵ -neighborhood of p is put in the same cluster, and the cluster is expanded by repeating the operation with the points belonging to the neighborhood. Pseudocode 4 describes the implementation of DBSCAN used in FALCON. It differs a bit from the original implementation because the ϵ -neighborhoods of the points are determined based on a distance matrix which is precomputed and provided to

DBSCAN. DBSCAN is a deterministic algorithm, except in the case where a point is a border point of two different clusters (the point is added in the first expanded cluster).

Since its publication, DBSCAN has been widely used for different applications. Here are some of its properties:

- It does not need to have prior knowledge about the number of clusters.
- It only works well if the clusters have similar densities. If this is not the case, the OPTICS algorithm can be used, a density-based algorithm that can detect clusters of different densities by increasing progressively ϵ [31].
- The algorithm supports any distance function, including the cosine distance.

Time complexity of DBSCAN The N_ϵ (ϵ -neighborhood) of each point is computed exactly once. The operation consisting of determining N_ϵ of a point is called a range query. Thus, the complexity of applying DBSCAN for charge z (remember that the spectra of different charges are clustered separately) is $\Theta(n_z * \text{RQ})$, where n_z is the number of spectra with precursor charge z and RQ is the number of operations required to get the ϵ -neighborhood of a point (this operation is called a **range query**). In FALCON, the distance matrix $D^{n_z \times n_z}$ is precomputed and provided to DBSCAN.

If we store the distance matrix using a naive structure, making a range query for spectrum $\mathbf{x}_i$ consists in accessing to all the elements in row i : it requires n_z operations (the size of the matrix is $n_z \times n_z$). In FALCON, only the w closest approximate neighbors of each vector are stored in the distance matrix. Thus, a special implementation for sparse matrices is used. To retrieve the elements stored in a row, it requires as many operations as there are elements in the row. Because this number is bounded by w , the time complexity of FALCON is:

$$\Theta(n_z * w)$$

which is $\Theta(n_z)$ if we consider w as being a constant parameter (for large n_z 's, this could lead to the loss of many close neighbors). Note that the spatial complexity corresponding to the sparse distance matrix is also $\Theta(n_z * w)$.

3.2.6 Computation of the distance matrix

The efficiency of FALCON is based on the idea that an approximate pairwise distance matrix can be built faster than the exact distance matrix. As we explained previously, the spectra are assigned to buckets based on their precursor charge p_z and their precursor mass p_m . Most of the potential neighbors of a spectrum are located in the same bucket. Thus, in FALCON, the distances are only computed between potential neighbors that are in the same bucket. To avoid having to compute the distance of a spectrum s_i (represented by the vector $\mathbf{x}'_i$) with all the other spectra of the same bucket B (which would be $O(n_B^2)$ without a specific structure such as an R-tree), the set of vectors representing the spectra is partitioned using k-means and the distance is only computed with vectors in partitions close to the partition of s_i . Pseudocode 5

Pseudocode 4 DBSCAN algorithm (adapted from Schubert et al. [30])

Input: $D_z^{n_z \times n_z}$, the distance matrix for spectra of charge z $\triangleright D_{i,j} \approx \text{dist}(s_i, s_j)$

Input: ϵ , the DBSCAN radius

Input: min_{pts} , the DBSCAN density threshold

```
1: function DBSCAN( $D_z, \epsilon, \text{min}_{\text{pts}}$ )
2:    $\mathbf{l} \leftarrow \{l_1 = \text{undefined}, \dots, l_{n_z} = \text{undefined}\}$   $\triangleright$  Default label for each spectrum index
3:   for  $i \in \{1, 2 \dots n_z\}$  do
4:     if  $l_i$  is undefined then
5:        $N_\epsilon(i) \leftarrow$  The indexes  $j$  where  $D_{i,j} < \epsilon$ 
6:       if  $|N_\epsilon(i)| < \text{min}_{\text{pts}}$  then  $\triangleright$  Not a core point
7:          $l_i \leftarrow$  noise
8:       else
9:          $l_i \leftarrow$  unused label
10:         $Q \leftarrow N_\epsilon \setminus \{i\}$ 
11:        EXPANDCLUSTER( $Q, l_i, D$ )  $\triangleright$  For each neighbor, expand the cluster
12:      end if
13:    end if
14:  end for
15:  return  $\mathbf{l} = \{l_1, l_2, \dots, l_n\}$   $\triangleright$  If  $l_i = l_j \neq \text{noise}$ ,  $s_i$  and  $s_j$  are in the same cluster
16: end function

17: function EXPANDCLUSTER( $Q, l, D$ )  $\triangleright l$  is the label value of the expanded cluster
18:   for all  $i \in Q$  do
19:     if  $l_i = \text{noise}$  then
20:        $l_i \leftarrow l$ 
21:     end if
22:     if  $l_i = \text{undefined}$  then
23:        $l_i \leftarrow l$ 
24:        $N_\epsilon(i) \leftarrow$  The indexes  $j$  where  $D_{i,j} < \epsilon$ 
25:       if  $|N_\epsilon(i)| \geq \text{min}_{\text{pts}}$  then  $\triangleright i$  is a core point
26:          $Q_i \leftarrow N_\epsilon(i) \setminus \{i\}$ 
27:         EXPANDCLUSTER( $Q_i, l, D$ )
28:       end if
29:     end if
30:   end for
31: end function
```

contains the steps of the construction of the distance matrix. Note that a different matrix is built for each precursor charge.

Distance function

The distance function used by FALCON is the cosine distance. For two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, the cosine distance is defined as

$$\text{dist}(\mathbf{x}_1, \mathbf{x}_2) = \text{dist}_\theta(\mathbf{x}_1, \mathbf{x}_2) = 1 - \cos(\theta(\mathbf{x}_1, \mathbf{x}_2))$$

where θ is the angle between $\mathbf{x}_1$ and $\mathbf{x}_2$.

Bucket partitioning

Lloyd's algorithm (k-means) is used to partition the spectra vectors contained in the bucket. At the end of the partitioning, each vector belongs to a Voronoï cells. Close vectors tend to be grouped in the same Voronoï cells. The time complexity of k-means is $O(n_B * k(n_B) * i_{\max} * d)$, where n_B is the number of points in the bucket B , $k(n_B)$ is the number of partitions (i.e. the number of centroids), $i_{\max}$ is the number of iterations realized and d is the dimension of the space. In FALCON, the space dimension is a constant, and the number of iterations is bounded (to 25). Thus, the time complexity becomes

$$O(n_B * k(n_B))$$

The number of partitions used for each bucket depends on the number n_B of vectors contained in the bucket: for each bucket, $k = f(n_B)$. However, f is not a continuous function. It depends on the range n_B belongs to (see Table 3.1). Thus, the time complexity of the partitioning step is expressed according to the considered range: for one bucket, it will grow in $O(n_B)$ until the bucket contains 1000 spectra, then it will grow in $O(n_B * 2^{\lfloor \log_2 \frac{n_B}{39} \rfloor})$ until 10^5 spectra (in the considered data set, each bucket contains at most $\approx 40\,000$ spectra, as it is shown in Figure 2.2).

Approximate nearest neighbors search

For each vector in the bucket, FALCON searches for the w_{all} approximate nearest neighbors: the vector is compared to all the vectors of the same partition and those of the m nearest partitions. Because the bucket is partitioned, this search is more efficient (the nearest neighbors are searched only in a few partitions), but there is no guarantee to find all the nearest neighbors; some of them can be lost. Thus, the vectors returned after searching the specified number of partitions are called **approximate nearest neighbors** (ANN).

Let's determine the time complexity of the approximate nearest neighbors search. Obviously, we need to make an assumption about the distribution of the vectors among the partitions. If all the spectra are in the same partition, this is equivalent to the situation where the bucket is not partitioned; the time complexity is $O(n_B^2)$ (each vector is compared to all the vectors of the bucket). Thus, we assume that

Pseudocode 5 Pairwise distance matrix construction for one precursor charge

Input: $\mathbf{B}_z$, the set of buckets containing spectra of charge z
Input: $\text{dist}(\mathbf{x}'_1, \mathbf{x}'_2)$, the distance function
Input: $k(n_B)$, the number of partitions to use in bucket B (function of $n_B = |B|$)
Input: m , the number of neighboring partitions to explore
Input: w_{all} , the number of ANN's to search for each spectrum
Input: w , the maximum number of ANN's stored in the distance matrix for each spectrum
Input: it_{max} , the maximal number of iterations of k-means
1: $D_z^{n_z \times n_z} \leftarrow$ A square matrix initialized to 1
2: **for all** buckets $B_{z,[a,b]}$ in $\mathbf{B}_z$ **do** ▷ k-means clustering
3: $n_B \leftarrow |B_{z,[a,b]}|$
4: Initialize $k(n_B)$ centroids.
5: **while** solution has not converged and $\text{it}_{\text{max}} < 25$ **do**
6: **for all** $\mathbf{x}' \in B_{z,[a,b]}$ **do**
7: Assign $\mathbf{x}'$ to the closest centroid
8: **end for**
9: **for all** centroids **do**
10: Redefine the centroid as the center of mass of the vectors assigned to
it
11: **end for**
12: **end while**
13: **for all** $\mathbf{x}'_i \in B_{z,[a,b]}$ **do** ▷ Approximate nearest neighbors' search
14: $c_i \leftarrow c(\mathbf{x}'_i)$ ▷ The index of the centroid $\mathbf{x}'$ is assigned to
15: $P \leftarrow$ The indices of the m nearest centroids to c_i (including c_i)
16: $R \leftarrow \{\}$
17: **for all** $\mathbf{x}'_j \iff c(\mathbf{x}'_j) \in P$ **do**
18: $R = R \cup \{(j, \text{dist}(\mathbf{x}'_i, \mathbf{x}'_j))\}$
19: **end for**
20: $R \leftarrow$ Keep the w_{all} tuples with smaller distances from R
21: $R \leftarrow$ Remove the tuples (j, dist) for which $|p_m(\mathbf{x}') - p_m(\mathbf{x}'_j)| > \text{tol}^{p_m}$
22: $R \leftarrow$ Keep the w tuples with smaller distances
23: $D_{i,j} \leftarrow$ Store the distances corresponding to tuples in R
24: **end for**
25: **end for**
26: **return** D_z , the approximate distance matrix for the spectra in buckets of charge z

Range	k	k-means	ANN search
$n_B < 100$	1	$O(1)$	$O(n_B^2)$
$100 < n_B \leq 10^5$	$2^{\lfloor \log_2 \frac{n_B}{39} \rfloor}$	$O(n_B * 2^{\lfloor \log_2 \frac{n_B}{39} \rfloor})$	$O(n_B^2 / 2^{\lfloor \log_2 \frac{n_B}{39} \rfloor})$
$10^5 < n_B \leq 10^6$	2^{16}	$O(n_B)$	$O(n_B^2)$
$10^6 < n_B \leq 10^7$	2^{18}	$O(n_B)$	$O(n_B^2)$
$n_B > 10^7$	2^{20}	$O(n_B)$	$O(n_B^2)$

Table 3.1: The number of partitions to use in each bucket depends on the number n_B of vectors contained in the bucket. Thus, a global time complexity cannot be determined, because it has to be expressed with respect to the considered range. For the Draft of the Human Proteome, a bucket covering an interval of 1 m/z contains at most $\approx 40\,000$ vectors (see Figure 2.2), which means that we are mainly interested in by the two first ranges ($n_B < 100$ and $100 < n_B \leq 10^5$). The number of partitions and the corresponding ranges used by the FALCON authors come from the Faiss documentation, Faiss being the library used to make the nearest neighbors search (see Section 3.3).

the number of spectra in each partition is proportional to n_B . When the number of partitions increases, the number of spectra in each partition tends to decrease. For a partition V , the number of spectra it contains tends to be proportional to

$$n_V \approx \frac{n_B}{k(n_B)}$$

where n_V is the number of vectors in partition V , and n_B is the number of spectra in the bucket. Assume we look for the approximate nearest neighbors of $\mathbf{x}$, which belongs to partition V . The number of comparisons $c_{\mathbf{x}}$ to do is the sum of the number of vectors in the m closest partitions:

$$c_{\mathbf{x}} \approx \sum_{j \in P_V} \frac{n_B}{k(n_B)} = m \frac{n_B}{k(n_B)}$$

where P_V is the set containing the m closest partitions to partition V . So, under the assumption made above, the time complexity to find the approximate nearest neighbors of one vector is $O(n_B/k(n_B))$, because m is a constant. If we consider all the vectors in the bucket, the time complexity becomes $O(n_B^2/k(n_B))$. As for k-means, the number of partitions $k(n_B)$ is a non-continuous function that depends on the range the number of vectors belongs to. The time complexity for each range is indicated in Table 3.1. The distances between $\mathbf{x}$ and the w approximate nearest neighbors respecting the constraints are stored in the distance matrix $D^{n_B \times n_B}$.

Approximate pairwise distance matrix

The approximate pairwise distance matrix can be seen as an approximation of the exact distance matrix, which only contains the w smallest distances for each spectrum. This matrix is computed with the vectors $\mathbf{x}'$ of the space of reduced dimension X' . It differs from the distance matrix that would have been computed from the original space X for several reasons:

1. The loss of precision caused by the conversion of high-dimension vectors $\mathbf{x}_i$ to vectors of lower dimension $\mathbf{x}'_i$.
2. The fact that only the w ANN of each spectrum are stored in D . If w is not large enough and there are many close neighbors, many of them will be lost because not enough partitions are explored.
3. A few potential neighbors can be in different buckets; they are not compared.

Suppose we assume the precursor masses of the spectra are more or less uniformly distributed inside each bucket. In that case, only a tiny proportion of pairs of spectra respect the precursor mass constraint. As we have seen previously, the distance between two spectra not respecting this constraint is assumed to be 1, the maximal distance. Thus, the distance matrix D contains a majority of entries equal to 1. It is a sparse matrix (to be exact, we should say that $I_{nz} - D$, the similarity matrix, is a sparse matrix such that the matrix contains a majority of 0's)³.

3.2.7 Postprocessing

The approximate pairwise distance matrix only contains distances between potential neighbors, i.e., spectra respecting the two constraints described above. However, during the clustering by DBSCAN, it can happen that two spectra violating the constraints are put in the same cluster by transitive closure. For example, s_1 is a potential neighbor of s_2 , s_2 is a potential neighbor of s_3 but s_1 and s_3 have a precursor mass difference which is larger than the threshold, DBSCAN can potentially cluster these points together. To respect the precursor mass constraint in each cluster, these clusters need to be split. This is done by hierarchical bottom-up clustering with complete linkage (the distance between two clusters is defined as the distance between the two spectra having the most dissimilar precursor mass). The postprocessing is illustrated in Pseudocode 6.

3.3 Implementation

Multithreading

Generally speaking, many parts of the program were designed to be run with several threads to improve performance. For example, the `Parallel` function of the `joblib` library is used several times to run as many threads as there are cores on the device running the software.

3.3.1 Buckets creation

Original implementation The `Pickle` library is used to store the buckets on disk to a) decrease the amount of main memory required b) be able to reuse the imported spectra from one execution to another. Concretely, one `.pk1` file is created for each

³ I is the identity matrix.

Pseudocode 6 Postprocessing of the clusters (hierarchical clustering)

Input: $\mathbf{C}$, the clusters of spectra generated by DBSCAN

Input: tol^{p_m} , the difference of precursor mass allowed by the constraint

```

for all clusters  $C_i$  in  $\mathbf{C}$  do
  if  $\exists s_1, s_2 \in C_i : |p_m(s_1) - p_m(s_2)| > \text{tol}^{p_m}$  then
    Remove  $C_i$  from  $\mathbf{C}$ 
     $\mathbf{N} \leftarrow []$  ▷ An empty list of clusters
  for all spectra  $s_j \in C_i = \{s_1, s_2, \dots, s_r\}$  do
     $N_j \leftarrow \{s_j\}$ , ▷ Create one cluster per spectrum
    Add  $N_j$  to  $\mathbf{N}$ 
  end for
  while True do
    Smallest distance  $\leftarrow \infty$ 
    Best pair  $\leftarrow \phi$ , the pair of clusters corresponding to the smallest distance
    for all pairs of clusters  $(N_p, N_q)$ ,  $p \neq q$  and  $N_p, N_q \in \mathbf{N}$  do
       $(s_p, s_q) = \arg \max_{s_a \in N_p, s_b \in N_q, N_p \neq N_q} (|p_m(s_a) - p_m(s_b)|)$ 
       $\text{dist}(N_p, N_q) \leftarrow |p_m(s_p) - p_m(s_q)|$ 
      if  $\text{dist}(N_p, N_q) < \text{smallest distance}$  then
        Smallest distance  $\leftarrow \text{dist}(N_p, N_q)$ 
        Best pair  $\leftarrow \text{Store the pair } (N_p, N_q)$ 
      end if
    end for
     $(N_1, N_2) \leftarrow \text{Best pair}$ 
    if  $\text{dist}(N_1, N_2) > \text{tol}^{p_m}$  then
      Remove from  $\mathbf{N}$  all the clusters containing only one spectrum ▷
      Labeled as noise
       $\mathbf{C} \leftarrow \text{Add the clusters that are in } \mathbf{N}$ 
      break ▷ No more clusters can be merged without violating the
    end if
    Remove  $N_1$  and  $N_2$  from  $\mathbf{N}$ 
    Add  $N_1 \cup N_2$  to  $\mathbf{N}$  ▷ Merge the clusters
  end while
end if
end for
return  $\mathbf{C}$ , the set of clusters respecting the precursor mass constraint

```

bucket. When all input spectra are read, they are serialized and then assigned to the corresponding bucket file. Then, the spectra of each bucket are ordered according to their precursor mass. Note that when all the spectra were read, they are ordered inside each bucket, which has a time complexity of $O(n_B * \log(n_B))$ (in the average case) with the quicksort algorithm.

Modification The spectra are written to the corresponding files only when all the spectra were read. If the number of spectra is high, the main memory becomes a bottleneck. To import more than a few million spectra, we had to modify the current original implementation to write the spectra at the same time they are read (with a consumer/producer implementation).

3.3.2 Spectra manipulation

To perform operations on the spectra (scaling, filtering the peaks...), the *spectrum_utils* Python package is used [32].

3.3.3 Fast neighbors search

The partitioning of the buckets followed by the nearest neighbors search is done with the Faiss library, developed by the Facebook AI Research [33]. According to its Github repository⁴, "*Faiss is a library for efficient similarity search and clustering of dense vectors. It contains algorithms that search in sets of vectors of any size, up to ones that may not fit in RAM. It also contains supporting code for evaluation and parameter tuning. Faiss is written in C++ with complete wrappers for Python/NumPy. Some of the most useful algorithms are implemented on the GPU*". Based on this description, we can understand why Faiss is well suited for big data.

More precisely, the `IndexIVFFlat` index is used to store the spectra vectors and partition them. By default, when we require the n nearest neighbors of a spectrum, it is compared to all the same partition spectra. However, it is possible to explore more partitions by specifying the `n_probe` argument. The partitions with a centroid located near to the centroid of the partition the spectrum belongs to are explored first. Faiss supports the cosine distance since it supports the dot product and the cosine distance is a dot product of normalized vectors. The Faiss library is implemented such that it can use the GPU of the device the program is executed on to perform similarity search for several spectra simultaneously.

The distance matrix is stored as a SciPy compressed sparse row matrix (see Figure 3.1) to avoid extreme memory requirements for a matrix that will contain a majority of entries equal to 1 (the maximum distance) corresponding to spectra that do not respect the precursor mass tolerance.

⁴<https://github.com/facebookresearch/faiss>

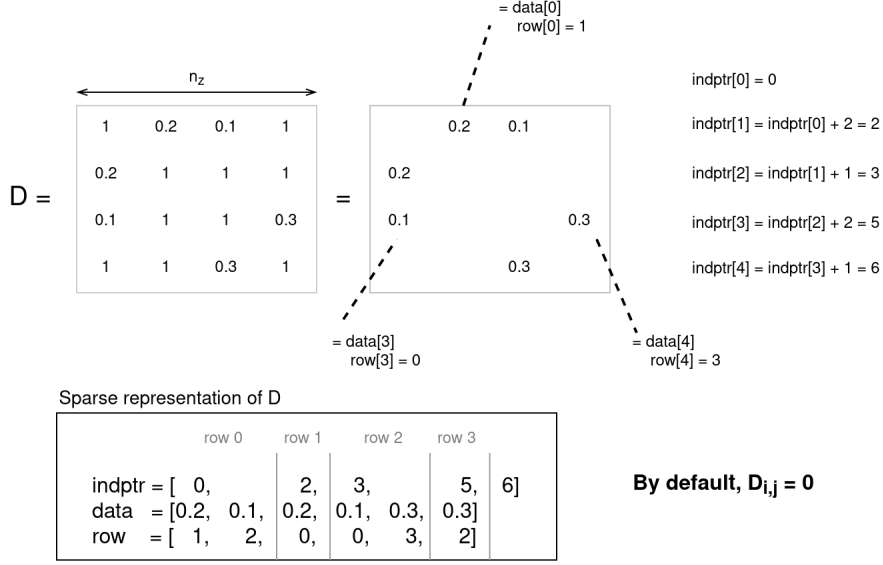


Figure 3.1: To avoid having to store the entire distance matrix, which would be $O(n_z^2)$ in space, a SciPy compressed sparse row matrix is used. It consists in three lists. The values in row i can be accessed as `data[indptr[i]]`, `data[indptr[i]+1]`, ..., `data[indptr[i+1]-2]`, `data[indptr[i+1]-1]`. The index of the row corresponding to `data[indptr[i]]` is stored in `row[indptr[i]]`, etc. With this structure, the spatial complexity to store the matrix depends on the number of values ($\neq 1$, which is the maximal distance) stored in the distance matrix. The time required to access all the elements in a row depends on the number of values ($\neq 1$) stored in the row. Since in FALCON this value is bounded to the w nearest neighbors (64 by default), the time complexity to access one row is $O(w) = O(1)$.

3.3.4 DBSCAN

The DBSCAN implementation of the scikit-learn Python library is used [34]. Note that instead of using the usual constructor, the preprocessing steps are done directly in FALCON. An inner function of the DBSCAN implementation is called to generate the clusters and "avoid unnecessary memory consumption", according to the author.

3.3.5 Postprocessing

The function used to generate the final clusters (respecting the precursor mass constraint) is the *fcluster* function of the SciPy package. This function requires the linkage matrix generated with the *fastcluster* function belonging to the package of the same name. This "complete linkage" matrix is based on the precursor mass of the spectra belonging to the cluster.

3.4 Time complexity of Falcon

The time complexity of FALCON determines its scalability. It is therefore essential to analyze it. In their publication, Bittremieux et al. [23] have shown that the measured execution time of FALCON is linear over the range $n \in [0; 25 * 10^6]$. In this section,

we analyze how the global complexity can be determined from the complexities of the different parts of the FALCON algorithm presented above. Moreover, we predict if its scalability will remain linear for larger datasets.

3.4.1 Preprocessing, DBSCAN and postprocessing

Exporting the spectra to their buckets is assumed to be $O(n)$. Ordering the buckets is $O(n_B * \log(n_B))$. These two steps are only done once for a dataset because the buckets are written on disk; they can be reused from one execution to another.

The time needed to run DBSCAN is assumed to be insignificant compared to the time required to build the approximate distance matrix (this assumption is confirmed in practice for our dataset size). Moreover, the time complexity of DBSCAN is $O(w * n_z) = O(n_z)$ if w , the number of nearest neighbors stored in the distance matrix for each spectrum, is considered as a constant (note that this could lead to an ever greater loss of near neighbors as the number of clustered spectra increases!). The time required for the postprocessing step can also be neglected because only a minority of clusters violate the precursor mass constraint, and the cluster sizes tend to be insignificant compared to the whole data set (see Chapter 4). Thus, we will that the execution time is dominated by the pairwise matrix's construction: the partitioning and the nearest neighbors search, in each bucket B .

3.4.2 Approximate distance matrix construction

In Table 3.1, we have indicated the time complexity of the k-means partitioning, and the ANN search, depending on the range n_B , the number of spectra in bucket B , belongs to. This time complexity is expressed for individual buckets. Since the number of buckets is constant and does not depend on the number of input spectra, the global time complexity of the distance matrix construction is determined by the complexity of the individual buckets (i.e., if for each bucket, the time complexity is $O(n^2)$, the global complexity is also $O(n^2)$).

Less than 10^5 spectra per bucket

For the Draft of the Human Proteome, with the default interval of 1 m/z covered by each bucket, the buckets size $n_B \in [0, 40000[$. In Figure 3.2, we plot the predicted time complexity (see Table 3.1) and the measured execution time for the two steps of the nearest neighbors search. A correlation can be observed between the predicted complexity and the measured execution time. If we disregard the buckets containing less than 100 elements ($n_B < 100$), for which the time required to perform an exhaustive nearest neighbors search is assumed to be negligible, all the buckets containing the spectra of the Draft of the Human Proteome dataset are in the range $100 < n_B \leq 10^5$: the number of Voronoï cells generated by k-means is determined as $k = 2^{\lfloor \log_2 \frac{n_B}{39} \rfloor} \approx \frac{n_B}{39}$. Suppose we disregard the function that rounds the exponent downwards, which gives the complexity function a particular shape (similar to a step function). In that case, we see that over the range of interest ($n_B < 10^5$), the k-means complexity tends to increase in $O(n_B^2)$ and the ANN search tends to

follow a $O(n_B)$ (this can be seen by removing the "rounding" in the expression of the complexity function, and by replacing k in the complexities determined previously).

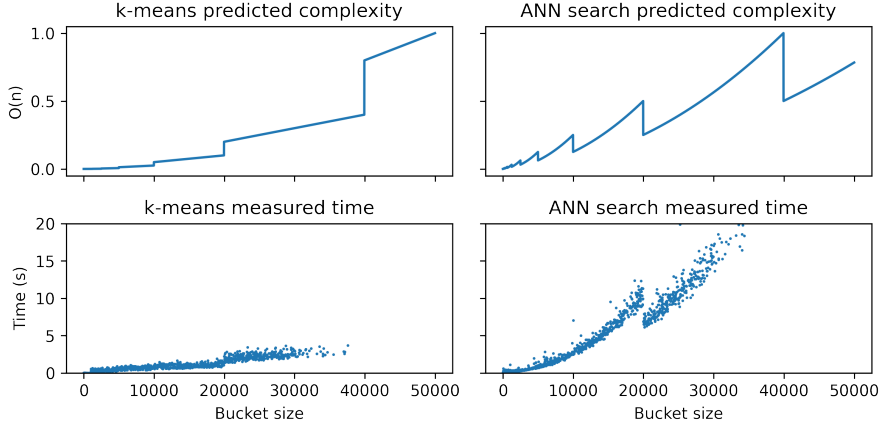


Figure 3.2: The complexity function (see Table 3.1) for the two steps of the pairwise distance matrix construction: the partitioning and the nearest neighbors' search. The characteristics of the device used for the time measurements are shown in Appendix A.

The measured times in Figure 3.2 show that even if k-means has a worse time complexity than the ANN search in theory, its execution time remains much lower in practice for the considered range. Thus, the time complexity of FALCON is dominated by the ANN search, which tends to grow linearly. These observations explain why the FALCON execution time grows linearly in the considered interval $100 < n_B \leq 10^5$, as long as the k-means execution time can be neglected compared to the ANN execution time.

Larger datasets

A 20 million spectrum dataset is large, but one might be interested in clustering even larger datasets. Thus, we can wonder if the time complexity will remain linear for larger buckets.

Table 3.1 shows that for $n_B > 10^7$, the number of Voronoï cells k remains constant ($k = 2^{20}$). Thus, the k-means time complexity is $O(n_B)$, but the ANN complexity is $O(n_B^2)$. Partitioning such a large dataset becomes very time consuming, but it is also possible to perform k-means on a subset and extend the Voronoï cells to the whole bucket. However, $n_B > 10^7$ is more than two orders of magnitude larger compared to the maximal bucket size we encounter in a large dataset such as the Draft of the Human Proteome. Even though buckets of size $n_B > 10^7$ could be found by clustering hundreds of datasets together, we will now look at the last intervals we have not yet considered, $10^5 < n_B \leq 10^6$ and $10^6 < n_B \leq 10^7$.

Since the number k of Voronoï cells is constant in each of these intervals, we expect k-means to be $O(n_B)$ and the ANN search $O(n_B^2)$ locally over these intervals. The number of Voronoï cell used by k-means is increased from one interval to another as n_B increases to keep a good ratio between the execution time of k-means and the

ANN search (increasing k decreases the time required for the ANN search at the cost of increasing the time required to partition the vectors). Thus, when considering both k-means and the ANN search for the intervals where $10^5 < n_B \leq 10^7$, we expect to observe a $O(n_B^2)$ complexity locally, but the execution time is "artificially" attenuated by increasing the value of k as n_B increases.

3.4.3 Comparison with MS-Clustering and Spectra-cluster

MS-Clustering [14] and Spectra-cluster [5] are two competitors of FALCON. They both rely on the same clustering algorithm, presented as a greedy agglomerative hierarchical clustering algorithm by the authors. In this section, we briefly present this clustering algorithm and compare its time complexity to that of FALCON.

MS-Clustering and Spectra-cluster also cluster subsets of spectra (that we will call buckets to keep the same terminology as in FALCON) independently by taking into account the precursor m/z constraint and the precursor charge constraint. However, we do not go into too many details, and we focus on the algorithm used to cluster the different subsets, presented in Pseudocode 7. The clustering algorithm requires several rounds, using decreasing distance thresholds. Note that the distance between two clusters is measured as the distance between their respective consensus spectra (a method to build such a consensus spectrum will be presented in Section 4.3.1), which avoids having to compute the distances between all the pairs of spectra of two clusters as it is usually the case with classical hierarchical clustering algorithms.

Pseudocode 7 Clustering algorithm used by MS-Clustering and Spectra-cluster

Input: B , the set containing the n_B spectra to cluster

Input: $[\epsilon_1, \epsilon_2, \dots, \epsilon_r]$, the distance threshold allowed during each round $\triangleright \epsilon_i < \epsilon_{i+1}$

```

1:  $\mathbf{C} \leftarrow [C_1, C_2, \dots, C_{n_B}]$   $\triangleright$  Assign each spectrum to a cluster and build a list
2: for all  $\epsilon_i$  do  $\triangleright$  Start with the smallest  $\epsilon$ 
3:   for all  $C_i \in \mathbf{C}$  do
4:     for all  $C_j \in \mathbf{C}, j < i$  do
5:       if  $\text{dist}(\text{consensus}(C_i), \text{consensus}(C_j)) < \epsilon_i$  then
6:         Add the spectra of  $C_i$  in  $C_j$ 
7:         Remove  $C_i$ 
8:         break
9:       end if
10:    end for
11:  end for
12: end for
13: return
```

Pseudocode 7 shows that during each round (the number of rounds is a parameter, typically 5), each cluster is compared to all the clusters that precede it in the list $\mathbf{C}$. Thus, the number of comparisons realized during each round is determined by the distance threshold ϵ_i and the distances between the spectra. We illustrate this by two border cases.

1. All the pairs of spectra in the bucket are perfectly similar (or the distance

threshold is maximal). In this case, during the first round, C_2 is first merged with C_1 . Then C_3 is merged with C_1 , etc: all the clusters will progressively be merged with C_1 . Each cluster is compared to exactly one cluster (C_1). Thus, for the particular case where each spectrum is near the other spectra, the time complexity is $O(n)$.

2. The distance between pairs of spectra is always larger than the threshold. In this case, no clusters are merged. During a round, C_2 is compared to C_1 , but the clusters are not merged since they are dissimilar. Then, C_3 is compared to C_1 and C_2 . In short, cluster C_i is compared with the $i - 1$ precedent clusters. Thus, the total number of comparisons is

$$\sum_{i=1}^{n_B-1} i = 1 + 2 + \dots + (n_B - 1) = \frac{(n_B - 1)n_B}{2}$$

and the time complexity is therefore $O(n^2)$.

We are between these two extreme cases for a particular dataset: the time complexity of MS-Clustering and Spectra-cluster is $O(n^2)$.

3.5 Why does Falcon seem to be a good choice?

Time efficiency The authors of FALCON have shown that it is several times faster than the widely used MS-Cluster and spectra-cluster (around 4x times faster for the Draft of the Human Proteome, the dataset of 25 million spectra they used to benchmark their algorithm). Its time complexity explains this: for a dataset containing less than 100 000 spectra per bucket, such as the Draft of the Human Proteome, the time complexity of FALCON is $O(n)$. In comparison, the time complexity of MS-Clustering and Spectra-cluster is $O(n^2)$.

Memory requirements Thanks to its ability to assign the spectra to buckets stored on disk and load the buckets one at a time during the construction of the pairwise matrix, it is possible to run FALCON on a device with a limited amount of main memory (the amount of main memory required is bounded by the size of the bucket containing the most significant number of spectra and the size occupied by the pairwise distance matrix, which grows linearly since the number of elements in each row is bounded).

Implementation The source code of FALCON is very clear and concise because it uses many well-known external libraries such as scikit-learn for DBSCAN, Faiss to build indexes and search nearest neighbors, etc. Consequently, it is a lot easier to make modifications to the FALCON than Spectra-cluster, for example.

Chapter 4

Results

After describing the Draft of the Human Proteome in Chapter 2, and the clustering tool we selected to cluster it in Chapter 3, it is now time to look at the results (i.e., the final clusters). In particular, we will see how we could identify many clusters containing unidentified spectra.

4.1 Parameters selection

As it was described in Chapter 3, FALCON relies on many parameters used for:

- The dimension reduction (e.g., the number of components in the low-dimensional space);
- The approximate neighbors search (the number of Voronoï cells to explore, the number of ANN to search, etc.);
- The DBSCAN clustering (the radius ϵ and the density threshold $\min_{pts}$).

Moreover, several parameters are related to the mass spectrometry, such as the m/z tolerance to consider different spectra peaks as "matching" and $\text{tol}^{m/z}$ described previously.

It would be impossible to perform a grid search with all the hyperparameters. The number of possibilities would be too large. However, these parameters are provided with default values, tuned by the authors of FALCON on the Draft of the Human Proteome (we use the same data set) [23]. Thus, it can be assumed that these parameters are well designed for the dataset, and we will only evaluate the impact of ϵ on the clustering. The clustering quality will be evaluated using two complementary metrics relying on the known spectra labels, the homogeneity, and the completeness.

4.1.1 Homogeneity, completeness and V-measure

Homogeneity and completeness are entropy-based metrics used to assess the quality of a clustering when some labels are known [35]. More precisely, these metrics can be used to evaluate if clusters tend to contain only points of the same class

(homogeneity) and if all the elements of the same class tend to be in the same clusters (completeness). Both metrics range from 0 to 1, where 1 is a desirable value.

These metrics are complementary. When homogeneity increases, completeness decreases and vice-versa. For example, if each point is assigned to a singleton cluster, the homogeneity will be maximal (equal to 1), and the completeness will be low. In the opposite extreme, where all points are assigned to a single, large cluster, the completeness will be maximal but not the homogeneity.

Homogeneity

The homogeneity is maximal (equal to 1) if each cluster contains points with the same label. It is defined as:

$$h = \begin{cases} 1 & \text{if } H(C|K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{otherwise} \end{cases} \quad (4.1)$$

where C is the set of classes (the set of different labels), K is the set of clusters, $H(C|K)$ is the conditional entropy of the class distribution given the proposed clustering, and $H(C)$ is the maximum reduction in entropy the clustering could provide ($H(C|K) \leq H(C)$). If N is the number of data points and a_{ij} is the number of elements of class i being in the cluster j , the previous entropies are defined as follows [35]:

$$\begin{aligned} H(C|K) &= - \sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}} \\ H(C) &= - \sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{|C|} \log \frac{\sum_{k=1}^{|K|} a_{ck}}{|C|} \end{aligned}$$

Completeness

The completeness is maximal (equal to 1) if all the points belonging to a same class are systematically grouped in the same clusters. It is defined as:

$$h = \begin{cases} 1 & \text{if } H(K|C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{otherwise} \end{cases} \quad (4.2)$$

where $H(K|C)$ is the conditional entropy of the proposed clustering given the class distribution, having a maximal value of $H(K)$. The definitions of $H(K|C)$ and $H(K)$ are obtained similarly as for $H(C|K)$ and $H(C)$, by permuting C and K .

V-measure

To take into account both the homogeneity and the completeness at the same time, Rosenberg et al. [35] propose to use the V-measure, which is the weighted harmonic mean of the homogeneity and the completeness:

$$V_\beta = \frac{(1 + \beta) * h * c}{\beta * h + c}$$

where β is the weight used to give more importance to the homogeneity ($\beta < 1$), more importance to the completeness ($\beta > 1$) or equal importance to them ($\beta = 1$).

4.1.2 Epsilon

Figure 4.1 shows the evolution of several metrics of the FALCON final clustering for different value of ϵ (0.01, 0.1, 0.2, 0.3, 0.4 and 0.5). The proportion of clustered spectra and the average size of the clusters increase with ϵ (for these measures, the spectra marked as "noise" by DBSCAN are considered as "not being in clusters"). By increasing ϵ , we allow less similar spectra to be clustered together, and the probability of finding at least one other similar spectrum in the data set increases. That leads to larger clusters on average, even though we can observe that the median cluster size remains constant (5 spectra per cluster).

We can observe that the homogeneity¹ decreases when ϵ increases (more spectra of different classes tend to be clustered together). Thus, a small homogeneity could indicate that the ϵ value used is too large. The completeness increases with ϵ . The "perfect clustering" (the clustering where all the spectra with the same label would be in the same (homogeneous) cluster) would correspond to homogeneity and completeness of 1. However, it can be observed in Figure 4.1 that this is impossible.

In previous studies [14] [17], it has been shown that it is often preferable to use small distance thresholds (for DBSCAN, a small ϵ , which favors homogeneity over completeness). These studies performed further analysis on the clusters by generating a consensus spectrum for each cluster, which is the average of the spectra in the cluster (an algorithm used to generate the consensus spectra will be described in Section 4.3.1). To obtain reliable results when submitting the consensus spectrum to the query engine, all the spectra in the cluster must correspond to the same peptide, which is more likely to occur if the homogeneity is high. If a small value of ϵ is used to have a high homogeneity, the risk is that homogeneous clusters will be split into several smaller clusters. However, this is usually not a problem since these clusters can be analyzed individually. In other words, for our application, homogeneity is more important than completeness.

Based on Figure 4.1, we will choose $\epsilon = 0.1$ for the following reasons:

- The homogeneity is very close to 1 ($h = 0.99$);
- The completeness is high ($c = 0.88$);
- The proportion of clustered spectra (i.e., spectra not tagged as being noise) is ≈ 0.5 , which is significantly larger than for $\epsilon = 0.01$;

Moreover, the V-measure is maximal for $\epsilon = 0.1$ when using a small weight (e.g. $\beta = 0.1$, which favors the homogeneity over the completeness).

¹The points labeled as noise were assigned to singleton clusters when measuring homogeneity and the completeness.

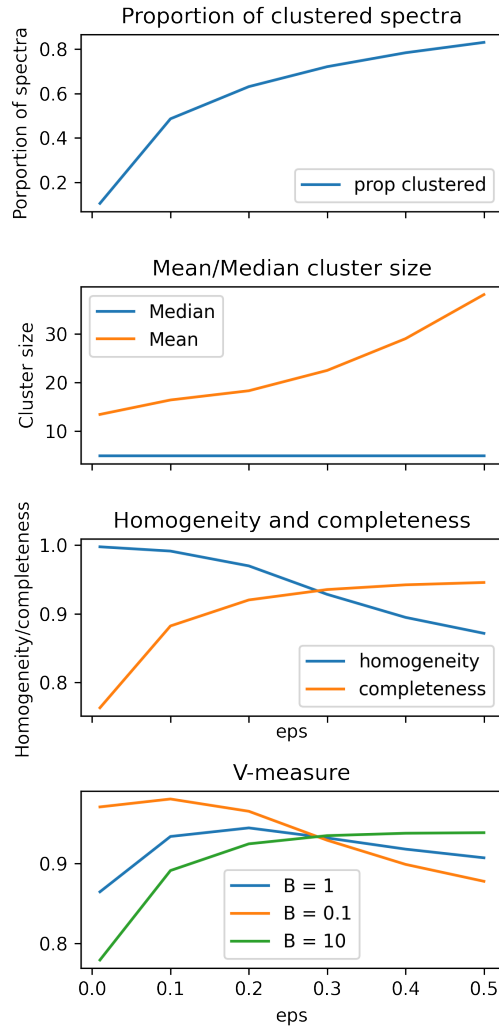


Figure 4.1: When ϵ increases, the proportion of clustered spectra increases as well the average size of the clusters. As expected, the homogeneity decreases and the completeness increases. The V-measure for $\beta = 0.1$ is maximal for $\epsilon = 0.1$.

4.2 Clusters overview

The clustering with the previously discussed parameters results in more than 1.2 million clusters containing on average 16.4 spectra.

4.2.1 Classification

Analyzing all the generated clusters would take a very long time. Instead, we can already make a first classification of the clusters according to the labels of the spectra they contain (Figure 4.2). This classification is partially inspired by [5]. We define four types of clusters:

- Unidentified clusters contain only unlabeled spectra;
- Identified clusters contain more than 70% of spectra sharing the same label;

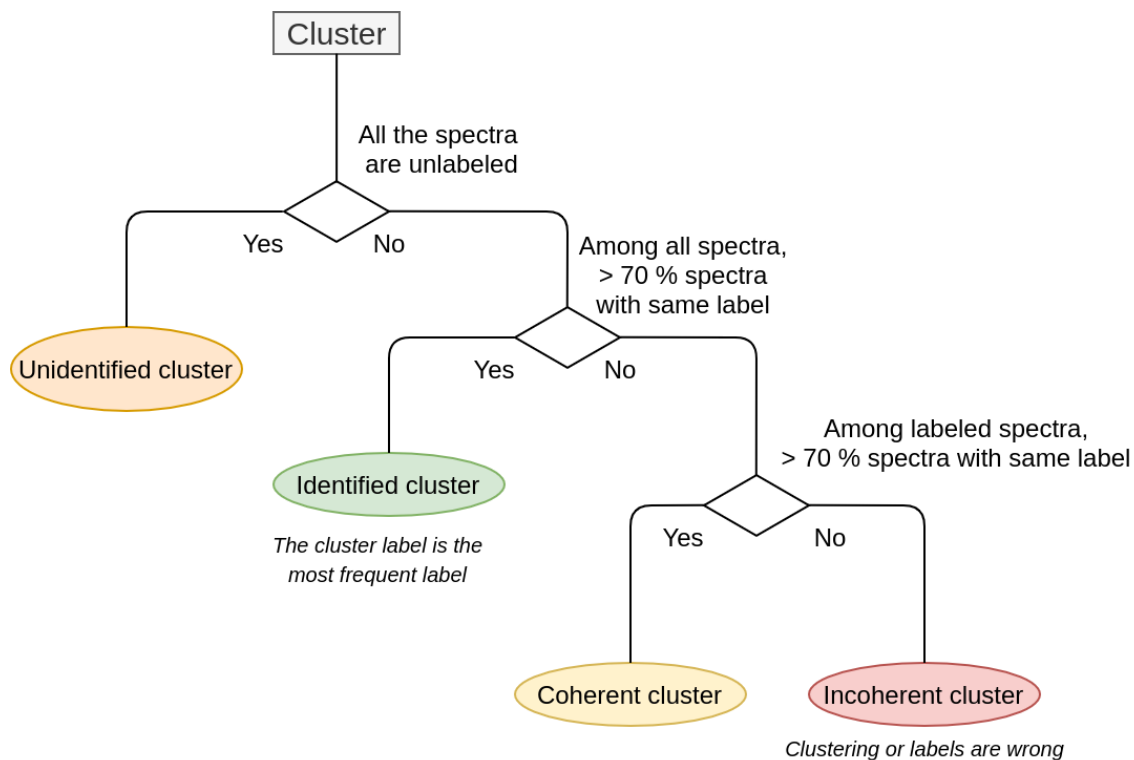


Figure 4.2: For convenience, we define several types of clusters, based on the labels of the spectra they contain.

- Coherent clusters can contain many unlabeled spectra, but the identified spectra tend to share the same label (more than 70 % of the identified spectra have the same label);
- Incoherent clusters tend to contain spectra with different labels.

These categories were defined for convenience to take a deeper look at the content of the clusters in the next section, but no further conclusions should be drawn at this stage. For example, a large cluster tagged as a "coherent cluster" could only contain one labeled spectrum for which the identification is wrong. Alternatively, an "identified cluster" could contain 80% of labeled spectra corresponding to one peptide and 20 % of spectra corresponding to another.

4.2.2 Clusters in the draft of the human proteome

Figure 4.3 shows the cluster sizes and cluster types distribution (left) and the proportion of each type of cluster (right). The following observations can be done.

Cluster sizes

The most represented cluster size is 2, and the majority of the clusters contain less than ten spectra. Most of the clusters contain less than 20 spectra (but note that

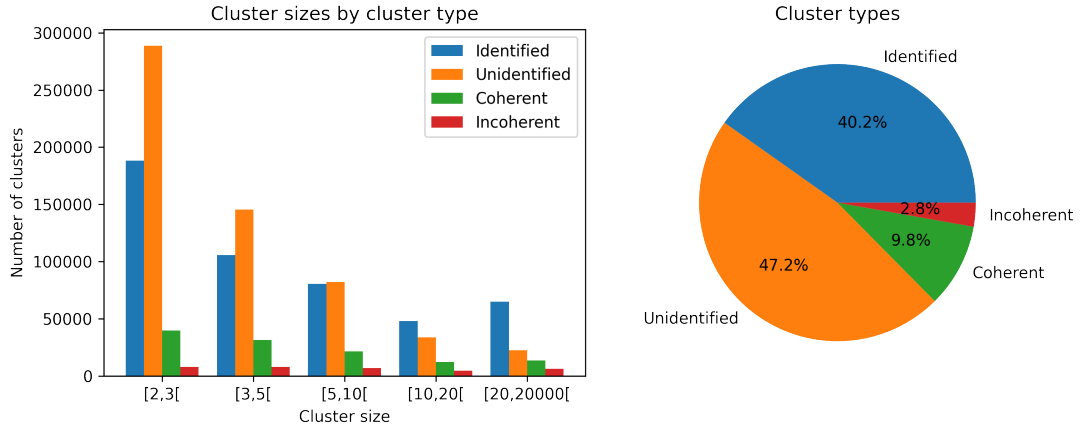


Figure 4.3: The most represented type of cluster is "unidentified", which means that no spectrum in the cluster has a label. The "incoherent" clusters only represent around 3% of the clusters. Even if the maximum cluster size is 16570, most clusters contain less than 20 spectra.

the largest cluster contains 16570 elements²). For clusters containing less than five spectra, the predominant cluster type is "unidentified" but for larger clusters (10 elements or more), the most represented type is "identified".

Cluster types

More than 47% of the spectra are "unidentified" (no spectrum in the cluster could be identified). More than 40% of the spectra are of type "identified", which means that more than 70% of the spectra have the same label. The "coherent" clusters represent around 10% of the clusters. Finally, the "incoherent" clusters (which contain many spectra with different labels) only represent around 3% of the clusters. This was expected because of the high value of the homogeneity for the parameters used in this clustering.

4.3 Unidentified clusters reanalysis

In the following, we will focus on the "unidentified" clusters. They represent almost half of the clusters (572104 clusters). Previous studies, such as Griss et al. [5], have shown that some of these clusters can correspond to peptides with PTMs (posttranslational modifications). Thus, we will reanalyze these clusters and ask the query engine to include more PTMs in the search. That leads to significantly longer search times, which is one reason why it would be impracticable to do a search including common PTMs for all the spectra. Another reason why common PTMs are usually not searched systematically is that very few peptides will typically

²For example, this particular cluster is "identified" and has the label 'VATVSLPR', which is a subsequence of the trypsin protein. Since the trypsin was used to digest the samples before processing them in the mass spectrometer, it seems logical to find many spectra corresponding to trypsin peptides.

present the PTM, which makes it unlikely to be found during the shotgun proteomics experiment.

4.3.1 Methodology

This section explains how we reanalyzed some of the unidentified clusters.

Consensus spectra

Instead of submitting all the spectra in the selected clusters, we decided to generate a consensus spectrum for each cluster. The same approach was used in many previous works, as briefly presented in the Chapter 1. The algorithm we used to build the consensus spectra is based on the method shown in the Supplementary information of Frank et al. [14]. It consists of grouping all peaks of all the spectra belonging to the same cluster and then merging the peaks with a similar m/z . The algorithm we used to generate the consensus spectra is described in Pseudocode 8. Note that it relies on the original representation of the spectra, not the vectors.

Database search

We analyzed the consensus spectra using MS-GF+ [26], at a 1% FDR. The PTMs we searched for can be classified into two categories. On the one hand, there are modifications which are not naturally present in cells and which are introduced (intentionally or not) during sample preparation:

- Carboxymethylation. This PTM is voluntarily introduced by biologists on the cysteine amino acid (using iodoacetamide before the protein digestion) to prevent the amino acid from forming disulfide bonds from one protein to another;
- Oxidation of the methionine amino acid. The oxidation is frequently unintentionally introduced during samples preprocessing;

On the other hand, these PTMs can happen naturally in the cells:

- Methylation of the lysine amino acid. This PTM can notably be used to promote/inhibit protein-protein interaction.
- Deamidation of asparagine and glutamine, which can modify the protein structure;
- N-terminal cyclization (glutamic acid $\rightarrow$ pyro-glutamic acid, and glycine $\rightarrow$ pyro-glutamic acid), which also can affect the protein structure.

It is important to note that not all of these posttranslational modifications present a deep biological interest. Since the first category of changes does not take place naturally in cells, it does not teach us anything about biological processes. However, not searching for these PTM would make it impossible to identify the spectra

corresponding to peptides presenting the PTM. Other PTMs, such as phosphorylation, are very interesting because they play a significant role in many biological processes.

Several PTMs can be present on the same peptide. This is the reason why searching for many modifications can lead to an exponential increase in the search duration: all the combinations have to be tested.

Cluster selection

Performing an open search with these PTMs on the consensus spectra of all these clusters would still be impracticable. For example, reanalyzing a hundred spectra can take approximately one hour on a typical computer³. Thus, we will focus on a subset of the "unidentified" clusters. Of course, that does not mean that no valuable information could be extracted from the other types of clusters (e.g., the incoherent clusters). However, for simplicity, we restrict our analysis to the unidentified clusters.

Since the number of "unidentified" clusters remains very high (47.2% of the 1.2M clusters), we decided to only select clusters in which all the spectra were acquired from the same tissue. That makes possible to identify some clusters containing spectra corresponding to peptides only present in some cellular types; which makes the biological interpretation of the consensus spectra reanalysis easier.

In short, we restrict our analysis to "unidentified" clusters containing only spectra generated from the same tissue. Thus, the results presented below cannot be generalized to all the unidentified clusters without further work.

Cluster sizes

Previously, Figure 4.3 showed that for the selected parameters, FALCON generates a majority of small clusters (containing 2 to 4 spectra), but also many larger clusters. Intuitively, we could wonder that larger clusters are more "reliable" than smaller clusters; the fact that many similar spectra were acquired during the experiment could indicate that this spectrum effectively corresponds to a given peptide. However, some large clusters could also contain spectra corresponding to different peptides (since we only analyze unidentified clusters, we cannot rely on labels to identify non homogeneous clusters). In that case, studying the consensus spectra generated for "mixed" clusters will not provide reliable results.

To evaluate the influence of the cluster size on the "reliability" of the consensus spectra (i.e., the sequence associated to the consensus spectrum can or cannot be identified by the query engine), we decided to analyze the consensus spectra for different sizes of clusters separately. We generated three consensus spectra batches:

1. Batch 1: 5001 consensus spectra, corresponding to clusters of size 2, 3, or 4.
2. Batch 2: 5001 consensus spectra, corresponding to clusters with sizes between 5 and 19.
3. Batch 3: 1121 consensus spectra, corresponding to all the clusters of size larger or equal to 20 respecting the criteria specified above.

³Dell Precision, 8 cores, 32 GB RAM.

4.3.2 Overview of the open search identifications

The proportion of consensus spectra that could be identified for each batch at a 1% FDR is shown in Table 4.1. It shows that consensus spectra corresponding to larger clusters are more likely to be identified during the open search. It seems coherent with the observations of Griss et al. [17], which reported that search engines identified consensus spectra of large clusters more reliably.

Batch	Cluster sizes	# consensus	# identified sp.	% identified sp.
1	[2, 5[	5001	2	0.04%
2	[5, 20[	5001	43	0.86%
3	> 20	1121	45	4.01%

Table 4.1: For the batch 1, containing the smaller clusters, only two spectra could be identified at a 1% FDR. The proportion of identified spectra is larger for batch 3 than for batch 2. These results could indicate that the consensus spectra of larger unidentified clusters are more likely to be identified.

Influence of the cluster size

Several factors can explain that the consensus spectra are more likely to be identified when the size of the cluster is larger.

Noise Not all the spectra generated by a mass spectrometer correspond to a single peptide. For example, chimeric spectra can contain peaks of different peptides. These spectra, the proportion of which is difficult to assess, are not very likely to be present in many copies in the dataset. However, as the number of clustered spectra increases, it becomes likely that two similar spectra are present in the dataset due to chance, even if they do not correspond to a single peptide. Thus, as the dataset size increases, a possibility could be to ignore the smallest clusters.

Consensus spectrum quality Previous studies [14] [5] have shown that using a consensus spectrum can improve the spectra identification, notably because it improves the signal-to-noise ratio. We can assume that this improvement is better for larger clusters than for clusters containing only two spectra. Thus, the fact that more spectra are used to generate the consensus spectrum can also explain that the consensus spectra of larger clusters can be more reliably identified.

4.3.3 Proteins and tissues

Since all the clusters we have selected contain spectra generated from the same tissues, it can be expected that some of the clusters correspond to proteins only present in that tissue. For many identifications (see examples in Table 4.2), it is the case. For example, finding a protein related to myelination in the spinal cord⁴ or a protein binding to pulmonary surfactant in the lungs is not surprising.

⁴The myelin is a substance used to insulate nerve cells

Many other identified proteins are not specific to one tissue or one cellular type. For example, a consensus spectrum is identified as a sequence from the "histone H2B". Since histones are present in (almost) all cells to "pack" the DNA, they can be found in all tissues. In that case, we could wonder why the cluster is formed only by spectra acquired from one tissue: it would be logical to find spectra from many different tissues. An explanation could be that by favoring homogeneity over completeness when choosing ϵ , we "split" a larger cluster into several small clusters, some of them containing only spectra from the same tissue due to chance. If the open search identified PTMs on the peptide, we could also hypothesize that the PTM is specific to one particular tissue, which would have a biological interest.

4.3.4 PostTranslational modifications

The occurrences of the PTMs are shown in Table 4.3. Note that several PTMs can be identified on the same sequence. Not all these PTMs have the same biological interest. For example, carbamidomethylations are introduced experimentally. Including this PTM in the open search is essential since it makes it possible to identify the sequences of the peptides presenting the PTM. Still, it does not provide any information about processes happening "naturally" in the cells. Note also that oxidations and carbamidomethylation were already included in the database search used to obtain the spectra labels.

Given their interest, many PTMs (such as phosphorylations and acetylations) are widely studied and referenced. Here, we investigate if the acetylations and phosphorylations identified during the open search are referenced in UniProt [36], a widely used collection of databases. We assume the identifications made during the open search are correct.

Batch 2

Among the four acetylations reported, three of them concern histones. It is interesting because it is known for a long time that the (de)acetylation of histones is used to regulate the transcription of the DNA [37]. Two of these acetylations are referenced on UniProt [36].

The last acetylation is identified on a sequence corresponding to the "serine hydroxymethyltransferase". It is not referenced on UniProt. Either the identification is a false positive (the database search identified the spectrum wrongly), or the PTM is not (yet) referenced on UniProt. A manual analysis of the spectrum would be required to decide if the identification could be a false positive.

The only phosphorylation reported in this batch concerns the "microtubule-associated protein tau" (UniProt: P10636). It is reported on UniProt (on the 498th amino acid).

Batch 3

The first acetylation concerns the nucleophosmin protein (UniProt: P06748). It is not reported on UniProt. The second concerns the thymosin beta-10 protein (UniProt: P63313), for which the PTM is referenced on UniProt (4th amino acid).

Phosphorylations were identified from five consensus spectra. Two of these spectra correspond to the same peptide (SQEESSEEGEEDATSEVDK) from the secretogranin-1 protein. The first one only has one phosphorylation on the first serine (S). The second one has two phosphorylations, on the first and the second serine. Thus, it is expected (and intended) that the spectra corresponding to these peptides are not in the same cluster since the additional phosphorylation shifts some of the spectrum peaks.

All of the phosphorylations are known on UniProt, with one exception. For the titin protein, a key component of striated muscles, two phosphorylations were identified. One of them (33247th amino acid) is referenced on UniProt, but not the other one, located on the 33241th amino acid.

4.3.5 Discussion

Newly identified clusters

The open search on consensus spectra of unidentified clusters makes it possible to identify peptides associated with many clusters. Since the identifications were obtained at a 1% FDR, they can be considered as being reliable. An interesting result is that almost no cluster containing less than five spectra could be identified. It can be expected that the more spectra are clustered, the more (small) clusters will appear due to chance. The proportion of identified consensus spectra seems to be positively correlated to the size of the clusters, but more detailed results should be produced to confirm this observation.

The consensus spectra identified during this reanalysis can be put in three categories:

- Some clusters can correspond to peptides presenting PTMs that were not searched during the individual analysis of the spectra.
- Other clusters could correspond to spectra identified during the individual analysis, but for which the confidence level was not sufficient (i.e., the FDR should have been relaxed to use the identification). It can be assumed that using consensus spectra, sometimes of better quality than individual spectra, played a role in being able to identify these spectra with sufficient confidence.
- Many clusters correspond to peptides without PTMs or with common PTMs that were already searched during the individual analysis of the spectra but could not be identified. For these spectra, it can also be assumed that generating a consensus spectrum increased the "quality" such that it could now be identified.

We observed that for many cluster, a direct link could be established between the identified protein and the tissue. Moreover, the majority of the PTMs reported where referenced in UniProt [36]. That makes us confident about the reliability of the identifications reported by the database search.

Clusters remaining unidentified

The proportion of identified consensus spectra can seem relatively low, even for batch 3 (4% of the consensus spectra). However, applying the reanalysis described in this Section to the entire set of unidentified clusters could lead to many new identifications (even if only clusters of minimum size are considered). Assigning sequences to unidentified spectra remains an open question, that Chick et al. [7] called the *unassigned-spectra enigma that is so pervasive in shotgun proteomics data sets*. They put several explanations forward.

First, in typical shotgun proteomics experiments, most unidentified spectra are of good quality. They are very likely to correspond to a peptide (this can be determined based on the peak pattern). Secondly, a large proportion of the unidentified spectra would originate from modified peptides (these include not only peptides with uncommon PTMs but also amino acid variants). By definition, database search can only identify sequences already known. Even by doing an open search, the peptides with amino acid variants will remain unidentified. To identify these spectra, one possibility would be to abandon the database search and use *de novo* sequencing to determine the spectrum sequence based on the peak pattern. Another possibility would be to include known variants in the database used for the database search. However, this would significantly increase the search space and could lead to more false positives, which would impact negatively the level of confidence in the identifications. Moreover, not all the protein variants are known.

Pseudocode 8 Consensus spectrum construction. Remember that each peak is defined by its m/z value and its intensity I , $P = (m/z, I)$.

Input: $C = \{s_1 = \{P_{1,1}, P_{1,2}, \dots\}, s_2 = \{P_{2,1}, P_{2,2}\}, \dots\}$ $\triangleright$ A cluster

Input: $\text{tol}_{m/z}$, the m/z tolerance to consider spectra peaks as matching

Input: $\max_P$, the maximal number of peaks to keep in one spectrum

```

1:  $s_{\text{cons}} = \{\}$   $\triangleright$  Initialize the consensus spectrum
2: for all  $s_i \in C$  do
3:    $s_i \leftarrow$  Remove peaks having a low intensity  $I$ 
4:    $s_i \leftarrow$  Only keep the  $\max_P$  most intense peaks
5:   for all  $P_{i,j} \in s_i$  do
6:      $w(P_{i,j}) = 1$   $\triangleright$  Initialize the peak weight
7:      $S(P_{i,j}) = \{s_i\}$   $\triangleright$  Remember the spectrum the peak comes from
8:      $s_{\text{cons}} \leftarrow s_{\text{cons}} \cup \{P_{i,j}\}$ 
9:   end for
10: end for
11: while True do
12:    $\{P_a, P_b\} = \{(m/z_a, I_a), (m/z_b, I_b)\} \leftarrow$  Find the pair of spectra that minimizes
    $|m/z_b - m/z_a|$ 
13:   if  $|m/z_b - m/z_a| > \text{tol}_{m/z}$  then
14:     break
15:   end if
16:    $m/z_{\text{new}} \leftarrow \frac{w(P_a) * m/z_a + w(P_b) * m/z_b}{w(P_a) + w(P_b)}$   $\triangleright$  Define a new peak
17:    $I_{\text{new}} \leftarrow I_a + I_b$ 
18:    $P_{\text{new}} \leftarrow (m/z_{\text{new}}, I_{\text{new}})$ 
19:   Remove  $P_a$  and  $P_b$  from  $s_{\text{cons}}$ 
20:    $w(P_{\text{new}}) \leftarrow w(P_a) + w(P_b)$ 
21:    $S(P_{\text{new}}) \leftarrow S(P_a) \cup S(P_b)$ 
22:    $s_{\text{cons}} \leftarrow s_{\text{cons}} \cup P_{\text{new}}$   $\triangleright$  Add the new peak to the consensus spectrum
23: end while
24: for all  $P = (m/z, I) \in s_{\text{cons}}$  do
25:    $\text{prob} \leftarrow \frac{|S(P)|}{|C|}$   $\triangleright$  The probability associated to the peak
26:    $I \leftarrow I * (0.95 + 0.05 * (1 + \text{prob})^5)$   $\triangleright$  Emphasize the intensity of peaks
   present in many spectra
27: end for
28: return  $s_{\text{cons}}$ 

```

Prot ID	Tissue	Protein name	UniProt description
P25189	Spinal cord	Myelin protein P0	Is an adhesion molecule necessary for normal myelination in the peripheral nervous system . [...]
P12883	Heart	Myosin-7	[...] together with actin thin filaments, constitute the fundamental contractile unit of skeletal and cardiac muscle .
Q99962	Frontal cortex	Endophilin-A1	Implicated in synaptic ⁵ vesicle endocytosis. [...]
Q9UPA5	Frontal cortex	Protein bassoon	Scaffold protein of the presynaptic cytomatrix at the active zone (CAZ) which is the place in the synapse where neurotransmitter is released. [...]
Q8WZ42	Heart	Titin	Key component in the assembly and functioning of vertebrate striated muscles . [...]
Q8IWL2	Lung	Pulmonary surfactant-associated protein A1	[...] contributes to lower the surface tension at the air-liquid interface in the alveoli of the mammalian lung and is essential for normal respiration. [...]
P12271	Retina	Retinaldehyde-binding protein 1	Soluble retinoid carrier essential the proper function of both rod and cone photoreceptors . [...]
P08100	Retina	Rhodopsin	Photoreceptor required for image-forming vision at low light intensity. [...]

Table 4.2: The consensus spectra analysis of the selected unidentified clusters could identify many peptide sequences, and the corresponding proteins. This table shows several examples for which the link between the identified protein and the tissue used to generate the spectra is obvious.

	Batch 1	Batch 2	Batch 3
Oxidation	1	26	33
Glu ->pyro-Glu		7	3
Gln ->pyro-Glu		3	3
Deamidation		4	2
Acetylation		4	2
Carbamidomethylation		1	8
Methylation		1	4
Phosphorylation		1	5
No PTM	1	5	1

Table 4.3: Occurrences of the PTMs among the identifications of the consensus spectra. The values represent the number of consensus spectra in which the PTM was detected at least once. For example, in batch 2, at least one acetylation was detected on four spectra.

Chapter 5

Areas for improvement

In Chapter 3, we presented the algorithms used in FALCON. We explained why FALCON shows a (desired) linear time complexity. Then, in Chapter 4, we clustered a large dataset. We were able to identify the sequences of many unidentified clusters. This chapter presents and analyzes several areas for improvement to determine if they seem to be effective and how they affect performances.

5.1 Dimension reduction

In this section, we compare the dimension reduction method used by FALCON, the *hashing trick* [28] (see Chapter 3), with two other linear dimension reduction methods: PCA, which uses singular value decomposition of the data, and Gaussian random projection which consists in projecting the high dimensional vectors on a matrix whose components are generated from a Gaussian distribution [38]. For each method, we reduced the high dimensional data to 800 components, the parameter used by default in FALCON. Then, we computed the cosine distance between pairs of high dimensional vectors $\mathbf{x}$ (corresponding to 5000 spectra taken in the same bucket), and between the same pairs of low dimensional vectors. Ideally, this distance should be preserved as much as possible. Figure 5.1 shows the results ($\text{dist}_\theta(\mathbf{x}_1, \mathbf{x}_2)$ versus $\text{dist}_\theta(\mathbf{x}'_1, \mathbf{x}'_2)$). All these methods tend to better preserve the distances between close pairs of vectors than distant spectra. For this number of components, the Gaussian projection offers the best precision: the mean squared error (MSE) between the high dimensional cosine distances and the corresponding low dimensional distances is the smallest. However, the average execution time for the hashing trick is 0.48s on average, while the execution time for the Gaussian random projection is 3.53s on average. PCA offers both the worst MSE and the longest execution time (41.45s on average).

Now, we investigate how the MSE evolves when reducing the number of components (Table 5.1). An interesting result is that Gaussian random projection can achieve a better dimension reduction (200 to 400 components versus 800 for the hashing trick) for a similar MSE. For 200 components, the Gaussian random projection requires on average 1.10s, which is about two times more than the hashing trick for 800 components. However, reducing the number of features from 800 to 200 should improve the speed of the nearest neighbors search used in FALCON. In

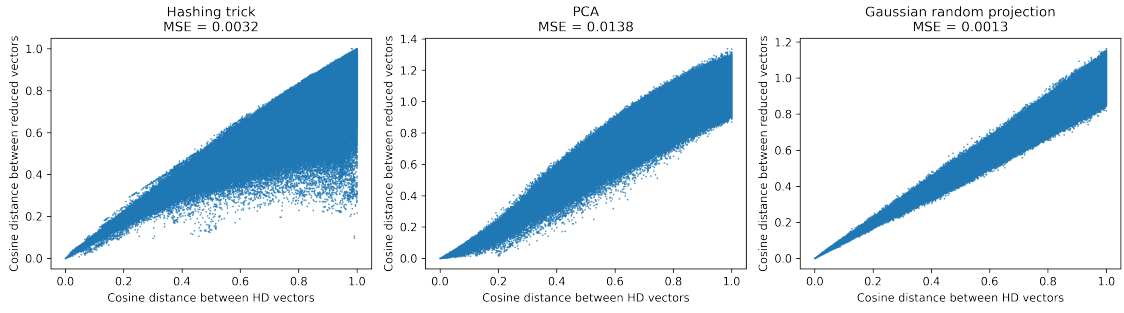


Figure 5.1: Comparison of three-dimensional reduction methods. Ideally, the distances should be preserved, and the points plotted on these figures should be as close as possible to the diagonal. Note that for PCA and the Gaussian random projection, distances between pairs of spectra can become larger than 1: the cosine between the vector can become negative because negative components can appear, which was not the case with the hashing trick.

conclusion, the use of Gaussian random projection instead of the hashing trick should be considered and tested in further work, given that it achieves more significant dimension reduction for similar precision.

# components	Hashing trick	PCA	Gaussian
800	0.0032	0.0138	0.0013
400	0.0069	0.0147	0.0024
200	0.0163	0.0169	0.0045
100	0.0379	0.022	0.0090
50	0.092	0.033	0.019

Table 5.1: Mean squared errors of the three dimensionality methods for different numbers of components.

5.2 DBSCAN concurrents

Since FALCON provides a way to convert spectra to vectors of reasonable dimension, any clustering algorithm could be used to cluster them. In this section, we compare three well-known algorithms on a subset of 10 000 spectra, to observe if DBSCAN offers the best results in terms of clustering quality.

5.2.1 Algorithms

We compare the following clustering algorithms:

- **DBSCAN**. This algorithm (used in FALCON) was presented in Pseudocode 4. Here, we computed the distance matrix exhaustively (to avoid the potential error that an approximate distance matrix, such as the one used in FALCON, could introduce).

- **k-means**, with centroids initialized on input vectors selected randomly. This algorithm has already been presented in Chapter 3. The number of centroids (which corresponds to the number of clusters) is the main parameter. The clustering ends when the solution has converged or the maximal number of iterations has been reached (at most, 300 iterations are allowed).
- **Agglomerative** (hierarchical) clustering. Initially, each spectrum is assigned to one cluster (there are as many clusters as spectra). At each iteration, the pair of clusters separated by the smallest distance is merged. The user can specify a distance threshold d_{thr} to determine when to stop the algorithm (when all pairs of clusters are separated by a distance larger than d_{thr}). Several methods exist to compute the distance between two clusters. We use **average-linkage**: the distance between two clusters C_A and C_B is defined as the average of the distances between all the different pairs of clusters.

We use the implementations provided by the *sklearn* Python library.

5.2.2 Methodology

The three clustering algorithms are very different. To compare them fairly, we used the following method.

Vectors

We used the *hashing trick*, the method provided in FALCON and described in Chapter 3, to convert the spectra to vectors of size 800, as is the case in FALCON. Not using the dimension reduction would have resulted in much longer execution times. As we have seen previously, this dimension reduction method preserves pretty well the distances for relatively close pairs of vectors.

Distance function

FALCON uses the cosine distance as a distance function. DBSCAN and agglomerative clustering can be used with any distance metric. However, k-means is typically used with euclidean distances (since for this metric, it can be mathematically proven the objective clustering function will converge to a minimum¹). This is not a problem as long as the vectors are normed: in this case, the relation between the cosine and the Euclidean distances simplifies to

$$\text{dist}_{\text{eucl}}(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\text{dist}_{\theta}(\mathbf{x}_1, \mathbf{x}_2)}$$

It implies that the relation $\text{dist}_{\text{eucl}}(\mathbf{x}_1, \mathbf{x}_2) < \text{dist}_{\text{eucl}}(\mathbf{x}_3, \mathbf{x}_4) \Rightarrow \text{dist}_{\theta}(\mathbf{x}_1, \mathbf{x}_2) < \text{dist}_{\theta}(\mathbf{x}_3, \mathbf{x}_4)$ is always respected. Thus, it is not a problem to use the cosine distance for DBSCAN and the agglomerative hierarchical clustering, and euclidean distance for k-means.

¹It can be a local minimum.

Constraints

The **precursor charge** constraint can easily be taken into account by only clustering spectra of the same charge (as it is the case in FALCON). Clustering a subset of spectra that respect the **precursor mass** constraint turns out to be more complicated since the number of spectra would typically be a few thousand or less. During the experiments, we observed that such a size of subset is not sufficient to produce results that can be generalized to larger datasets.

For DBSCAN and the agglomerative hierarchical clustering, we could have taken into account the constraint similarly as in FALCON, by setting the distance between pairs of spectra violating the constraint to the maximal value ($=1$). For k-means, it is more complicated. Adaptations of k-means able to take into account *must-link* and *cannot-link* constraints have been released [39].

However, to keep the comparison as simple as possible, we decided to relax the precursor mass constraint for the comparison for the three algorithms. Indeed, the goal of the precursor mass constraint is mainly to reduce the search space. We expect this constraint to be naturally respected (at least, when the clustering parameters are chosen such that dissimilar spectra can be clustered separately), because close pairs of spectra tend to have close precursor m/z^2 (because they are likely to correspond to the same peptide).

Subset selection

Since we have relaxed the precursor mass constraint, the size of the search space has considerably increased. Thus, it is necessary to limit the number of compared spectra to make the clustering in a reasonable time. However, the number of clustered spectra needs to be large enough such that the results obtained on the subset can be generalized. When selecting subsets of 10 000 spectra or more, the results (presented below) tend to become very similar from one bucket to another. Thus, we used a subset of this size (the first 10 000 spectra of the bucket of charge 2 and $p_{m/z} \in [600, 601[$ of the Draft of the Human Proteome). Note that it is crucial to select spectra having a similar precursor m/z (e.g., spectra in the same bucket). By choosing a small subset of spectra randomly over the whole dataset, the probability of selecting several spectra corresponding to the same peptide (and having a similar precursor m/z) would be very low, and the chosen spectra would appear dissimilar to each other.

Noise and singleton clusters

The three algorithms compared are very different. DBSCAN does not put all the spectra in clusters: some points are tagged as noise (the number of which depends notably on the radius ϵ used as a parameter). They do not correspond to any cluster. By opposition, agglomerative hierarchical clustering and k-means assign all the points to clusters. It can be considered that the equivalent of *noise* points in DBSCAN are the points in clusters of size one in k-means and agglomerative clustering. Thus,

²More than 99% of the pairs of spectra at a cosine distance smaller than 0.1 respect the precursor mass constraint of 20 ppm.

we reassigned all the noise points of DBSCAN to singleton clusters (such that the completeness is penalized when DBSCAN only cluster a very small proportion of spectra).

Clusters visualization

Homogeneity and completeness are very useful metrics to assess if a clustering is coherent with the labels we have for part of the spectra. However, it can always be useful to visualize the data. To plot visualize the spatial distribution of the vectors in two dimensions, we used Barnes Hut t-SNE to perform the neighbor embedding and get 2-dimensional data. Prior to applying t-SNE, we used PCA to reduce the high-dimensional vectors ($\approx 30\,000$) to vectors of 50 dimensions. We used the implementation available in *sklearn*, with the parameters `n_components = 2`, `perplexity = 30`, `n_iter = 1000` and `learning_rate = 200`.

5.2.3 Comparison

The homogeneity and the completeness of the three clustering algorithms, as a function of their respective main parameter, are shown in Figure 5.2. For DBSCAN, the parameter $\epsilon = 0.1$ previously used for the clustering of the whole Draft of the Human Proteome provides a good trade off between a homogeneity close to one ($h = 0.97$) and a relatively good completeness ($c = 0.82$). Since a very good homogeneity is critical when clustering mass-spectrometry data (as discussed in Chapter 4), we compare the three algorithms at a similar homogeneity score. For k-means, it requires using a very large number of clusters (2500, for the 10 000 clustered spectra) to get a similar homogeneity to DBSCAN ($h = 0.97$), which leads to a decrease of the completeness to $c = 0.71$. For the agglomerative hierarchical clustering, we use $d_{\text{thr}} = 0.25$, which results in a homogeneity $h = 0.97$ and a completeness $c = 0.85$ (slightly better but comparable to DBSCAN).

Now that we have selected suitable parameters for each algorithm, we can visualize the corresponding clusterings on Figure 5.3.

Reference plot

All the subplots in Figure 5.3) shows the result of the neighbor embedding performed by the Barnes Hut t-SNE, which defines the position of the vectors in the 2-dimensional space. In the top left subplot (that we will the reference plot), the colors correspond to the labels. It shows there is a strong correlation between the sequences (the colors) and the clusters of points visible on the figure. This indicates that the selected parameters for t-SNE seem to give the expected result (close pairs of high-dimensional vectors are close in the 2-dimensional space). Many clusters contain no labeled spectrum (all the points are in grey). The clusters are well delimited on the periphery. Still, the center of the plot seems to contain a poorly defined cluster containing a majority of unlabeled spectra (grey points). This large cluster is an artifact caused by t-SNE and containing points that have no or very few close neighbors.

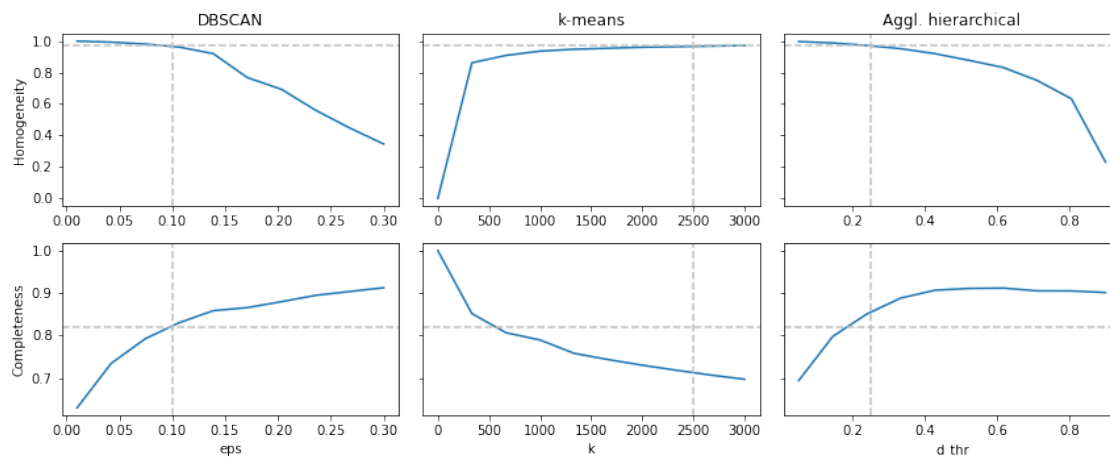


Figure 5.2: Comparison of the homogeneity and completeness scores of the three clustering algorithms (DBSCAN, k-means and agglomerative hierarchical clustering). The vertical grey lines correspond to the parameter used to get a homogeneity $h = 0.97$, and the horizontal grey lines correspond to the homogeneity and completeness scores of DBSCAN (for $\epsilon = 0.1$).

DBSCAN

In Figure 5.3, many similarities can be seen between the reference plot (top left) and the clustering produced by DBSCAN. In other words, DBSCAN tends to assign the spectra having the same label to the same clusters. The main particularity of DBSCAN, compared to the other compared algorithms, is that it tags most of the points belonging to the central cluster as noise, while the other algorithms tend to cluster a larger proportion of these points.

k-means

To get the expected homogeneity, it is necessary to use a large number of clusters k , which explains that the completeness of k-means is low compared to the other algorithms: all the large clusters are split into several clusters. Thus, the homogeneity is very low (visually, it can be observed in Figure 5.3 that contrary to the reference plot, there are points with many different colors in each cluster generated by t-SNE). Given that there is a large number of clusters, the average number of spectra per cluster tends to be low, which is problematic (the results we got in Chapter 4 tend to indicate that large clusters are more likely to be identified).

In Figure 5.2, it can be seen that $k = 2500$ does not provide the best trade-off between homogeneity and completeness, even though it provides a high homogeneity desired for the comparison. k-means is typically used with a number of centroids k significantly smaller than the number of samples. The parameter $k = 400$ gives a better overview of the k-means capabilities (see Figure 5.4). However, it still splits many large clusters into several smaller clusters, including clusters in which all spectra have the same label (this can be seen by comparing this plot to the reference plot in Figure 5.3). Selecting $k < 400$ results in splitting fewer clusters, but it also decreases the homogeneity.

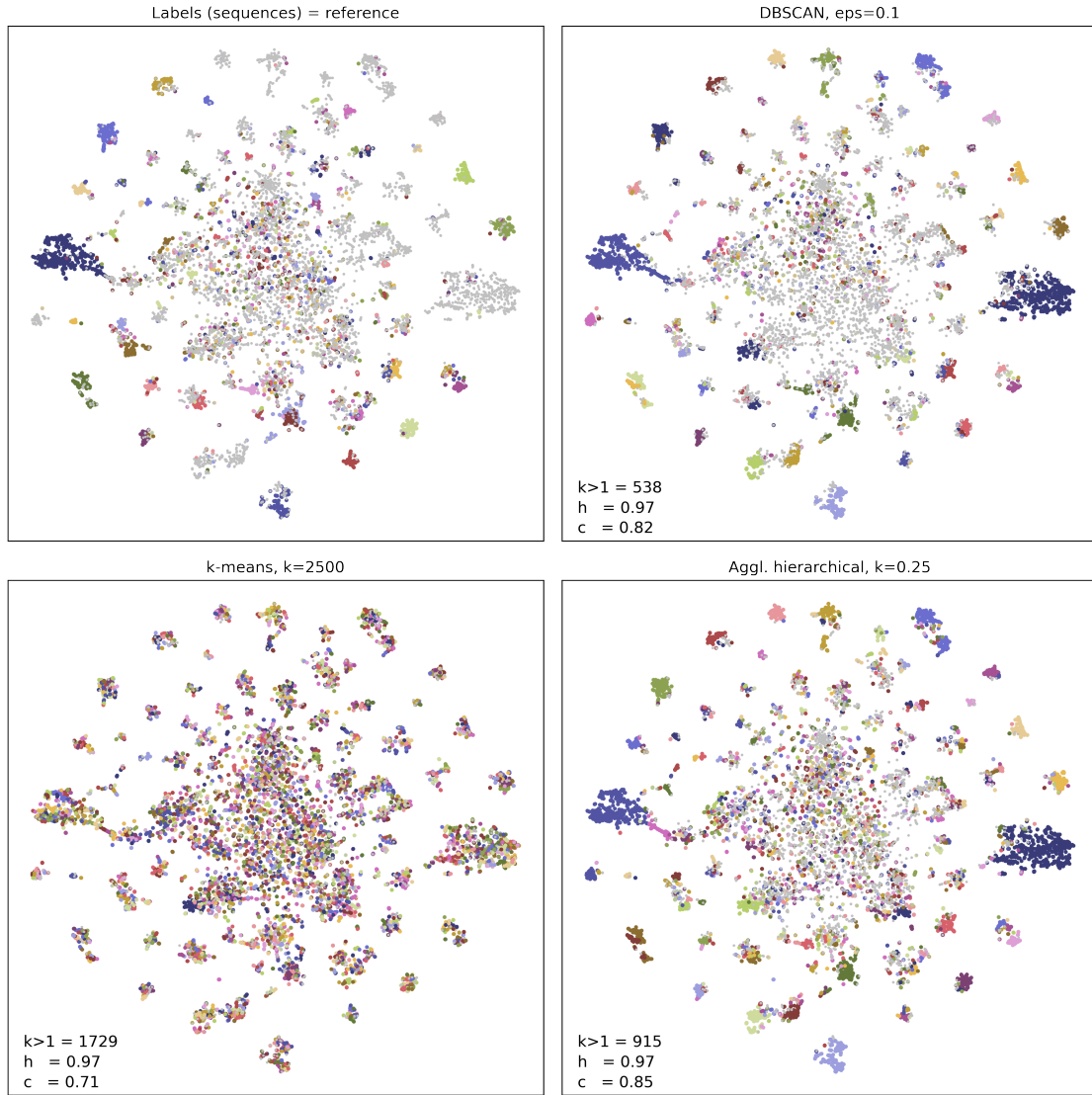


Figure 5.3: By using PCA to reduce the number of components of the high-dimensional vectors to 50, and then Barnes Hut t-SNE for the neighbor embeddings, it becomes possible to visualize the 10000 vectors and the clusters they form in the 2-dimensional space. The four subplots show the 2-dimensional points obtained with t-SNE. The top-left subplot shows the labels (one color per label, unlabeled spectra in grey). It shows that each cluster resulting from the application of t-SNE tends to contain spectra with the same label, which is expected (spectra having the same label tend to be very similar). For the three subplots showing the result of the clustering algorithms, each point is colored according to the cluster it was assigned to. Grey points correspond to the noise (DBSCAN) and the clusters containing only one element (k-means and agglomerative hierarchical clustering). Note that t-SNE was only used for the visualization, the clustering was done on vectors generated by the dimension reduction method used in FALCON. In the bottom-left corner of each subplot are indicated the number of clusters containing at least one element ($k>1$), the homogeneity h and the completeness c .

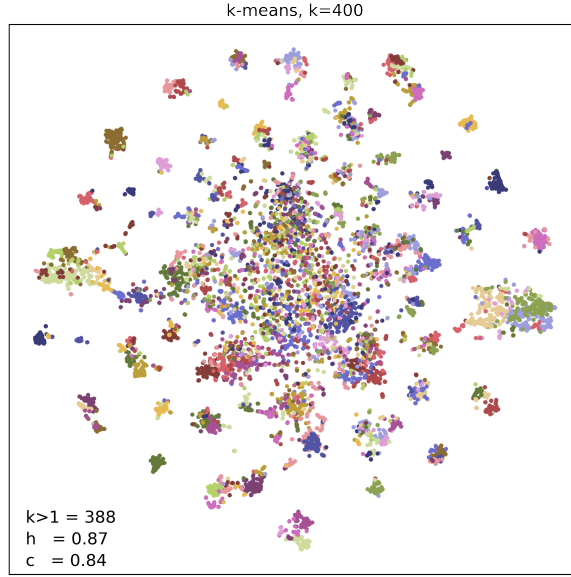


Figure 5.4: The parameter $k = 400$, even if it provides a lower homogeneity than $k = 2500$, results in a better trade-off between homogeneity and completeness for k-means.

Agglomerative hierarchical clustering

For the same homogeneity as DBSCAN ($h = 0.97$), hierarchical clustering provides a marginally better completeness than DBSCAN ($c = 0.85$ versus 0.82) for this subset. This is surprising, since hierarchical clustering results in many more clusters than DBSCAN (see Table 5.2): intuitively, we could expect a lower completeness, since the large number of clusters could indicate that clusters are split. As for DBSCAN, many similarities can be observed with the reference plot.

	DBSCAN	Aggl. hierarchical
h	0.97	0.97
c	0.82	0.85
# clusters > 1	538	915
Time (s)	0.97 ± 0.06	30.79 ± 0.22

Table 5.2: Comparison between DBSCAN and hierarchical clustering. '# clusters > 1' corresponds to the number of clusters containing at least two spectra. The time measures were repeated three times and are shown with the associated standard deviation (the characteristics of the device used for the measurement are indicated in Appendix A).

Visually (see Figure 5.3), the main difference with DBSCAN is that many more points belonging to the central and poorly defined cluster tend to be grouped in small clusters. Given that most of the spectra belonging to this cluster do not have a label, the clusters they are assigned to does not influence the homogeneity and completeness scores. Thus, we can hypothesize that the larger number of clusters compared to DBSCAN is because hierarchical clustering forms many more small clusters containing

unlabeled spectra. Figure 5.5 confirms that the more important number of clusters achieved by the hierarchical clustering is mainly due to small clusters of size smaller than 20 (and particularly, smaller than 10).

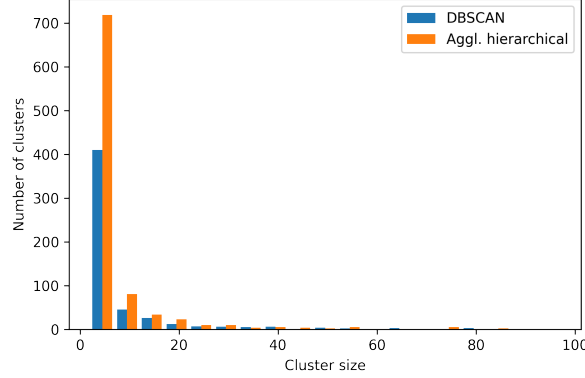


Figure 5.5: Agglomerative hierarchical clustering produces many more small clusters.

Even though hierarchical clustering achieves better completeness than DBSCAN, its main drawback is its performance. At each iteration, it finds the best pair of clusters (time complexity $O(n^2)$). There are at most n iterations (where n is the number of samples to cluster): the resulting time complexity is $O(n^3)$ with a naive implementation (by using a priority queue, it is possible to reduce the time complexity to $O(n^2 * \log(n))$ [40]). This complexity implies that hierarchical clustering is not scalable for large datasets. Table 5.2 shows that for our subset of 10 000 spectra, hierarchical clustering takes 30x more time than DBSCAN.

5.2.4 Discussion

This comparison between three widely used clustering algorithms made possible to better assess their properties and their performances on a subset of spectra. Since our application of clustering requires a high homogeneity, k-means is not a good candidate, because achieving very high homogeneity requires sacrificing completeness: many small clusters are formed. In this subset, for the same homogeneity score, hierarchical clustering achieves better completeness than DBSCAN. However, the difference is marginal and we cannot conclude that agglomerative hierarchical clustering offers a better completeness without further work.

Even though hierarchical clustering achieves better completeness than DBSCAN (at the same homogeneity score), its time complexity makes it intractable for large datasets. In conclusion, among the three compared algorithms, DBSCAN seems to present the best trade-off between the homogeneity and completeness scores and the performances.

5.3 Avoiding the postprocessing step

In the original implementation of FALCON, a postprocessing step is required to split the clusters violating the precursor mass constraint by transitive closure. That is

not very elegant. Other versions of DBSCAN could take this constraint into account directly. For example:

- C-DBSCAN, by Ruiz et al. [41], supports the use of cannot link constraints: we could apply such constraints to all the pairs of spectra that do not respect the precursor mass constraint.
- ST-DBSCAN, by Birant et al. [42], is based on DBSCAN but can cluster points based on two distances, the spatial distance, and the non-spatial distance. One ϵ parameter has to be defined for each of these distances. If their algorithm was used in FALCON, the precursor masses difference of two spectra could be used as non-spatial distance ($\epsilon_{\text{non_spatial}} \propto \text{tol}^{p_m}$). No postprocessing step would be required because all the clusters would respect the precursor mass constraint. According to the authors, the time complexity of ST-DBSCAN is the same as for DBSCAN.

Chapter 6

Conclusion

First, we presented the interest in clustering datasets generated during shotgun proteomics experiments in Chapter 1. We justified the need to make use of efficient algorithms by the ever-increasing available data.

In Chapter 3, we presented FALCON, an efficient software for clustering tandem mass spectra. We explained why in practice, the time complexity of FALCON depends on the order of magnitude of the dataset size. For datasets of several tens of million spectra, for which the number of spectra per bucket is inferior to 10^5 , the execution time of FALCON grows linearly as the size of the dataset increases. For larger datasets (e.g., hundreds/thousands of millions of spectra), we can predict that the time complexity will be $O(n^2)$. However, this complexity is attenuated as the order of magnitude of the dataset size increases by increasing the number of Voronoï cells used for the approximate neighbors search.

In Chapter 4, we reanalyzed many unidentified clusters generated by FALCON after clustering the Draft of the Human Proteome (presented in Chapter 2). We were able to identify the sequences corresponding to many unidentified clusters. We observed a link between many proteins and the tissue from which the spectra were acquired. We observed that larger clusters were more likely to be identified.

Finally, we analyzed several areas of improvements for FALCON in Chapter 5. We showed that Gaussian random projections could achieve further dimension reduction of the vectors representing the spectra while preserving the distances. The comparison between three clustering algorithms on a subset of spectra showed that DBSCAN performs as well as or better than its competitors, and it is faster.

The interest in clustering data acquired during shotgun proteomics experiments is not limited to the possibility of reanalyzing clusters generated from a single dataset. The ability, provided by FALCON, to analyze datasets containing tens of millions of spectra in reasonable time and without requiring a supercomputer or cloud computing makes clustering of proteomics data much more accessible. Thus, we think that opens the door to many new applications of clustering. For example, by clustering the Draft of the Human Proteome with a dataset acquired from cancer cells and then analyzing large clusters containing only spectra obtained from the cancer cells, it could be possible to identify peptide variants only expressed in cancer cells, which would have a significant biological interest.

Appendices

Appendix A

Reproducibility

A.1 Setup

Unless it is mentioned otherwise, the time measurements were made on an ASUS UX410UQK laptop with 16 GB of RAM, an Intel Core i7-7500U (2.70 GHz) CPU, and an NVIDIA GeForce 940MX graphic card. The OS is Ubuntu 20.04.2 LTS (64-bit). The spectra' buckets were stored on and read from an external HDD of 4 TB (ext4 partition, 5400 rpm) connected to a 3.0 USB port. Note that in order to cluster the entire Draft of the Human Proteome dataset, we used a 64GB swapfile stored on the SSD (256GB), because the main memory was not sufficient.

A.2 Draft of the Human Proteome

The files containing the spectra were downloaded from the PRIDE Archive [18]¹ (ID: MSV000079514). Similarly to Bittremieux et al. [23], we used ThermoRawFileParser [43] to convert the RAW files to MGF files that FALCON can handle. We also used the same identification data, from the MassIVE Reanalysis platform².

A.3 Source code

The code used to generate the results presented in this document is written in Python and available in the Jupyter notebooks available in the *notebooks* folder of the following repository <https://github.com/KilianMaes/falcon/>, which is a fork of the repository that stores the FALCON clustering tool (by Bittremieux et al. [23]). The version of FALCON used for the algorithmic description (Chapter 3 and the clustering (Chapter 4) corresponds to the commit 5b6acc5282856efa91b325ee6fa51a31abae9f66 of FALCON (March 13, 2021).

As it is indicated in the file, `bittremieux_utils.py` reuses several function of one notebook written by Wout Bittremieux and available on its repository `falcon_notebooks`³.

¹<https://www.ebi.ac.uk/pride/archive/projects/PXD000561>

²<https://massive.ucsd.edu/ProteoSAFe>

³https://github.com/bittremieux/falcon_notebooks

Other files available on the *notebooks* folder are:

- `analyzePXD000561spectra.ipynb`: generate the figures presented in Chapter 2 (the number of spectra, the precursor m/z distribution, etc).
- `analyzeFalconAlgorithm.ipynb`: generate the figures of the predicted and measured execution time for the construction of the distance matrix presented in Chapter 3.
- `analyzeFalconCluster.ipynb`: used to compare the homogeneity and the completeness of FALCON for several ϵ parameters (see Chapter 4).
- `clusteridentification.ipynb`: determine the number of identified cluster, unidentified clusters, etc. This file was used to generate plots presented in Chapter 4.
- `consensusReanalyze.ipynb`: analyze the identification results of the consensus spectra reanalysis, and fetch the protein descriptions from UniProt.
- `compareClusteringAlgorithms_v2.ipynb`: make the comparison between several clustering algorithms on a subset of spectra in Chapter 5. This file was also used to make the comparison between the dimension reduction methods presented in the same chapter.

Most of these files use "imports" of the FALCON source code (e.g., to read the spectra in a particular file, we make an import of the corresponding function in FALCON).

A.4 Parameters used for the clustering

To obtain the results presented in Chapter 4, FALCON was run with the following configuration (extracted from `config.py`):

```
mz_interval = 1
charges, mz_s = (2, 3, 4, 5), np.arange(50, 2501, mz_interval)

# Spectrum preprocessing.
min_peaks = 5
min_mz_range = 250.
min_mz, max_mz = 101., 1500.
remove_precursor_tolerance = 0.5
min_intensity = 0.01
max_peaks_used = 50
scaling = None

# Spectrum to vector conversion.
fragment_mz_tolerance = 0.05
hash_len = 800
```

```
# Spectrum matching.
precursor_tol_mass, precursor_tol_mode = 20, 'ppm'

# NN index construction and querying.
n_neighbors, n_neighbors_ann = 64, 128
n_probe = 32
batch_size = 2*16

# DBSCAN clustering.
eps = 0.1
min_samples = 2
```

A.5 Libraries

The computations and other plots shown in this thesis were realized with Python 3.8.6, with the following libraries:

- Matplotlib 3.3.3
- Numpy 1.19.4
- Pandas 1.2.0
- Scikit-learn 0.24.0
- Scipy 1.5.3
- Spectrum_utils 0.3.4

Bibliography

- [1] J Craig Venter, Mark D Adams, Eugene W Myers, Peter W Li, Richard J Mural, Granger G Sutton, Hamilton O Smith, Mark Yandell, Cheryl A Evans, Robert A Holt, et al. The sequence of the human genome. *science*, 291(5507):1304–1351, 2001.
- [2] Elena A Ponomarenko, Ekaterina V Poverennaya, Ekaterina V Ilgisonis, Mikhail A Pyatnitskiy, Arthur T Kopylov, Victor G Zgoda, Andrey V Lisitsa, and Alexander I Archakov. The size of the human proteome: the width and depth. *International journal of analytical chemistry*, 2016, 2016.
- [3] Alexey I Nesvizhskii, Olga Vitek, and Ruedi Aebersold. Analysis and validation of proteomic data generated by tandem mass spectrometry. *Nature methods*, 4(10):787–797, 2007.
- [4] Joshua J Coon, John EP Syka, Jeffrey Shabanowitz, and Donald F Hunt. Tandem mass spectrometry for peptide and protein sequence analysis. *Biotechniques*, 38(4):519–523, 2005.
- [5] Johannes Griss, Yasset Perez-Riverol, Steve Lewis, David L Tabb, José A Dianes, Noemi Del-Toro, Marc Rurik, Mathias Walzer, Oliver Kohlbacher, Henning Hermjakob, et al. Recognizing millions of consistently unidentified spectra across hundreds of shotgun proteomics datasets. *Nature methods*, 13(8):651–656, 2016.
- [6] Jason Gallia, Katelyn Lavrich, Anna Tan-Wilson, and Patrick H Madden. Filtering of ms/ms data for peptide identification. *BMC genomics*, 14(7):1–9, 2013.
- [7] Joel M Chick, Deepak Kolippakkam, David P Nusinow, Bo Zhai, Ramin Rad, Edward L Huttlin, and Steven P Gygi. A mass-tolerant database search identifies a large proportion of unassigned spectra in shotgun proteomics as modified peptides. *Nature biotechnology*, 33(7):743–749, 2015.
- [8] David L Tabb, Michael J MacCoss, Christine C Wu, Scott D Anderson, and John R Yates. Similarity among tandem mass spectra from proteomic experiments: detection, significance, and utility. *Analytical chemistry*, 75(10):2470–2477, 2003.
- [9] Ilan Beer, Eilon Barnea, Tamar Ziv, and Arie Admon. Improving large-scale proteomics by clustering of mass spectrometry data. *Proteomics*, 4(4):950–960, 2004.

- [10] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–297, 1999.
- [11] David L Tabb, Melissa R Thompson, Gurusahai Khalsa-Moyers, Nathan C VerBerkmoes, and W Hayes McDonald. Ms2grouper: group assessment and synthetic replacement of duplicate proteomic tandem mass spectra. *Journal of the American Society for Mass Spectrometry*, 16(8):1250–1261, 2005.
- [12] HC Johnston. Cliques of a graph-variations on the bron-kerbosch algorithm. *International Journal of Computer & Information Sciences*, 5(3):209–238, 1976.
- [13] Kristian Flikka, Jeroen Meukens, Kenny Helsens, Joël Vandekerckhove, Ingvar Eidhammer, Kris Gevaert, and Lennart Martens. Implementation and application of a versatile clustering tool for tandem mass spectrometry data. *Proteomics*, 7(18):3245–3258, 2007.
- [14] Ari M Frank, Nuno Bandeira, Zhouxin Shen, Stephen Tanner, Steven P Briggs, Richard D Smith, and Pavel A Pevzner. Clustering millions of tandem mass spectra. *Journal of proteome research*, 7(01):113–122, 2008.
- [15] Fahad Saeed, Trairak Pisitkun, Mark A Knepper, and Jason D Hoffert. An efficient algorithm for clustering of large-scale mass spectrometry data. In *2012 IEEE International Conference on Bioinformatics and Biomedicine*, pages 1–4. IEEE, 2012.
- [16] Fahad Saeed, Jason D Hoffert, and Mark A Knepper. Cams-rs: clustering algorithm for large-scale mass spectrometry data using restricted search space and intelligent random sampling. *IEEE/ACM transactions on computational biology and bioinformatics*, 11(1):128–141, 2013.
- [17] Johannes Griss, Joseph M Foster, Henning Hermjakob, and Juan Antonio Vizcaíno. Pride cluster: building a consensus of proteomics data. *Nature methods*, 10(2):95–96, 2013.
- [18] Juan Antonio Vizcaíno, Attila Csordas, Noemi Del-Toro, José A Dianes, Johannes Griss, Ilias Lavidas, Gerhard Mayer, Yasset Perez-Riverol, Florian Reisinger, Tobias Ternent, et al. 2016 update of the pride database and its related tools. *Nucleic acids research*, 44(D1):D447–D456, 2016.
- [19] Matthew The and Lukas Käll. Maracluster: A fragment rarity metric for clustering fragment spectra in shotgun proteomics. *Journal of proteome research*, 15(3):713–720, 2016.
- [20] Debojyoti Dutta and Ting Chen. Speeding up tandem mass spectrometry database search: metric embeddings and fast near neighbor search. *Bioinformatics*, 23(5):612–618, 2007.
- [21] Lei Wang, Sujun Li, and Haixu Tang. mscrush: fast tandem mass spectral clustering using locality sensitive hashing. *Journal of proteome research*, 18(1):147–158, 2018.

- [22] Yasset Perez-Riverol, Juan Antonio Vizcaíno, and Johannes Griss. Future prospects of spectral clustering approaches in proteomics. *Proteomics*, 18(14):1700454, 2018.
- [23] Wout Bittremieux, Kris Laukens, William Stafford Noble, and Pieter C Dorrestein. Large-scale tandem mass spectrum clustering using fast nearest neighbor searching. *bioRxiv*, 2021.
- [24] Min-Sik Kim, Sneha M Pinto, Derese Getnet, Raja Sekhar Nirujogi, Srikanth S Manda, Raghothama Chaerkady, Anil K Madugundu, Dhanashree S Kelkar, Ruth Isserlin, Shobhit Jain, et al. A draft map of the human proteome. *Nature*, 509(7502):575–581, 2014.
- [25] Mathias Wilhelm, Judith Schlegl, Hannes Hahne, Amin Moghaddas Gholami, Marcus Lieberenz, Mikhail M Savitski, Emanuel Ziegler, Lars Butzmann, Siegfried Gessulat, Harald Marx, et al. Mass-spectrometry-based draft of the human proteome. *Nature*, 509(7502):582–587, 2014.
- [26] Sangtae Kim and Pavel A Pevzner. Ms-gf+ makes progress towards a universal database search tool for proteomics. *Nature communications*, 5(1):1–10, 2014.
- [27] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008.
- [28] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 1113–1120, 2009.
- [29] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.
- [30] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.
- [31] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.
- [32] Wout Bittremieux. spectrum_utils: A python package for mass spectrometry data processing and visualization. *Analytical chemistry*, 92(1):659–661, 2019.
- [33] Yue Qin, Shifei Ding, Lijuan Wang, and Yanru Wang. Research progress on semi-supervised clustering. *Cognitive Computation*, 11(5):599–612, 2019.
- [34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

- [35] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.
- [36] UniProt Consortium. Uniprot: a hub for protein information. *Nucleic acids research*, 43(D1):D204–D212, 2015.
- [37] Vincent G Allfrey, R Faulkner, and AE Mirsky. Acetylation and methylation of histones and their possible role in the regulation of rna synthesis. *Proceedings of the National Academy of Sciences of the United States of America*, 51(5):786, 1964.
- [38] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, 2001.
- [39] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schroedl, et al. Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584, 2001.
- [40] William HE Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.
- [41] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan: Density-based clustering with constraints. In *International workshop on rough sets, fuzzy sets, data mining, and granular-soft computing*, pages 216–223. Springer, 2007.
- [42] Derya Birant and Alp Kut. St-dbscan: An algorithm for clustering spatial-temporal data. *Data & knowledge engineering*, 60(1):208–221, 2007.
- [43] Harald Barsnes, Marc Vaudel, Niklaas Colaert, Kenny Helsens, Albert Sickmann, Frode S Berven, and Lennart Martens. compomics-utilities: an open-source java library for computational proteomics. *BMC bioinformatics*, 12(1):1–6, 2011.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl