Large-scale tandem mass spectrum clustering using fast nearest neighbor searching

Wout Bittremieux[1,2,*], **Kris Laukens**[2], **William Stafford Noble**[3], **Pieter C. Dorrestein**[1]

[1]Skaggs School of Pharmacy and Pharmaceutical Sciences, University of California San Diego, La Jolla, CA 92093, USA; [2]Department of Computer Science, University of Antwerp, 2020 Antwerp, Belgium; [3]Department of Genome Sciences, University of Washington, Seattle, WA 98195, USA.

Corresponding author: wbittremieux@health.ucsd.edu

Abstract

Rationale: Advanced algorithmic solutions are necessary to process the ever increasing amounts of mass spectrometry data that is being generated. Here we describe the *falcon* spectrum clustering tool for efficient clustering of millions of MS/MS spectra.
Methods: *falcon* succeeds in efficiently clustering large amounts of mass spectral data using advanced techniques for fast spectrum similarity searching. First, high-resolution spectra are binned and converted to low-dimensional vectors using feature hashing. Next, the spectrum vectors are used to construct nearest neighbor indexes for fast similarity searching. The nearest neighbor indexes are used to efficiently compute a sparse pairwise distance matrix without having to exhaustively compare all spectra to each other. Finally, density-based clustering is performed to group similar spectra into clusters.
Results: Using a large draft human proteome dataset consisting of 25 million spectra, *falcon* is able to generate clusters of a similar quality as MS-Cluster and spectra-cluster, two widely used clustering tools, while being considerably faster. Notably, at comparable cluster quality levels, *falcon* generates larger clusters than alternative tools, leading to a larger reduction in data volume without the loss of relevant information for more efficient downstream processing.
Conclusions: *falcon* is a highly efficient spectrum clustering tool. It is publicly available as open source under the permissive BSD license at `https://github.com/bittremieux/falcon`.

1 Introduction

To obtain a comprehensive view of an organism's proteome, modern shotgun proteomics experiments generate thousands [21, 45] to tens of thousands [18, 19] of tandem mass spectrometry (MS/MS) runs, with tens of thousands of MS/MS spectra acquired during each individual run. Besides the significant efforts to acquire such large amounts of spectral data, processing these large data volumes poses a computational challenge that requires efficient algorithmic solutions.

Typically, the MS/MS spectra are processed using a sequence database search engine to derive their peptide and protein identities [34]. Alternatively, rather than having to search all of the raw spectra, as a preprocessing step spectrum clustering can be used to reduce the data volume [9, 12, 13, 37, 39]. Spectrum clustering groups highly similar spectra, after which each cluster can be represented by a single consensus spectrum. In this fashion a data reduction can be achieved because only the cluster representatives need to be processed. Additionally, because consensus spectra can have a higher signal-to-noise ratio than the raw spectra and because low-quality, unclustered spectra can be filtered out, the clustering approach can boost the sensitivity of the subsequent identification procedure. Furthermore, repository-scale clustering can be used to automatically generate comprehensive and high-quality spectral libraries in a data-driven fashion without having to rely on synthetic samples [12, 41], and the clustering results can be analyzed to gain deeper insights into the nature of repeatedly observed yet unidentified spectra [13].

Several spectrum clustering tools have been introduced, including MS-Cluster [9], spectra-cluster [12, 13], MaRaCluster [37], and msCRUSH [39]. In general, a clustering algorithm consists of several components: (i) a similarity measure to perform pairwise spectrum comparisons, (ii) a clustering method to group similar spectra, and (iii) optional optimizations to improve its computational efficiency. MS-Cluster [9] uses the cosine similarity as similarity measure. It obtains an approximate hierarchical clustering result by merging spectra that exceed an iteratively decreasing similarity threshold in a greedy fashion (rather than always merging the most similar spectra, as in standard hierarchical clustering). To avoid unnecessary similarity calculations only pairs of spectra that share at least one peak among their five most intense peaks are compared to each other. The spectra-cluster approach was originally developed as a reimplementation of MS-Cluster, with some refinements to improve the

cluster quality [12]. A highly parallel implementation was subsequently developed [13] to efficiently cluster large amounts of public data available in the PRoteomics IDEntifications (PRIDE) database [31]. Similar to MS-Cluster, spectra-cluster uses an iterative greedy approach to merge similar spectra. In contrast, instead of the cosine similarity a probabilistic scoring scheme [7] was adopted as similarity measure [13]. MaRaCluster [37] uses a specialized similarity measure that relies on the rarity of fragment peaks to compare MS/MS spectra. Based on the intuition that peaks shared by only a few spectra offer more evidence than peaks shared by a large number of spectra, relative to a background frequency of fragment peaks with specific m/z values, matches of highly frequent fragment peaks contribute less to the spectrum similarity than matches of rare peaks. Next, MaRaCluster uses hierarchical clustering with complete linkage to group similar spectra in clusters. Finally, msCRUSH [39] is a fast spectrum clustering tool based on locality-sensitive hashing [11]. By efficiently hashing similar spectra to identical buckets, unnecessary pairwise spectrum comparisons can be avoided. Next, within each bucket a similar greedy spectrum merging strategy is performed as employed by MS-Cluster and spectra-cluster.

Because each of these spectrum clustering tools use different spectrum similarity measures, clustering methods, and computational optimizations, their clustering results and computational performance will exhibit different characteristics. Here we introduce *falcon*, a fast spectrum clustering approach, and compare it to two widely used clustering tools, MS-Cluster and spectra-cluster. By making use of advanced algorithmic techniques, *falcon* is optimized for highly efficient spectrum clustering. It uses feature hashing to convert high-resolution MS/MS spectra to low-dimensional vectors [4], in combination with efficient nearest neighbor searching in the vector metric space using the cosine similarity [5]. Next, spectra are grouped into clusters by density-based clustering [8]. We compare *falcon*'s performance to MS-Cluster and spectra-cluster in terms of runtime and clustering quality, and show that it succeeds in efficiently clustering large amounts of spectral data. *falcon* is freely available as open source under the permissive BSD license at `https://github.com/bittremieux/falcon`.

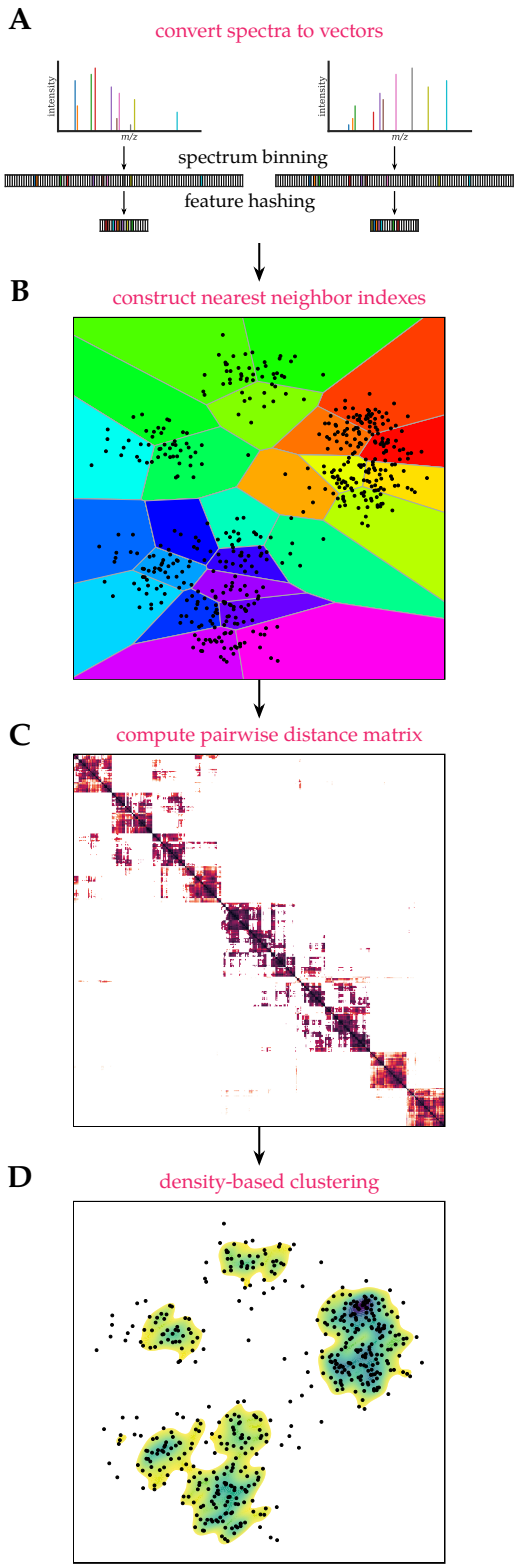

Figure 1: *falcon* spectrum clustering workflow. (**A**) High-resolution MS/MS spectra are converted to low-dimensional vectors using feature hashing. (**B**) Vectors are split into intervals based on the precursor m/z of the corresponding spectra to construct nearest neighbor indexes for highly efficient spectrum comparison. (**C**) A sparse pairwise distance matrix is computed by retrieving similarities to neighboring spectra using the nearest neighbor indexes. (**D**) Density-based clustering using the pairwise distance matrix is performed to find spectrum clusters.

2 Methods

2.1 Spectrum preprocessing

Only MS/MS spectra with precursor charges 2 and 3 are considered. The spectra are preprocessed by removing peaks corresponding to the precursor ion and noise peaks with an intensity below 1 % of the base peak intensity, and, if applicable, spectra are further restricted to their 50 most intense peaks.

Low-quality spectra that have fewer than five peaks remaining or with a mass range between their minimum and maximum peak less than 250 Da after peak removal are discarded. Finally, peak intensities are rank transformed to de-emphasize overly dominant peaks [25].

2.2 Feature hashing to convert high-resolution spectra to low-dimensional vectors

To build a nearest neighbor index for efficient spectrum similarity searching, spectra need to be vectorized to represent them as points in a multidimensional space. MS/MS spectra typically contain dozens to hundreds of peaks, whose m/z values are measured at a resolution in the order of $1/100\,m/z$. As such, a straightforward approach to convert spectra to vectors by dividing the mass range into small bins and assigning each peak's intensity to the corresponding bin would result in extremely high-dimensional, sparse vectors that are not suitable for efficient nearest neighbor searching due to the curse of dimensionality [2]. Alternatively, larger mass bins can be used to reduce the vectors' dimensionality and sparsity. However, because such larger mass bins considerably exceed the fragment mass tolerance when dealing with high-resolution spectra, multiple distinct fragments can get merged into the same mass bin. This merging leads to an overestimation of the spectrum similarity when comparing two spectra to each other using their vector representations due to spurious matches between fragments.

Instead, a feature hashing scheme [44] is used to convert high-resolution MS/MS spectra to low-dimensional vectors while closely capturing their fine-grained mass resolution. The following two-step procedure is used to convert a high-resolution MS/MS spectrum to a vector (figure 1A) [4]:

1. Convert the spectrum to a sparse vector using small mass bins to tightly capture fragment masses.

2. Hash the sparse, high-dimensional, vector to a lower-dimensional vector by using a hash function to map the mass bins separately to a small number of hash bins.

More precisely, let $h : \mathbb{N} \rightarrow \{1, \ldots, m\}$ be a random hash function. Then h can be used to convert a vector $x = \langle x_1, \ldots, x_n \rangle$ to a vector $x' = \langle x'_1, \ldots, x'_m \rangle$, with $m \ll n$:

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

As hash function h the MurmurHash3 algorithm [1], a popular non-cryptographic hash function, is used.

It can be proven that under moderate assumptions feature hashing approximately conserves the Euclidean norm [10], and hence, the cosine similarity between hashed vectors can be used to approximate the similarity between the original, high-dimensional vectors and spectra.

2.3 Efficient density-based clustering using nearest neighbor searching

Nearest neighbor searching is used to process large search spaces for efficient spectrum clustering [5]. Per precursor charge the MS/MS spectra are partitioned into $1\,m/z$ buckets based on their precursor mass and converted to vectors as described previously. Next, the spectrum vectors in each bucket are partitioned into data subspaces to create a Voronoi diagram (figure 1B). The Voronoi diagram is encoded by an inverted index, with each Voronoi cell defined by a single vector, determined using k-means clustering to find a user-specified number of representative vectors, and all vectors are assigned to their nearest representative vector. This inverted index can then be used for efficient similarity searching. Instead of having to compare all spectrum vectors to all other vectors in the bucket to find their nearest neighbors, after mapping the vectors to their Voronoi cells they only need to be compared to the limited number of vectors therein.

The accuracy and speed of similarity searching is governed by two hyperparameters: the number of Voronoi cells to use during construction of the inverted index and the number of neighboring cells to explore during searching. Using a greater number of Voronoi cells achieves a more fine-grained partitioning of the data space, and exploring more cells during searching decreases the chance of missing a nearest neighbor in the high-dimensional space. In practice, for m/z buckets that contain fewer than 1000 spectra a brute-force search is used. For larger m/z buckets the number of Voronoi cells is dynamically set based on the number of spectra in the bucket N. For buckets that consist of up to one million spectra $2^{\lfloor \log_2 \frac{N}{39} \rfloor}$ Voronoi cells are used, for buckets that consist of up to ten million spectra 2^{16} Voronoi cells are used, for buckets that consist of up to one hundred million spectra 2^{18} Voronoi cells are used, and for larger buckets 2^{20} Voronoi cells are used. During searching maximum 32 neighboring Voronoi cells per query vector are explored.

This efficient similarity searching is used to construct a sparse pairwise distance matrix that contains the cosine distances between each spectrum and a limited number of its nearest neighbors, additionally filtered using a precursor mass tolerance (figure 1C). Besides being able to retrieve the nearest neighbors highly efficiently, not having to compute and store all pairwise similarities also avoids extreme memory requirements.

Next, the pairwise distance matrix is used to cluster the data using the DBSCAN algorithm (figure 1D) [8]. Briefly, if a given number of spectra are

close to each other and form a dense data subspace, with closeness defined relative to a user-specified distance threshold, they will be grouped in clusters. An important advantage of DBSCAN is that the number of clusters is not required to be known in advance. Instead, it is able to find clusters in dense regions, whereas spectra in low-density regions, without a sufficient number of close neighbors, will be marked as noise. Additionally, DBSCAN is scalable: using a sparse pairwise distance matrix as input it can effortlessly process millions to billions of data points.

A disadvantage of this clustering approach, however, is that despite using a precursor mass filter during construction of the pairwise distance matrix, spectra within a single cluster can still exceed the precursor mass tolerance if they are connected by another spectrum with an intermediate precursor mass. To avoid such false positives, the clusters reported by DBSCAN are postprocessed by hierarchical clustering with maximum linkage of the cluster members' precursor masses. In this fashion, some clusters are split into smaller, coherent clusters so that none of the spectra in a single cluster have a pairwise precursor mass difference that exceeds the precursor mass tolerance.

2.4 Data

Spectrum clustering was performed on the human draft proteome dataset by Kim et al. [21]. This dataset aims to cover the whole human proteome and consists of 30 human samples in 2212 raw files, corresponding to 25 million MS/MS spectra. For full details on the sample preparation and acquisition see the original publication by Kim et al. [21]. Raw files were downloaded from PRIDE [31] (project PXD000561) and converted to MGF files using ThermoRawFileParser (version 1.2.3) [16].

Spectrum identifications were downloaded from MassIVE reanalysis RMSV000000091.3. These identifications were obtained via automatic reanalysis of public data on MassIVE using MS-GF+ [22]. Spectra were searched against the UniProtKB/Swiss-Prot human reference proteome (downloaded 2016/05/23) [6] augmented with common contaminants. Search settings included a 50 ppm precursor mass tolerance, trypsin cleavage with maximum one non-enzymatic peptide terminus, and cysteine carbamidomethylation as a static modification. Methionine oxidation, formation of pyroglutamate from N-terminal glutamine, N-terminal carbamylation, N-terminal acetylation, and deamidation of asparagine and glutamine were specified as variable modifications, with maximum one modification per peptide. Peptide-spectrum matches (PSMs) were filtered at 1 % false discovery rate, resulting in 10 487 235 spectrum identifications.

All clustering results are available on Zenodo at doi:10.5281/zenodo.4479887.

2.5 Clustering evaluation

Clusters are required to consist of minimum two spectra. *falcon* explicitly designates non-clustered spectra as noise points with cluster label "-1". In contrast, MS-Cluster and spectra-cluster report singleton clusters consisting of single spectra with unique cluster labels. To evaluate the cluster quality in a consistent fashion, such clusters in the MS-Cluster and spectra-cluster output are postprocessed and labeled as noise as well.

The following evaluation measures are used to assess cluster quality:

Clustered spectra The number of spectra in non-noise clusters divided by the total number of spectra.

Incorrectly clustered spectra The number of incorrectly clustered spectra in non-noise clusters divided by the total number of identified spectra in non-noise clusters. Spectra are considered incorrectly clustered if their peptide labels deviate from the most frequent peptide label in their clusters, with unidentified spectra not considered.

Completeness Completeness measures the fragmentation of spectra corresponding to the same peptide across multiple clusters and is based on the notion of *entropy* in information theory. A clustering result that perfectly satisfies the completeness criterium (value "1") assigns all PSMs with an identical peptide label to a single cluster. Completeness is computed as one minus the conditional entropy of the cluster distribution given the peptide assignments divided by the maximum reduction in entropy the peptide assignments could provide [33].

Runtime measurements were acquired on a single compute node with two 14-core Intel E5-2680v4 CPUs and 128 GB memory.

2.6 Clustering configuration

2.6.1 *falcon*

Spectrum preprocessing was performed as described in section 2.1. To convert the spectra to vectors, first virtual vectors with bin width 0.05 m/z were created. Next, these vectors were converted to vectors of length 800 using feature hashing.

To match spectra to each other they were first partitioned into 1 m/z buckets based on their precursor mass. Next, for each bucket a Voronoi diagram was created consisting of up to 65 536 cells, depending on the number of spectra in the bucket. During querying, at most 32 cells per query were explored. For each spectrum, its 128 nearest neighbors were retrieved via similarity searching. Neighbors whose precursor mass tolerance exceeded 20 ppm

were omitted, after which for each spectrum the cosine distances to the maximum 64 nearest neighbors were stored in the sparse pairwise distance matrix.

During DBSCAN clustering dense regions were defined as having minimum two spectra with a cosine distance below 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, or 0.35. Clusters were split in a postprocessing step to ensure that pairwise precursor mass differences between cluster members did not exceed 20 ppm.

2.6.2 MS-Cluster

MS-Cluster (version 2.00) [9] was run using its "LTQ_TRYP" model for three rounds of clustering with mixture probability 0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, or 0.2. The fragment mass tolerance and precursor mass tolerance were 0.05 Da and 20 ppm, respectively, and precursor charges were read from the input files. Other options were kept at their default values.

2.6.3 spectra-cluster

spectra-cluster (version 1.1.2) [12, 13] was run in its "fast mode" for three rounds of clustering with the final clustering threshold 0.999 99, 0.9999, 0.999, 0.99, 0.95, 0.9, 0.8, or 0.7. The fragment mass tolerance and precursor mass tolerance were 0.05 Da and 20 ppm, respectively, and MGF comment strings were ignored. Other options were kept at their default values.

2.7 Code availability

falcon was implemented in Python 3.8. Pyteomics (version 4.4.0) [24] was used to read MS/MS spectra in the mzML [26], mzXML, and MGF format. spectrum_utils (version 0.3.4) [3] was used for spectrum preprocessing. Faiss (version 1.6.5) [20] was used for efficient similarity searching. Scikit-Learn (version 0.24.0) [30] was used for DBSCAN clustering, and fastcluster (version 1.1.26) [29] was used for hierarchical clustering. Additional scientific computing was done using NumPy (version 1.19.4) [14], SciPy (version 1.5.3) [35], Numba (version 0.52.0) [23], and Pandas (version 1.2.0) [28]. Data analysis and visualization were performed using Jupyter Notebooks [38], matplotlib (version 3.3.3) [17], and Seaborn (version 0.11.0) [42].

All code is available as open source under the permissive BSD license at `https://github.com/bittremieux/falcon`. Code used to compare various spectrum clustering tools and to generate the figures presented here is available on GitHub (`https://github.com/bittremieux/falcon_notebooks`).

3 Results

An ideal clustering result groups MS/MS spectra corresponding to distinct peptides in individual, disjoint clusters. The main aspects that influence clustering quality are the spectrum similarity measure and the algorithm that is used to group similar spectra. As various spectrum clustering tools differ in these choices, even when processing identical MS/MS data they will produce cluster assignments with different characteristics. Additionally, the clustering algorithms should exhibit a good computational efficiency to be able to process large amounts of mass spectral data.

The comparison between *falcon*, MS-Cluster, and spectra-cluster in terms of their clustering quality shows that spectra-cluster succeeds in clustering a higher number of spectra at a comparable rate of incorrectly clustered spectra, while *falcon* and MS-Cluster achieve similar performance (figure 2A). Notably, *falcon* and MS-Cluster both use the cosine similarity as their spectrum similarity measure, while Griss et al. [13] report that replacing the cosine similarity with a probabilistic scoring approach in an updated version of spectra-cluster helped to improve its clustering accuracy.

Besides generating pure clusters containing spectra corresponding to a single peptide, clusters should also be as complete as possible, i.e. all spectra corresponding to a specific peptide should be grouped in a single cluster. Because each cluster can be represented by a consensus spectrum for subsequent downstream processing, a more complete clustering will achieve an increased data reduction by minimizing the generation of redundant cluster representatives. MS-Cluster and *falcon* outperform spectra-cluster in terms of completeness (figure 2B), and *falcon* consistently achieves the highest completeness at a comparable rate of incorrectly clustered spectra, indicating that it more often groups spectra corresponding to the same peptides together in single clusters. In contrast, spectra-cluster is not able to improve completeness as more spectra are clustered (figure 2B) and produces a higher number of small clusters (figure 2C, table 1). Even though spectra-cluster manages to correctly cluster a large number of spectra, it produces more fragmented clusters than the alternative tools as it does not group all observations of repeatedly occurring peptides together but instead splits these spectra across multiple smaller clusters.

To better understand the observed differences in completeness of the various clustering approaches, we investigated some clusters manually. For example, VATVSIPR/2, which is part of the pig trypsin contaminant protein P00761, is the most frequently identified peptide in the dataset (observed 18 657 times, figure 2D). In the spectra-cluster results, the spectra corresponding to this peptide are split into 1688 small clusters, with the largest cluster consisting of only 121 spectra and the other clusters con-

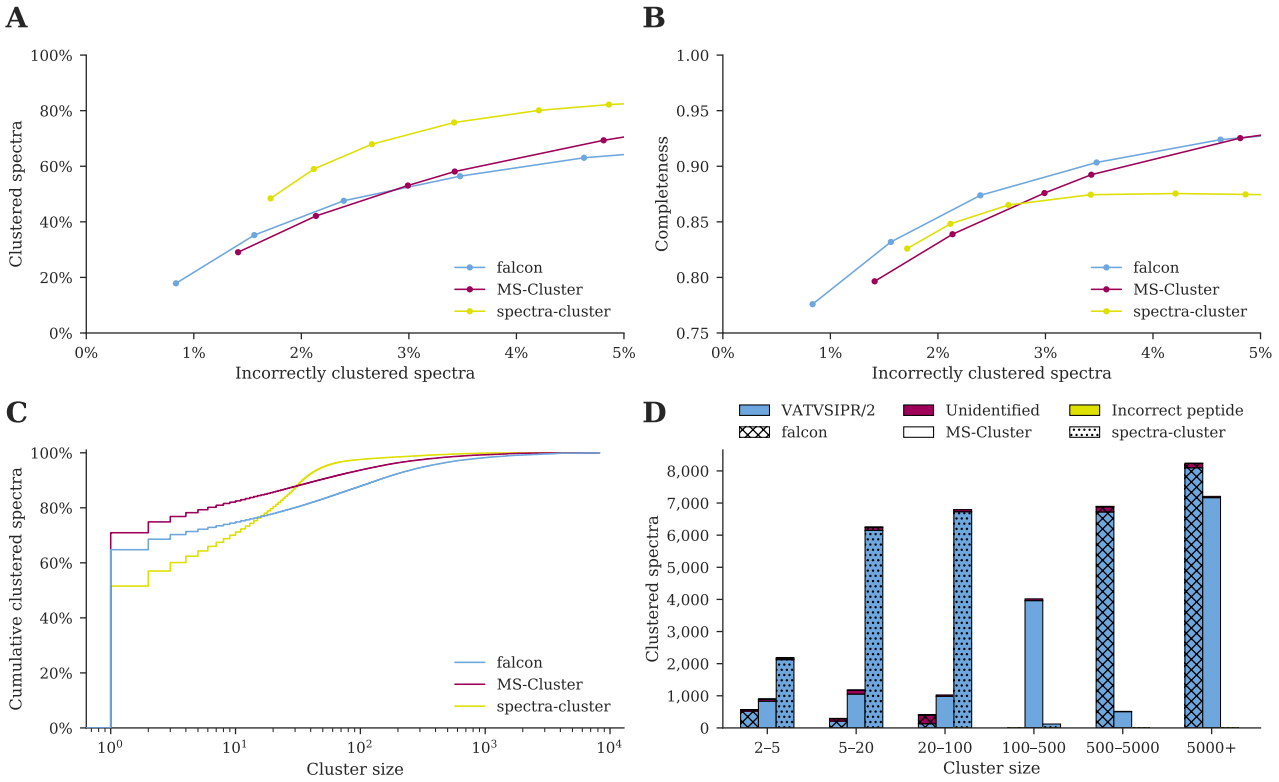

Figure 2: Clustering comparison between *falcon*, MS-Cluster, and spectra-cluster. (**A**) spectra-cluster succeeds in clustering more spectra than *falcon* and MS-Cluster at a comparable rate of incorrectly clustered spectra. (**B**) *falcon* produces a more complete clustering at a comparable rate of incorrectly clustered spectra, grouping spectra corresponding to specific peptides in single clusters. (**C**) Empirical cumulative distribution of the cluster sizes for the clustering results reported in table 1. spectra-cluster successfully clusters more spectra (less singleton clusters) but predominantly generates clusters consisting of fewer than 100 spectra. Both *falcon* and MS-Cluster produce more large clusters, with *falcon* creating the largest clusters. (**D**) Cluster sizes for frequently occuring peptide VATVSIPR/2. spectra-cluster splits spectra corresponding to this peptide into multiple small clusters consisting of less than 100 spectra each, whereas *falcon* and MS-Cluster produce a few large clusters that contain more than thousand spectra instead.

Tool	Runtime	Clustered spectra	Incorrectly clustered spectra	Clusters	Completeness
falcon	4 h 15 min ± 4 min	7 457 929 (35.23 %)	1.56 %	797 157	0.8319
MS-Cluster	19 h 0 min ± 32 min	6 139 161 (29.06 %)	1.41 %	859 798	0.7964
spectra-cluster	21 h 31 min ± 3 h 28 min	10 430 972 (48.44 %)	1.71 %	1 462 066	0.8259

Table 1: Comparison at a similar rate of incorrectly clustered spectra between *falcon*, MS-Cluster, and spectra-cluster in terms of clustering quality and runtime. The reported runtimes are the average and standard deviation for three exections of each clustering tool.

taining less than a hundred spectra. In contrast, MS-Cluster groups these spectra into 532 unique clusters, with the largest cluster consisting of 7208 spectra. Finally, *falcon* splits these spectra into only 289 unique clusters, of which the largest three clusters consist of 8238, 3779, and 3118 spectra respectively.

As current shotgun proteomics datasets grow ever larger, clustering algorithms need to be scalable to be able to process larger data volumes. For all three clustering tools, settings that are optimized for speed were used to ensure a fair comparison in terms of runtime. For MS-Cluster and spectra-cluster only three rounds of their iterative cluster refinement procedure were used. Additionally, spectra-cluster's "fast mode" was enabled. For *falcon* a limited number of Voronoi cells per query were examined. *falcon* was able to cluster the full dataset consisting of 25 million spectra in slightly over four hours — considerably faster than MS-Cluster and spectra-cluster (table 1). *falcon* achieves this excellent computational performance through its advanced nearest neighbor searching strategy, which enables it to efficiently compute a sparse pairwise distance matrix with only the relevant spectrum–spectrum match scores instead of having to consider all pairwise spectrum comparisons. Additionally, evaluating different hyperparameters of the density-based clustering step to obtain stricter or looser clusters can be done in only a matter of minutes using a precomputed pairwise distance matrix.

4 Conclusion

Here we have introduced the *falcon* spectrum clustering tool. *falcon* is optimized for computational efficiency using advanced spectrum vectorization and nearest neighbor searching strategies. It outperforms the popular spectrum clustering tools MS-Cluster and spectra-cluster in terms of runtime while generating clusters of a comparable quality. As such, *falcon* is ideally suited to process the ever increasing amounts of mass spectral data generated during mass spectrometry experiments, for example to efficiently cluster publicly available data on a repository scale [12, 13, 41].

Although *falcon* succeeds in clustering a similar number of spectra as MS-Cluster at a comparable rate of incorrectly clustered spectra, both tools are outperformed by spectra-cluster. Notably, *falcon* and MS-Cluster compare spectra to each other using the cosine similarity, whereas spectra-cluster uses a more advanced probabilistic scoring approach. The cosine similarity is a commonly used similarity measure [36]. Some of its advantages are that it can accurately capture spectrum similarity, it is easy to implement and interpret, and it is fast to evaluate. Consequently, the cosine similarity is a highly competitive and ubiquitous baseline method. However, alternative spectrum similarity methods, such as spectra-cluster's probabilistic score [13], MaRaCluster's fragment rarity score [37], or similarity measures derived from machine learning [15, 27] might be able to more sensitively capture spectrum similarity. Additionally, it is important to evaluate how MS/MS spectrum preprocessing [32] influences various similarity measures.

Among the three clustering tools evaluated, *falcon* generates the most complete clustering result and achieves the largest reduction in data volume when representing the clustered data by their consensus spectra. This can be especially relevant when spectrum clustering is used as a preprocessing step prior to a subsequent computationally intensive analysis, such as open modification searching [5]. Additionally, the reduction of redundant information can facilitate downstream interpretation, for example, by avoiding uninformative nodes and edges during molecular networking [40, 43].

falcon is freely available as open source under the permissive BSD license. The source code and instructions can be found at `https://github.com/bittremieux/falcon`.

Acknowledgments

W.B. is a postdoctoral researcher of the Research Foundation – Flanders (12W0418N). The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation – Flanders (FWO) and the Flemish Government. This work was supported in part by National Institutes of Health award U19 AG063744-01.

References

[1] *aappleby/smhasher*. `https://github.com/aappleby/smhasher`. (accessed: 2020-12-29).

[2] Beyer, K. S., Goldstein, J., Ramakrishnan, R., and Shaft, U. "When Is "Nearest Neighbor" Meaningful?" In: *Proceedings of the 7th International Conference on Database Theory - ICDT '99*. Jerusalem, Israel: Springer-Verlag, 1999, pp. 217–235.

[3] Bittremieux, W. "Spectrum_utils: A Python Package for Mass Spectrometry Data Processing and Visualization." In: *Analytical Chemistry* 92.1 (Jan. 7, 2020), pp. 659–661. DOI: `10.1021/acs.analchem.9b04884`.

[4] Bittremieux, W., Laukens, K., and Noble, W. S. "Extremely Fast and Accurate Open Modification Spectral Library Searching of High-Resolution Mass Spectra Using Feature Hashing and Graphics Processing Units." In: *Journal of Proteome Research* 18.10 (Aug. 26, 2019), pp. 3792–3799. DOI: `10.1021/acs.jproteome.9b00291`.

[5] Bittremieux, W., Meysman, P., Noble, W. S., and Laukens, K. "Fast Open Modification Spectral Library Searching through Approximate Nearest Neighbor Indexing." In: *Journal of Proteome Research* 17.10 (Oct. 5, 2018), pp. 3463–3474. DOI: 10.1021/acs.jproteome.8b00359.

[6] Breuza, L., Poux, S., Estreicher, A., Famiglietti, M. L., et al. "The UniProtKB Guide to the Human Proteome." In: *Database* 2016 (2016), bav120. DOI: 10.1093/database/bav120.

[7] Dasari, S., Chambers, M. C., Martinez, M. A., Carpenter, K. L., et al. "Pepitome: Evaluating Improved Spectral Library Search for Identification Complementarity and Quality Assessment." In: *Journal of Proteome Research* 11.3 (Mar. 2, 2012), pp. 1686–1695. DOI: 10.1021/pr200874e.

[8] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining - KDD'96*. Portland, OR, USA: AAAI Press, Aug. 2, 1996, pp. 226–231.

[9] Frank, A. M., Bandeira, N., Shen, Z., Tanner, S., et al. "Clustering Millions of Tandem Mass Spectra." In: *Journal of Proteome Research* 7.1 (Jan. 4, 2008), pp. 113–122. DOI: 10.1021/pr070361e.

[10] Freksen, C. B., Kamma, L., and Larsen, K. G. *Fully Understanding the Hashing Trick*. May 22, 2018. arXiv: 1805.08539 [cs, math, stat]. URL: http://arxiv.org/abs/1805.08539.

[11] Gionis, A., Indyk, P., and Motwani, R. "Similarity Search in High Dimensions via Hashing." In: *Proceedings of the 25th International Conference on Very Large Data Bases - VLDB '99*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Sept. 7, 1999, pp. 518–529.

[12] Griss, J., Foster, J. M., Hermjakob, H., and Vizcaíno, J. A. "PRIDE Cluster: Building a Consensus of Proteomics Data." In: *Nature Methods* 10.2 (Jan. 30, 2013), pp. 95–96. DOI: 10.1038/nmeth.2343.

[13] Griss, J., Perez-Riverol, Y., Lewis, S., Tabb, D. L., et al. "Recognizing Millions of Consistently Unidentified Spectra across Hundreds of Shotgun Proteomics Datasets." In: *Nature Methods* 13.8 (June 27, 2016), pp. 651–656. DOI: 10.1038/nmeth.3902.

[14] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., et al. "Array Programming with NumPy." In: *Nature* 585.7825 (Sept. 17, 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[15] Huber, F., Ridder, L., Verhoeven, S., Spaaks, J. H., et al. "Spec2Vec: Improved Mass Spectral Similarity Scoring through Learning of Structural Relationships." In: *bioRxiv* (Sept. 25, 2020). DOI: 10.1101/2020.08.11.245928.

[16] Hulstaert, N., Shofstahl, J., Sachsenberg, T., Walzer, M., et al. "ThermoRawFileParser: Modular, Scalable, and Cross-Platform RAW File Conversion." In: *Journal of Proteome Research* 19.1 (Jan. 3, 2020), pp. 537–542. DOI: 10.1021/acs.jproteome.9b00328.

[17] Hunter, J. D. "Matplotlib: A 2D Graphics Environment." In: *Computing in Science & Engineering* 9.3 (June 18, 2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[18] Huttlin, E. L., Bruckner, R. J., Paulo, J. A., Cannon, J. R., et al. "Architecture of the Human Interactome Defines Protein Communities and Disease Networks." In: *Nature* 545.7655 (May 17, 2017), pp. 505–509. DOI: 10.1038/nature22366.

[19] Huttlin, E. L., Ting, L., Bruckner, R. J., Gebreab, F., et al. "The BioPlex Network: A Systematic Exploration of the Human Interactome." In: *Cell* 162.2 (July 16, 2015), pp. 425–440. DOI: 10.1016/j.cell.2015.06.043.

[20] Johnson, J., Douze, M., and Jégou, H. *Billion-Scale Similarity Search with GPUs*. Feb. 28, 2017. arXiv: 1702.08734 [cs]. URL: http://arxiv.org/abs/1702.08734.

[21] Kim, M.-S., Pinto, S. M., Getnet, D., Nirujogi, R. S., et al. "A Draft Map of the Human Proteome." In: *Nature* 509.7502 (May 28, 2014), pp. 575–581. DOI: 10.1038/nature13302.

[22] Kim, S. and Pevzner, P. A. "MS-GF+ Makes Progress towards a Universal Database Search Tool for Proteomics." In: *Nature Communications* 5 (Oct. 31, 2014), p. 5277. DOI: 10.1038/ncomms6277.

[23] Lam, S. K., Pitrou, A., and Seibert, S. "Numba: A LLVM-Based Python JIT Compiler." In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM '15*. Austin, TX, USA: ACM Press, Nov. 15, 2015, pp. 1–6. DOI: 10.1145/2833157.2833162.

[24] Levitsky, L. I., Klein, J. A., Ivanov, M. V., and Gorshkov, M. "Pyteomics 4.0: Five Years of Development of a Python Proteomics Framework." In: *Journal of Proteome Research* 18.2 (Feb. 1, 2019), pp. 709–714. DOI: 10.1021/acs.jproteome.8b00717.

[25] Ma, C. W. M. and Lam, H. "Hunting for Unexpected Post-Translational Modifications by Spectral Library Searching with Tier-Wise Scoring." In: *Journal of Proteome Research* 13.5 (May 2, 2014), pp. 2262–2271. DOI: 10.1021/pr401006g.

[26] Martens, L., Chambers, M., Sturm, M., Kessner, D., et al. "mzML—a Community Standard for Mass Spectrometry Data." In: *Molecular & Cellular Proteomics* 10.1 (Jan. 1, 2011), R110.000133–R110.000133. DOI: 10.1074/mcp.R110.000133.

[27] May, D. H., Bilmes, J., and Noble, W. S. "A Learned Embedding for Efficient Joint Analysis of Millions of Mass Spectra." In: *bioRxiv* (Nov. 29, 2018). DOI: 10.1101/483263.

[28] McKinney, W. "Data Structures for Statistical Computing in Python." In: *Proceedings of the 9th Python in Science Conference*. Ed. by van der Walt, S. and Millman, J. Austin, Texas, USA, 2010, pp. 51–56.

[29] Müllner, D. "Fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python." In: *Journal of Statistical Software* 53.9 (May 29, 2013). DOI: 10.18637/jss.v053.i09.

[30] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., et al. "Scikit-Learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (Oct 2011), pp. 2825–2830.

[31] Perez-Riverol, Y., Csordas, A., Bai, J., Bernal-Llinares, M., et al. "The PRIDE Database and Related Tools and Resources in 2019: Improving Support for Quantification Data." In: *Nucleic Acids Research* 47.D1 (Jan. 8, 2019), pp. D442–D450. DOI: 10.1093/nar/gky1106.

[32] Renard, B. Y., Kirchner, M., Monigatti, F., Ivanov, A. R., et al. "When Less Can Yield More - Computational Preprocessing of MS/MS Spectra for Peptide Identification." In: *PROTEOMICS* 9.21 (Nov. 2009), pp. 4978–4984. DOI: 10.1002/pmic.200900326.

[33] Rosenberg, A. and Hirschberg, J. "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) - CoNLL-EMNLP 2007*. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 410–420.

[34] Sadygov, R. G., Cociorva, D., and Yates, J. R. I. "Large-Scale Database Searching Using Tandem Mass Spectra: Looking up the Answer in the Back of the Book." In: *Nature Methods* 1.3 (Nov. 18, 2004), pp. 195–202. DOI: 10.1038/nmeth725.

[35] SciPy 1.0 Contributors, Virtanen, P., Gommers, R., Oliphant, T. E., et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* (Feb. 3, 2020). DOI: 10.1038/s41592-019-0686-2.

[36] Stein, S. E. and Scott, D. R. "Optimization and Testing of Mass Spectral Library Search Algorithms for Compound Identification." In: *Journal of the American Society for Mass Spectrometry* 5.9 (Sept. 1994), pp. 859–866. DOI: 10.1016/1044-0305(94)87009-8.

[37] The, M. and Käll, L. "MaRaCluster: A Fragment Rarity Metric for Clustering Fragment Spectra in Shotgun Proteomics." In: *Journal of Proteome Research* 15.3 (Mar. 4, 2016), pp. 713–720. DOI: 10.1021/acs.jproteome.5b00749.

[38] Thomas, K., Benjamin, R.-K., Fernando, P., Brian, G., et al. "Jupyter Notebooks – A Publishing Format for Reproducible Computational Workflows." In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, 2016, pp. 87–90.

[39] Wang, L., Li, S., and Tang, H. "msCRUSH: Fast Tandem Mass Spectral Clustering Using Locality Sensitive Hashing." In: *Journal of Proteome Research* (Jan. 4, 2019). DOI: 10.1021/acs.jproteome.8b00448.

[40] Wang, M., Carver, J. J., Phelan, V. V., Sanchez, L. M., et al. "Sharing and Community Curation of Mass Spectrometry Data with Global Natural Products Social Molecular Networking." In: *Nature Biotechnology* 34.8 (Aug. 9, 2016), pp. 828–837. DOI: 10.1038/nbt.3597.

[41] Wang, M., Wang, J., Carver, J., Pullman, B. S., et al. "Assembling the Community-Scale Discoverable Human Proteome." In: *Cell Systems* 7.4 (Oct. 24, 2018), 412–421.e5. DOI: 10.1016/j.cels.2018.08.004.

[42] Waskom, M. and team, the seaborn development. *mwaskom/seaborn*. https://doi.org/10.5281/zenodo.592845. Version latest. Sept. 2020. DOI: 10.5281/zenodo.592845.

[43] Watrous, J., Roach, P., Alexandrov, T., Heath, B. S., et al. "Mass Spectral Molecular Networking of Living Microbial Colonies." In: *Proceedings of the National Academy of Sciences* 109.26 (June 26, 2012), E1743–E1752. DOI: 10.1073/pnas.1203689109.

[44] Weinberger, K., Dasgupta, A., Langford, J., Smola, A., et al. "Feature Hashing for Large Scale Multitask Learning." In: *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. Montreal, Quebec, Canada: ACM Press, 2009, pp. 1113–1120. DOI: 10.1145/1553374.1553516.

[45] Wilhelm, M., Schlegl, J., Hahne, H., Gholami, A. M., et al. "Mass-Spectrometry-Based Draft of the Human Proteome." In: *Nature* 509.7502 (May 28, 2014), pp. 582–587. DOI: 10.1038/nature13319.