Efficient Shared Peak Counting in Database Peptide Search Using *Compact* Data Structure for Fragment-Ion Index

Muhammad Haseeb[†], Fahad Saeed*[†]
[†]School of Computing and Information Sciences
Florida International University, Miami, Florida 33199, USA
Email: {mhaseeb,fsaeed}@fiu.edu

***Abstract*—Database search is the most commonly employed method for identification of peptides from MS/MS spectra data that is obtained from a shotgun LC-MS/MS experiment. The search involves comparing experimentally obtained MS/MS spectra against a set of theoretical spectra predicted from a protein sequence database. One of the most commonly employed similarity metrics for spectral comparison is the shared-peak count between a pair of MS/MS spectra. Most modern methods index all generated fragment-ion data from theoretical spectra to speed up the shared peak count computations between a given experimental spectrum and all theoretical spectra. However, the bottleneck for this method is the gigantic memory footprint of fragment-ion index that leads to non-scalable solutions. In this paper, we present a novel data structure, called *Compact Fragment-Ion Index Representation (CFIR)*, that efficiently compresses highly redundant ion-mass information in the data to reduce the index size. Our proposed data structure outperforms all existing fragment-ion indexing data structures by at least 2× in memory consumption while exhibiting the same time complexity for index construction and peptide search. Our experimental results confirm superior memory efficiency for CFIR data structure over existing data structures and methods. The results also show comparable indexing speed, search speed and speedup scalability for CFIR based index and state of the art algorithms. Further, the results show optimal performance scaling for CFIR data structure for up to 100% more fragment-ion data compared to the best existing data structure.**

***Index Terms*—indexing, proteomics, optimization, memory-efficiency**

I. Introduction

In a shotgun proteomics experiment, a complex protein mixture is proteolyzed using an enzyme, most commonly, Trypsin. The digested peptides are then fed to a pipeline of liquid-chromatography (LC) followed by mass spectrometry (MS/MS). The acquired tandem MS/MS spectra data (called experimental spectra) are then identified and assigned to a peptide sequence using computational techniques. The most commonly used computational method, called database search, involves searching the experimental MS/MS spectra against a set of theoretical MS/MS spectra predicted from a protein sequence database [1]. The search speed and accuracy of database peptide search algorithms depend on numerous factors such as search parameters [2], data pre-processing methods [3], [4], [5], feature selection [6], [7], [8], post-translational modifications (PTMs) included [9], [10], [5], [11], database filtration methods [12], [13], [14], [10], peptide-to-spectrum scoring algorithm [11], [15], [16], [17], [18], [19], [20], [21], [22], [23], confidence assignment algorithm [24], [14], [10], [13], [12], [25] and so on.

* Corresponding Author

Recent works have shown that the unrestricted peptide search methods allow identification of a much larger fraction of experimental MS/MS spectra as compared to restricted peptide search [9], [8], [10], [14], [13]. However, the unrestricted search consumes massive search times due to massively increased number of required computationally expensive peptide-to-spectrum comparisons. Most state-of-the-art peptide search algorithms employ one or more database filtration methods to substantially reduce the search space to only the related database entries when a given experimental spectrum. Some of the most commonly employed database filtration methods include *peptide precursor mass* [26], [27], [5], [13], *sequence-tagging* [14], [28], [29], [30], [31], [32], [33], [13], [12] and *shared-peak counting* [34], [35], [36], [10], [37], [38], [39], [12], [19], [27]. Several additional features including peak labels, complementary peaks, peptide lengths, average sequence length per missed cleavage, precursor mass and charge, enzyme specificity, and known post-translational modifications (PTMs) are also used to correctly assign MS/MS spectra to database peptide sequences.

Motivation: The *shared-peak count* is commonly employed by various modern peptide search algorithms in combination with several other similarity metrics for search space filtration and peptide deduction. However, computing shared peak count between a pair of spectra using naive methods is highly compute intensive and memory-inefficient. Therefore, modern shared peak counting techniques usually construct an (inverted) fragment-ion index to speed up the computation at the cost of substantially large memory footprint [14], [10], [38], [7]. The fragment-ion look up (search) from a large index, specially using multiple cores, is accompanied by huge memory read/write and caching overheads. Further, as the database size increases, the index spans over multiple NUMA nodes (non-local access), may be swapped several times (page faults), and/or may need to be split into smaller independent chunks and processed sequentially [10]. All these factors bottleneck the system performance specially when running in a multi-processor or a distributed memory environment. Therefore, this work intends to optimize the fragment-ion index size without compromising the shared-peak counting (index look up) speed so that the required memory resources per unit node are optimized resulting in overall resource and cost-effective solutions.

Contributions: In this paper, we present a *compact* data structure for fragment-ion data that outperforms all existing data structures in memory footprint. Our proposed data structure, called Compact Fragment-Ion Index Representation (CFIR), is based on the observation that the number of unique ion-masses that appear in the fragment-ion data are extremely small as compared to the size of data. This property in data allows us to sort and compress the repeated ion-mass information in an efficient way significantly reducing the index size. Our experimental results confirm that the CFIR data structure consumes at most 50% memory ($2\times$ improvement) as the best existing data structure while exhibiting the same complexity for index construction and peptide search time. The results also depict a linear speedup scalability for CFIR based peptide search with increasing number of parallel cores. Further, the peptide search speed performance comparison with increasing index size reveals that the CFIR data structure exhibits optimal scalability for up to at least 100% more fragment-ion data as compared to the best existing data structure. Our results indicate that the proposed data structure can be used in memory distributed peptide search algorithms where the algorithm spans over several parallel nodes and the resources per node must be optimally consumed to avoid power, network, load imbalance and compute cost penalties. The CFIR data structure has been implemented as an open-source software using C/C++ and OpenMP.

II. Related Work

A. Shared Peak Counting

A peak is said to be shared between a pair of MS/MS spectra if both spectra contain a fragment-ion (peak) having m/z within a certain mass difference (ΔF). This tolerance allows conversion of floating point ion m/z's or simply ion-masses to integers by scaling m/z's with a factor (r) where $r \geq 1/\Delta F$. Therefore, a MS/MS spectrum can be represented as a (sparse) binary vector over a discrete range of dimensions $(0 : r : M]$ where r is the step size and M is the maximum integer mass in the spectrum. In this representation, a dot product (with relaxed mass difference (ΔF)) between a pair of MS/MS spectra will yield the number of shared peaks between them. Similarly, the shared-peak counting problem between a (query) MS/MS spectrum q and a matrix N constructed by stacking all (say X) theoretically predicted spectra in the database can be solved by applying a Matrix Vector multiplication. The result of this operation would be a vector (c) containing the number of shared peaks between the query spectrum and all theoretical spectra in the database. However, it is interesting to note that the sparse vectors (theoretical spectra) contained in N will have 1's at only particular dimensions in the matrix and almost most of the columns in N will be all zeros resulting in a sparse matrix. An example is illustrated as follows:

$$N_{X\times M} \times q_{M\times 1} = c_{X\times 1}$$

$$\begin{bmatrix} 0 & 1 & .. & 0 & .. & 1 \\ 1 & 0 & .. & 1 & .. & 0 \\ 0 & 0 & .. & 0 & .. & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ .. \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 7 \\ 4 \end{bmatrix}$$

B. Fragment-Ion Indexing Methods

Since the matrix N is sparse, conventional matrix representations cannot be efficiently used for N. Therefore, variants of sparse matrix representations such as Compressed Sparse Row (CSR) and Compressed Sparse Column (CSC) [40] are used to construct an inverted index only storing the data about the non-zero (NNZ) entries in the matrix with some overhead. When constructing the fragment-ion index, it is typical to store the information about (integer) ion-masses, ion-series, ion-number, and spectra IDs for each fragment-ion in N. Further, the two following query operations are supported by most fragment-ion index data structures:

- $rank(T, m)$: Returns the number of occurrences of fragment-ion with mass m/z (m) in T.
- $select(T, m, k)$: Returns the position of k^{th} occurrence of fragment ion with m/z (m) in T. Return all positions if k is not specified.

The shared peak count algorithm can be translated into a combination of above two operations and therefore, all fragment-ion indexing methods strive to optimize the performance of these operations. For instance, pFind-Alioth constructs a hash-map where the keys are ion-masses and values are 12-byte entries per indexed fragment-ion to store ion-origin information. MSFragger constructs a sorted-by-ion-mass array containing 8-byte entries per indexed fragment-ion to store ion mass and ion origin information. ProteoStorm filters the search space from several proteomics databases using MSFragger-like index for meta-proteomics application. SpecOMS first clusters and buckets ions based on their masses and then constructs FP-tree like data structure to encode the shared peak count between spectra for classification. ANN-SoLo works with spectral libraries and extracts ~50 features (peaks) from each spectrum in the library to construct an approximate nearest neighbor index. The experimental spectra are first classified to their appropriate nearest neighborhoods which are then formally compared. PEAKS-DB filters the protein sequence database using sequence-tagging and then uses shared-peak counts as one of 9 similarity features for further database filtration and formal scoring. Andromeda and OMSSA first filter the search space based on precursor masses and then compute shared peak score between an experimental spectrum and all filtered peptides on the fly. X!Tandem, [41] and [36] use a vectorized dot product approach and compute variants of Cosine distance between vectors. OMSSA, X!Tandem, Andromeda do not construct any index and compute shared peaks scores on the fly and use them as one of the similarity metric in computation of peptide-to-spectrum similarity score. However, the search engines including ProteoStorm, MSFragger, pFind-Alioth, SpecOMS, ANN-SoLo are capable of unrestricted search and therefore, employ index based methods for search space filtration.

III. Methods

In this section, we first discuss the data representation for fragment-ion data followed by the construction method for the proposed data structure. Then we discuss the algorithm for counting shared peaks and peptide search using the compact fragment-ion index.

A. Data Representation

We represent each theoretical MS/MS spectrum as an ordered list of numbers where each number represents the (integer) m/z of a predicted fragment-ion in the spectrum. The ions in each generated spectrum S_i are first ordered by their ion-series (b/y) and then ordered by the fragment charge and finally by their m/z's. However, since the length of a theoretical spectrum varies with the length of parent peptide sequence, we cannot stack all theoretical spectra together to form the matrix N. Therefore, we first group the database peptide sequences by their length and then using all theoretical spectra in each group, we construct an instance of matrix $N = [S_1, S_2, S_3, ..., S_X]$ where the fragment-ion m/z's in each S_i lie within the range $R(N) = 0 \leq i_j \leq M - 1$. The construction of matrix N using X spectra of length n is shown in Figure 1 can be written as:

$$N = \begin{bmatrix} i_{11} & .. & i_{1k} & .. & i_{1n} \\ i_{j1} & .. & i_{jk} & .. & i_{jn} \\ i_{X1} & .. & i_{Xk} & .. & i_{Xn} \end{bmatrix}$$

where i_{jk} corresponds to the m/z of k^{th} fragment-ion in the j^{th} spectrum. To further simplify the ion indices in N, we re-enumerate the ions to form a flattened list $T = [i_0, i_1, i_2, ..., i_{Xn-1}]$ where original ion-positions can be computed using information about N's dimensions i.e. $dim(N) = (X \times n)$; remember that the ions are ordered in each spectrum and the spectrum length is fixed per instance of N.

B. Compact Fragment-Ion Index Representation (CFIR)

We observed the information in above constructed list T and found that the number of appearing unique ion m/z's (ion-masses) is extremely small as compared to the size of list (Figure 2). Therefore, the ion-mass information which typically consumes at least 4 bytes (integer) per fragment-ion can be compressed into a compact representation in the index as follows: We define a list A containing the indices of ions in T given as $A = [0, 1, 2, .., Xn - 1]$. Next, we apply the *Stable KeyValue Sort* operation on A using T as key to yield two arrays: Ω and Λ respectively.

$$\Omega, \Lambda = StableKeyValueSort(T, A)$$

The list Λ now contains the locations of sorted fragment-ions in T in the same order as they appear in T whereas Ω is the sorted version of T. Now since we know that the T only had a small number of unique ion-masses, its sorted version (Ω) will contain large number of repeated entries at adjacent indices(extremely low entropy). Therefore, we significantly compress Ω into a list Ω' by storing only the number of occurrences (frequencies) of each appearing unique ion-mass. i.e. $\Omega'[i] = \{freq[i]\ ;\ i\ \epsilon[0, M - 1]\}$. Hence, the new size of $|\Omega'| = M << Xn = |\Omega|$ where M is the maximum appearing integer mass in T. Next, we successively accumulate the frequencies in Ω' to compute an equivalent list Ω'' given as $\Omega''[i] = \Omega''[i-1] + \Omega'[i-1]$. The Ω'' now corresponds to the start positions in Λ for each indexed unique ion-mass.

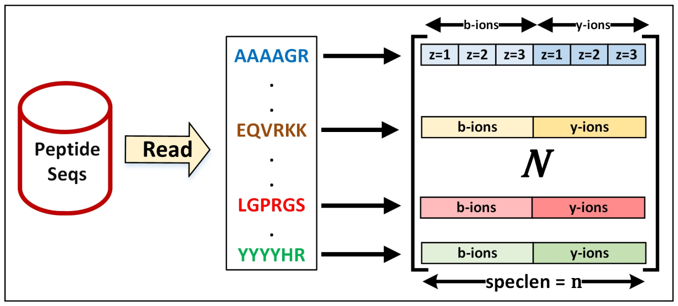


Fig. 1. The ions in each spectrum are first ordered by the ion-series (b/y) followed by fragment charge and finally by their m/z's. The spectra from each length group are then stacked in an instance of N.

The result of the above transformation on T is the two arrays, Λ and Ω''. The total size of the constructed index is approximately equal to the size of T since $|\Lambda| = Xn = |T|$ whereas $|\Omega''| = M << |T|$. The memory footprint of the index is ~4-bytes per indexed ion. The same transformation is applied on all instances of N to construct the complete fragment-ion index. The CFIR index construction method is shown in Figure 3 while an example is illustrated in Figure 4.

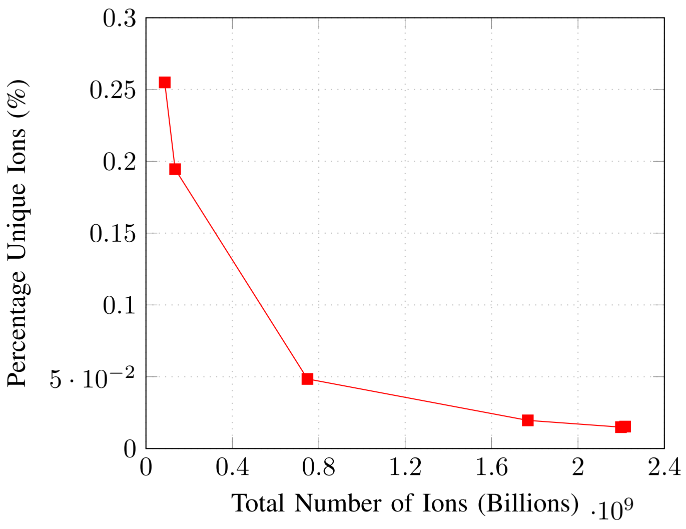


Fig. 2. The plot depicts that the ratio of unique ion masses to total number of ions in a fragment-ion is extremely small for increasing size of index.

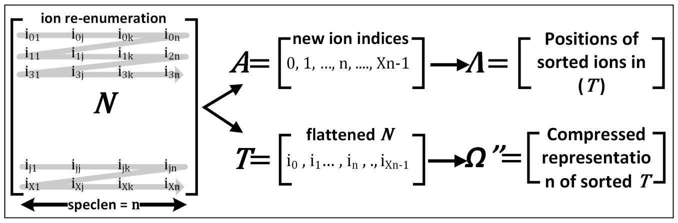
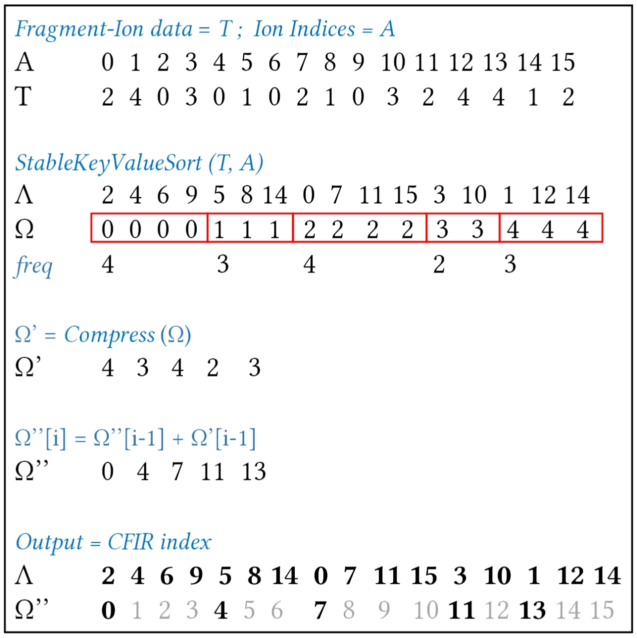


Fig. 3. The fragment ions and their positions in N are split into two arrays T and A respectively. The $StableKeyValueSort$ operation is applied on A using T as keys to obtain Λ and Ω respectively. The Ω array is then compressed into an equivalent representation Ω''.



Fragment-Ion data = T ; Ion Indices = A

A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	2	4	0	3	0	1	0	2	1	0	3	2	4	4	1	2

StableKeyValueSort (T, A)

Λ	2	4	6	9	5	8	14	0	7	11	15	3	10	1	12	14
Ω	0	0	0	0	1	1	1	2	2	2	2	3	3	4	4	4
freq	4				3			4				2		3		

Ω' = Compress (Ω)

Ω' 4 3 4 2 3

Ω''[i] = Ω''[i-1] + Ω'[i-1]

Ω'' 0 4 7 11 13

Output = CFIR index

Λ	**2**	**4**	**6**	**9**	**5**	**8**	**14**	**0**	**7**	**11**	**15**	**3**	**10**	**1**	**12**	**14**
Ω''	**0**	1	2	3	**4**	5	6	**7**	8	9	10	**11**	12	**13**	14	15

Fig. 4. Illustrates an example of CFIR index construction for fragment-ion data in the range $[0, 4]$. Notice that the output list Ω'' contains only the bolded black entries while the greyed ones are omitted.

C. Querying CFIR and Shared-Peak Counting

The shared-peaks between an experimental (query) spectrum and the indexed spectra can be counted by first locating occurrences of all query fragment-ion (peak) in T. The occurrence locations in T are then mapped to their origin spectrum IDs ($osid$) in N followed by counting the number of times a theoretical spectrum's ID ($osid$) shows up. The shared peak counting algorithm can be translated in terms of the two index query operations, $rank$ and $select$, along with $backtrack$ and $update\_score$ as shown in Algorithm 1. Next we discuss how the two query operations performed using the CFIR representation of fragment-ion index.

- $rank(T, mz)$: The number of occurrences of fragment-ion with m/z (mz) in T can be computed as $\Omega''[mz+1] - \Omega''[mz]$ which is done in $O(1)$ time.
- $select(T, mz, k)$: The k^{th} occurrence of fragment-ion with m/z (mz) in T can be located at $\Lambda[\Omega''[mz]+k]$ whereas all occurrences can be located at locations between: $[\Lambda[\Omega''[mz]],$ $\Lambda[\Omega''[mz]+rank(T,mz)-1]]$ which is also done in $O(1)$ time as well.

The Λ elements accessed in $select$ are the ion-positions (pos) in T, which can be mapped to origin-spectrum ID ($osid$), ion-charge ($ichg$) and ion-series ($iser$) information in constant time using the ion ordering information by the $backtrack(pos)$ function depicted in Algorithm 2. An example of $rank$ and $select$ is illustrated in Figure 5. Note that the $update\_score$ function may have different implementations depending on the employed scoring algorithm. In our implementation, we adopted the MSFragger's hyperscore function that computes the similarity score between a spectral pair j and k given as:

$$hyperscore(j,k) = \log(n_b!n_y!\Sigma I_b \Sigma I_y)$$

where n_b and n_y correspond to the count of shared b- and y-ions between the spectral pair while ΣI_b and ΣI_y represents the sum of (normalized) intensities of the shared peaks. In our implementation of $update\_score$, we use the query ion information (qm), the matched origin spectrum's ID ($osid$), and the matched ion series information ($iser$) and update the four individual components ($n_b, n_y, \Sigma I_b, \Sigma I_y$) of hyperscore which are processed at the end of fragment-ion index look up to obtain final results.

Algorithm 1: Shared Peak Counting

```
Data: Query fragment mass ($qm$)
Result: Shared Peak Scorecard
  /* Get the range of occurrences of $qm$ */
1 $occs = rank(T, qm)$;
  /* Iterate over the range */
2 for $k$ in $occs$ do
      /* Locate each occurrence */
3     $pos = select(T, qm, k)$;
      /* Backtrack origin spectrum ID */
4     $osid, inum, ichg, iser = backtrack(pos)$;
      /* Update shared peak count */
5     $update\_score(qm, osid, iser, +1)$;
```

IV. Results

A. Memory Footprint

We compared the memory footprint results for CFIR fragment-ion index against MSFragger, ANN-SoLo and SpecOMS index for varying data sizes. The index size for MSFragger and CFIR fragment-ion index was varied by increasing the number and types of modifications incorporated in the protein database. The index was constructed using Human proteome database plus reverse decoys and cRAP contaminants. The index size for ANN-SoLo was varied by using spectral libraries of different sizes at the input. We used the Human Orbitrap spectral library from ISB and Mouse spectral library from NIST for our experiments with ANN-SoLo. The index size for SpecOMS was varied by increasing

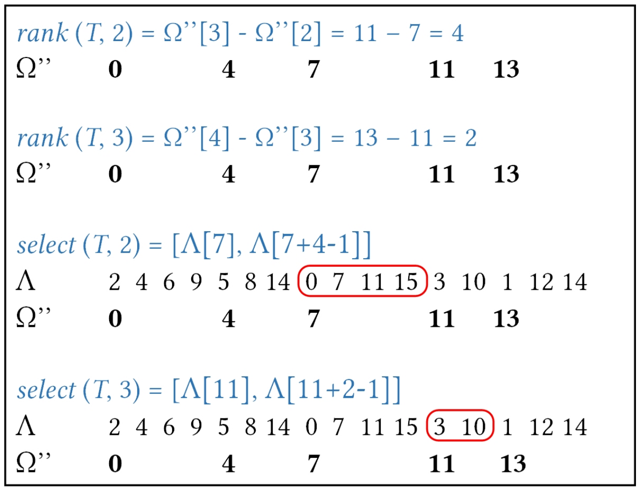

rank (T, 2) = Ω''[3] - Ω''[2] = 11 − 7 = 4
Ω'' 0 4 7 11 13
rank (T, 3) = Ω''[4] - Ω''[3] = 13 − 11 = 2
Ω'' 0 4 7 11 13
select (T, 2) = [Λ[7], Λ[7+4-1]]
Λ 2 4 6 9 5 8 14 0 7 11 15 3 10 1 12 14
Ω'' 0 4 7 11 13
select (T, 3) = [Λ[11], Λ[11+2-1]]
Λ 2 4 6 9 5 8 14 0 7 11 15 3 10 1 12 14
Ω'' 0 4 7 11 13

Fig. 5. Illustrates an example of *rank* and *select* for two integer query fragment-ions with m/z 2 and 3.

Algorithm 2: Backtracking Algorithm

```
Data: Fragment-Ion position in $T$ ($pos$), spectrum length ($len$), max ion-charge in index ($z_{max}$)
Result: Fragment-Ion information in $N$
/* Assuming $dim(N) = (X \times len)$ */
/* Origin spectrum ID */
1 $osid = \lfloor pos/len \rfloor$;
/* ion-number in respective (b/y) ion series */
2 $inum = (pos \mod (len/2)) + 1$;
/* ion-charge in origin spectrum */
3 $ichg = ((inum - 1)/(len/2z_{max}) + 1)$;
/* ion-series in origin spectrum */
4 if $(pos \mod len) \leq (len/2)$ then
5     $iser = b\_ion$;
6 else
7     $iser = y\_ion$;
8 return $osid, inum, ichg, iser$;
```

the number of input protein sequences obtained by sequentially concatenating UniProt Homo sapiens (UP000005640), Rabbit (UP000001811), Drosophilia (UP000000803) and Dog (UP000002254) proteome databases plus reversed decoys and cRAP contaminants at its input.

Further, since SpecOMS only works on Windows, all experiments were run on a Windows machine equipped with 20GB RAM. The memory footprint for MSFragger was recorded from its execution logs since MSFragger usually splits its peptide index into smaller chunks which are stored onto the disk while the fragment index chunks are generated on the fly for each peptide index chunk. The recorded results for MSFragger do not include the disk memory for MSFragger Peptide Index. The results show that the CFIR based fragment-ion index consumes at most half the memory as MSFragger fragment-ion index and orders-of-magnitude improvement over ANN-SoLo and SpecOMS index as depicted in Figure 6. Further, we extended the memory footprint results for MSFragger and CFIR fragment-ion index for large data using a powerful server machine equipped with 128GB of RAM as seen in Figure 7.

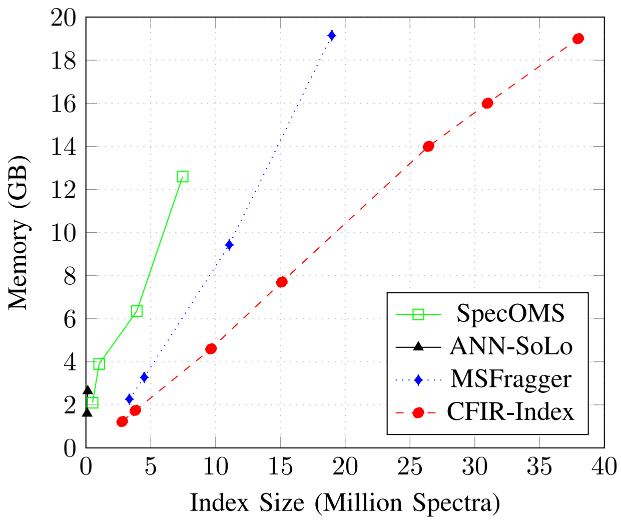


Fig. 6. The results show that CFIR data structure can index about 40 million spectra within 20GB RAM outperforming existing techniques by a factor of at least 2×.

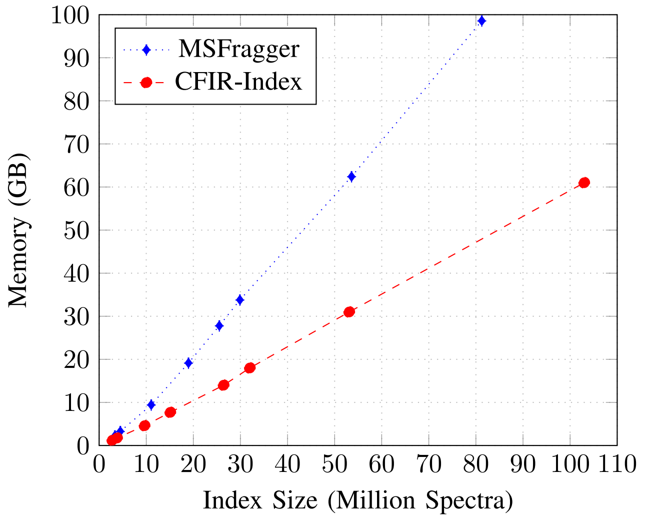


Fig. 7. Extended memory footprint results for MSFragger and CFIR fragment-ion index show a consistent 2× improvement for CFIR over MSFragger.

B. Indexing Time

We compared the index construction time for CFIR-Index against MSFragger, ANN-SoLo and SpecOMS. Figure 8 shows indexing speed for the three indexing methods with varying index size. The results show that both MSFragger and CFIR Index have the same time complexity i.e. $O(N \log N)$ since the main operation performed is a (stable) sort in both methods. However, since the index construction usually consumes ∼5% of the total experiment time in normal cases, it can be avoided in favor of much better memory efficiency.

Further, it can be seen that the time complexity for both SpecOMS and ANN-SoLo varies *exponentially* since both methods involve spectral clustering and tree/nearest-neighbor construction which are highly compute intensive.

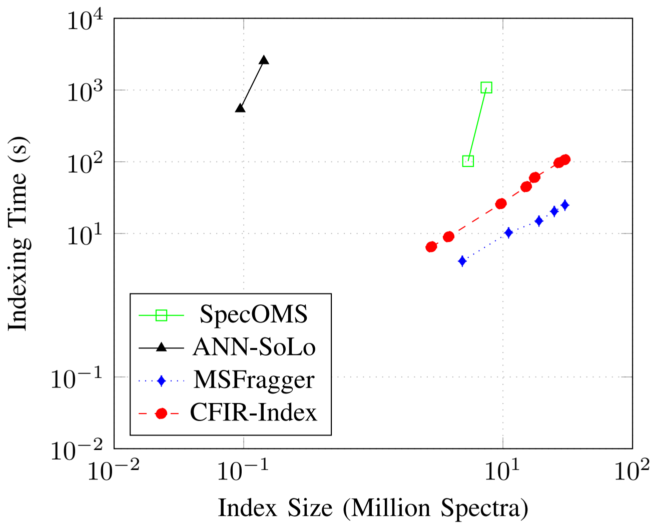


Fig. 8. The results for index construction time show that both MSFragger and CFIR index exhibit same time complexity while SpecOMS and ANN-SoLo exhibit exponential time complexity.

C. Peptide Search Time

We analyzed the peptide search time (shared peak counting + hyperscore computations) for CFIR index based algorithm for varying MS/MS data and index sizes. We employed the PRIDE Archive data set (PXD009072) containing about 305,000 MS/MS spectra searched against searched against CFIR index constructed from UniProtKB Homo sapiens protein sequence database (UP000005640). The database digestion settings were set to: tryptic digestion with 1 allowed missed cleavages, peptide lengths from 6 to 40, peptide masses from 100 to 5000amu, duplicate peptide sequences removed. The variable modifications including Methionine oxidation, gly-gly adducts on Cysteine and Lysine, and deamidation on Asparagine and Glutamine residues were added successively in the search keeping static cysteine Carbamidomethylation modification in all experiments. The search settings were as follows: resolution factor $1/r = 0.01Da$, maximum ion charge $z = +3$, peptide precursor mass tolerance $\Delta M = \pm 500Da$, and fragment mass tolerance $\Delta F = \pm 0.05Da$. The results depicted a roughly linear trend with increase in MS/MS data confirming the constant time complexity for peptide search speed as shown in Figure 9. Note that the peptide search time cannot be perfectly linear because the number of potential matches in database per query MS/MS spectrum are variable but for large database and large index size the average matches per query can be assumed to be roughly constant. Further, we analyzed the speedup scalability with respect to increasing number of cores and the results depicted a linear trend similar to that of MSFragger and other leading fragment-ion index based search engines as shown in Figure 10.

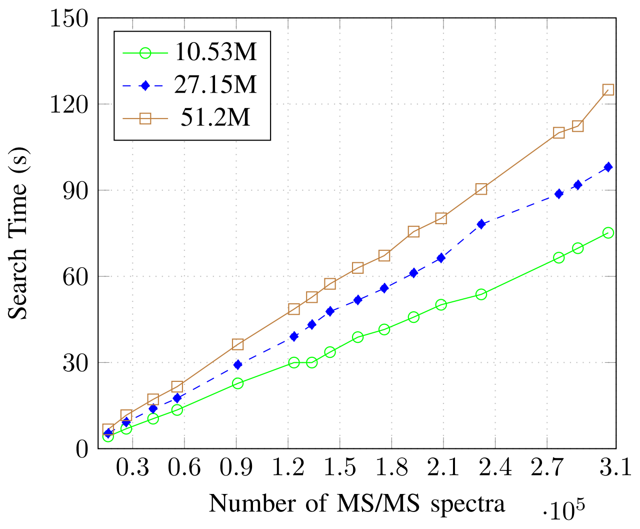


Fig. 9. The search time plot shows an almost linear trend with increasing number query MS/MS spectra for given index size indicating near constant search time per spectrum (slope).

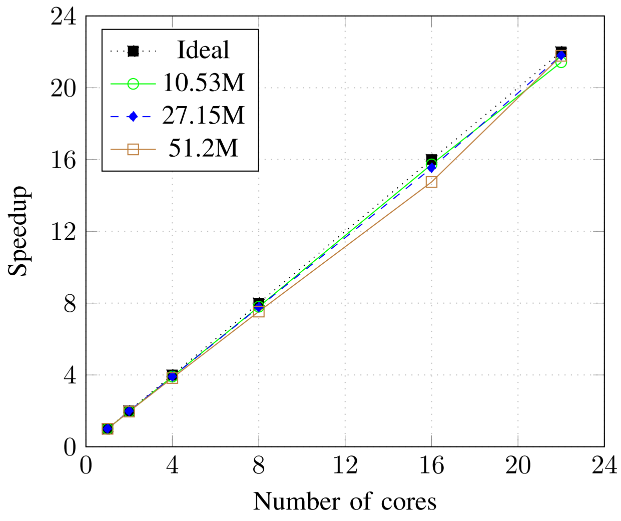


Fig. 10. The results show linear speedup scalability with number of cores for peptide search time irrespective of CFIR-index size.

D. Performance Scalability

We analyzed the peptide search performance scalability with increasing fragment-ion data. The experiments were performed on a workstation machine equipped with 8 cores $\times$ 2 sockets connected to 2 NUMA nodes (16 GB each). The experiments were performed by searching the file: FL0320_MSQ805_IBombik_Stimb_6ul.ms2 (17,595 spectra) from the PXD009072 data set against index of increasing sizes constructed by successively incorporating post-translationally modified theoretical peptides in the index. The index was allowed to span across NUMA nodes while the number of parallel cores used in each experiment was fixed to 8 (1 socket). The results in Figure 11 shows a performance slowdown beyond index size of 16 million for MSFragger (∼12% at 32 million size). The slowdown is caused by non-local memory accessing across NUMA nodes as also discussed in [10]. However, the CFIR-index based search shows optimal performance up to index size of 32 million indicating superior performance per compute unit (machine).

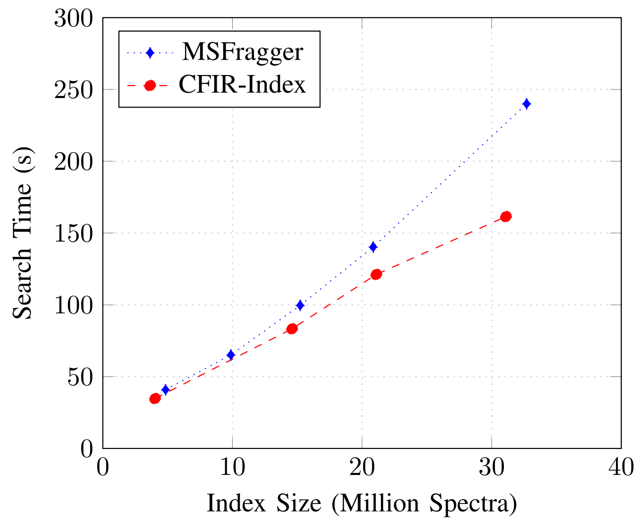


Fig. 11. The plot show a slowdown for MSFragger beyond index size of 16 million due to NUMA while CFIR-Index performs optimally up to 32 million index size

V. Discussion

The results obtained for CFIR data structure indicate that it reduces the memory footprint for the fragment-ion index while performing comparably with state of the art in terms of index construction and search speed. The results further depict that the CFIR data structure based solution runs at optimal performance for at least 2× the index size per compute unit (NUMA node). The modern mass-spectrometry instruments can produce data at astonishing speed and there has been significant research effort in algorithmic development for peptide identification. However, most of the effort has not focused on improving algorithm performance scalability and resource efficiency with increasing problem size. With the advent of high-throughput mass spectrometers the peptide search algorithms must utilize the compute, memory and network resources from ubiquitous multicore and multiprocessor architectures for optimal performance. Further, the results shown in Figure 11 also conclusively show that the CFIR data structure based solutions can significantly reduce the required number of compute nodes (units), and therefore the cost for data processing.

VI. Conclusion

In this paper, we presented a *lossless* and *compact* data structure of fragment-ion index called Compact Fragment-Ion Index Representation (CFIR). The proposed data structure leverages the low entropy in ion-mass information to compress the size of constructed index. Our experiments show that the proposed fragment-ion data structure outperforms all existing fragment-ion indexing structures in terms of memory consumption while providing the essential index query functions including $rank$ and $select$ at no additional computational costs. Our experimental results also indicate that the CFIR fragment-ion index beats the all existing fragment-ion structures by at least 2× in memory while exhibiting same time complexity for data indexing and querying as existing leading algorithms. The results also depict that the CFIR based solutions can optimally process 100% more fragment-ion than best existing data structure significantly reducing the required resources and cost for data processing. Since the proposed data structure allows efficient in-core analysis of more fragment-ion data per unit memory, it will be instrumental in development of high performance algorithmic solutions and software for peptide identification in proteomics and proteogenomics.

Funding

This research was supported by National Science Foundations (NSF) under Award Numbers NSF CRII CCF-1855441, and NSF CAREER OAC-1925960. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Science Foundation.

References

[1] J. K. Eng, B. C. Searle, K. R. Clauser, and D. L. Tabb, "A face in the crowd: recognizing peptides through database search," *Molecular & Cellular Proteomics*, pp. mcp–R111, 2011.
[2] D. H. May, K. Tamura, and W. S. Noble, "Param-medic: A tool for improving ms/ms database search yield by optimizing parameter settings," *Journal of proteome research*, vol. 16, no. 4, pp. 1817–1824, 2017.
[3] M. G. Awan and F. Saeed, "Ms-reduce: An ultrafast technique for reduction of big mass spectrometry data for high-throughput processing," *Bioinformatics*, vol. 32, no. 10, pp. 1518–1526, 2016.
[4] Z.-F. e. Yuan, C. Liu, H.-P. Wang, R.-X. Sun, Y. Fu, J.-F. Zhang, L.-H. Wang, H. Chi, Y. Li, L.-Y. Xiu, *et al.*, "pparse: A method for accurate determination of monoisotopic peaks in high-resolution mass spectra," *Proteomics*, vol. 12, no. 2, pp. 226–235, 2012.
[5] J. K. Eng, T. A. Jahan, and M. R. Hoopmann, "Comet: an open-source ms/ms sequence database search tool," *Proteomics*, vol. 13, no. 1, pp. 22–24, 2013.
[6] W. Bittremieux, P. Meysman, W. S. Noble, and K. Laukens, "Fast open modification spectral library searching through approximate nearest neighbor indexing," *bioRxiv*, p. 326173, 2018.
[7] M. David, G. Fertin, H. Rogniaux, and D. Tessier, "Specoms: a full open modification search method performing all-to-all spectra comparisons within minutes," *Journal of proteome research*, vol. 16, no. 8, pp. 3030–3038, 2017.
[8] J. Griss, Y. Perez-Riverol, S. Lewis, D. L. Tabb, J. A. Dianes, N. del Toro, M. Rurik, M. Walzer, O. Kohlbacher, H. Hermjakob, *et al.*, "Recognizing millions of consistently unidentified spectra across hundreds of shotgun proteomics datasets," *Nature methods*, vol. 13, no. 8, p. 651, 2016.
[9] J. M. Chick, D. Kolippakkam, D. P. Nusinow, B. Zhai, R. Rad, E. L. Huttlin, and S. P. Gygi, "A mass-tolerant database search identifies a large proportion of unassigned spectra in shotgun proteomics as modified peptides," *Nature biotechnology*, vol. 33, no. 7, p. 743, 2015.
[10] A. T. Kong, F. V. Leprevost, D. M. Avtonomov, D. Mellacheruvu, and A. I. Nesvizhskii, "Msfragger: ultrafast and comprehensive peptide identification in mass spectrometry–based proteomics," *Nature methods*, vol. 14, no. 5, p. 513, 2017.

[11] B. J. Diament and W. S. Noble, "Faster sequest searching for peptide identification from tandem mass spectra," *Journal of proteome research*, vol. 10, no. 9, pp. 3871–3879, 2011.

[12] J. Zhang, L. Xin, B. Shan, W. Chen, M. Xie, D. Yuen, W. Zhang, Z. Zhang, G. A. Lajoie, and B. Ma, "Peaks db: de novo sequencing assisted database search for sensitive and accurate peptide identification," *Molecular & Cellular Proteomics*, vol. 11, no. 4, pp. M111–010587, 2012.

[13] A. Devabhaktuni, S. Lin, L. Zhang, K. Swaminathan, C. G. Gonzalez, N. Olsson, S. M. Pearlman, K. Rawson, and J. E. Elias, "Taggraph reveals vast protein modification landscapes from large tandem mass spectrometry datasets," *Nature biotechnology*, p. 1, 2019.

[14] H. Chi, C. Liu, H. Yang, W.-F. Zeng, L. Wu, W.-J. Zhou, X.-N. Niu, Y.-H. Ding, Y. Zhang, R.-M. Wang, *et al.*, "Open-pfind enables precise, comprehensive and rapid peptide identification in shotgun proteomics," *bioRxiv*, p. 285395, 2018.

[15] D. H. Lundgren, D. K. Han, and J. K. Eng, "Protein identification using turbosequest," *Current protocols in bioinformatics*, vol. 10, no. 1, pp. 13–3, 2005.

[16] J. K. Eng, B. Fischer, J. Grossmann, and M. J. MacCoss, "A fast sequest cross correlation algorithm," *Journal of proteome research*, vol. 7, no. 10, pp. 4598–4602, 2008.

[17] C. Y. Park, A. A. Klammer, L. Kall, M. J. MacCoss, and W. S. Noble, "Rapid and accurate peptide identification from tandem mass spectra," *Journal of proteome research*, vol. 7, no. 7, pp. 3022–3027, 2008.

[18] S. Kim and P. A. Pevzner, "Ms-gf+ makes progress towards a universal database search tool for proteomics," *Nature communications*, vol. 5, p. 5277, 2014.

[19] L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant, "Open mass spectrometry search algorithm," *Journal of proteome research*, vol. 3, no. 5, pp. 958–964, 2004.

[20] K. R. Clauser, P. Baker, and A. L. Burlingame, "Role of accurate mass measurement ($\pm$10 ppm) in protein identification strategies employing ms or ms/ms and database searching," *Analytical chemistry*, vol. 71, no. 14, pp. 2871–2882, 1999.

[21] D. N. Perkins, D. J. Pappin, D. M. Creasy, and J. S. Cottrell, "Probability-based protein identification by searching sequence databases using mass spectrometry data," *ELECTROPHORESIS: An International Journal*, vol. 20, no. 18, pp. 3551–3567, 1999.

[22] J. K. Eng, A. L. McCormack, and J. R. Yates, "An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database," *Journal of the American Society for Mass Spectrometry*, vol. 5, no. 11, pp. 976–989, 1994.

[23] R. Craig and R. C. Beavis, "Tandem: matching proteins with tandem mass spectra," *Bioinformatics*, vol. 20, no. 9, pp. 1466–1467, 2004.

[24] U. Keich, K. Tamura, and W. Noble, "An averaging strategy to reduce variability in target-decoy estimates of false discovery rate," *bioRxiv*, p. 440594, 2018.

[25] M. Brosch, L. Yu, T. Hubbard, and J. Choudhary, "Accurate and sensitive peptide identification with mascot percolator," *Journal of proteome research*, vol. 8, no. 6, pp. 3176–3181, 2009.

[26] S. McIlwain, K. Tamura, A. Kertesz-Farkas, C. E. Grant, B. Diament, B. Frewen, J. J. Howbert, M. R. Hoopmann, L. Käll, J. K. Eng, *et al.*, "Crux: rapid open source protein tandem mass spectrometry analysis," *Journal of proteome research*, vol. 13, no. 10, pp. 4488–4491, 2014.

[27] J. Cox, N. Neuhauser, A. Michalski, R. A. Scheltema, J. V. Olsen, and M. Mann, "Andromeda: a peptide search engine integrated into the maxquant environment," *Journal of proteome research*, vol. 10, no. 4, pp. 1794–1805, 2011.

[28] M. Mann and M. Wilm, "Error-tolerant identification of peptides in sequence databases by peptide sequence tags," *Analytical chemistry*, vol. 66, no. 24, pp. 4390–4399, 1994.

[29] D. L. Tabb, A. Saraf, and J. R. Yates, "Gutentag: high-throughput sequence tagging via an empirically derived fragmentation model," *Analytical chemistry*, vol. 75, no. 23, pp. 6415–6421, 2003.

[30] S. Dasari, M. C. Chambers, S. G. Codreanu, D. C. Liebler, B. C. Collins, S. R. Pennington, W. M. Gallagher, and D. L. Tabb, "Sequence tagging reveals unexpected modifications in toxicoproteomics," *Chemical research in toxicology*, vol. 24, no. 2, pp. 204–216, 2011.

[31] S. Dasari, M. C. Chambers, R. J. Slebos, L. J. Zimmerman, A.-J. L. Ham, and D. L. Tabb, "Tagrecon: high-throughput mutation identification through sequence tagging," *Journal of proteome research*, vol. 9, no. 4, pp. 1716–1726, 2010.

[32] B. C. Searle, S. Dasari, P. A. Wilmarth, M. Turner, A. P. Reddy, L. L. David, and S. R. Nagalla, "Identification of protein modifications using ms/ms de novo sequencing and the opensea alignment algorithm," *Journal of proteome research*, vol. 4, no. 2, pp. 546–554, 2005.

[33] S. Tanner, H. Shu, A. Frank, L.-C. Wang, E. Zandi, M. Mumby, P. A. Pevzner, and V. Bafna, "Inspect: identification of posttranslationally modified peptides from tandem mass spectra," *Analytical chemistry*, vol. 77, no. 14, pp. 4626–4639, 2005.

[34] W. H. Tang, B. R. Halpern, I. V. Shilov, S. L. Seymour, S. P. Keating, A. Loboda, A. A. Patel, D. A. Schaeffer, and L. M. Nuwaysir, "Discovering known and unanticipated protein modifications using ms/ms database searching," *Analytical Chemistry*, vol. 77, no. 13, pp. 3931–3946, 2005.

[35] D. Beyter, M. S. Lin, Y. Yu, R. Pieper, and V. Bafna, "Proteostorm: An ultrafast metaproteomics database search framework," *Cell systems*, vol. 7, no. 4, pp. 463–467, 2018.

[36] S. R. Ramakrishnan, R. Mao, A. A. Nakorchevskiy, J. T. Prince, W. S. Willard, W. Xu, E. M. Marcotte, and D. P. Miranker, "A fast coarse filtering method for peptide identification by mass spectrometry," *Bioinformatics*, vol. 22, no. 12, pp. 1524–1531, 2006.

[37] M. Bern, Y. Cai, and D. Goldberg, "Lookup peaks: a hybrid of de novo sequencing and database search for protein identification by tandem mass spectrometry," *Analytical chemistry*, vol. 79, no. 4, pp. 1393–1400, 2007.

[38] H. Chi, K. He, B. Yang, Z. Chen, R.-X. Sun, S.-B. Fan, K. Zhang, C. Liu, Z.-F. Yuan, Q.-H. Wang, *et al.*, "pfind–alioth: A novel unrestricted database search algorithm to improve the interpretation of high-resolution ms/ms data," *Journal of proteomics*, vol. 125, pp. 89–97, 2015.

[39] Y. Li, H. Chi, L.-H. Wang, H.-P. Wang, Y. Fu, Z.-F. Yuan, S.-J. Li, Y.-S. Liu, R.-X. Sun, R. Zeng, *et al.*, "Speeding up tandem mass spectrometry based database searching by peptide and spectrum indexing," *Rapid Communications in Mass Spectrometry*, vol. 24, no. 6, pp. 807–814, 2010.

[40] A. Buluc and J. R. Gilbert, "On the representation and multiplication of hypersparse matrices," in *2008 IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–11, IEEE, 2008.

[41] D. L. Tabb, M. J. MacCoss, C. C. Wu, S. D. Anderson, and J. R. Yates, "Similarity among tandem mass spectra from proteomic experiments: detection, significance, and utility," *Analytical chemistry*, vol. 75, no. 10, pp. 2470–2477, 2003.