Stack-ZDPD: A Novel Encoding Scheme for Mass Spectrometry Data

Jinyin Wang (✉ wangjinyin@westlake.edu.cn)

Westlake University

Miaoshan Lu

Westlake University

Ruiming Wang

Westlake University

Shaowei An

Westlake University

Cong Xie

Shandong First Medical University

Changbin Yu

Shandong First Medical University

Research Article

Keywords: raw mass spectrometry, proteomics, metabolomics data, decompression, Stack-ZDPD, Flight technology

Posted Date: October 15th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-959796/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Abstract

As the pervasive, standardized format for interchange and deposition of raw mass spectrometry (MS) proteomics and metabolomics data, text-based mzML is inefficiently utilized on various analysis platforms due to its sheer volume of samples and limited read/write speed. Database-developed solution guarantees the efficiency of random file reading, but nevertheless the efforts in compression and third-party software support are insufficient, while most research on compression algorithms rarely provides flexible random file reading scheme. Under the premise of ensuring the efficiency of decompression, we propose an encoding scheme “Stack-ZDPD” that is optimized for storage of raw MS data, designed for the format “Aird”, a computation-oriented format with fast accessing and decoding time, where the core compression algorithm is “ZDPD”. Stack-ZDPD reduces the volume of data stored in mzML format by around 80% or more, depending on the data acquisition pattern, and the compression ratio is approximately 30% compared to ZDPD for data generated using Time of Flight technology. Our approach is available on Airdpro, for file conversion and the Java-API Aird SDK, for data parsing.

Introduction

Currently, the most widely used format for long-term storage in mass spectrometry (MS) is original proprietary vendor format. Pitifully, this kind of format cannot meet the needs of cross-platform computing and is lack of adaptability to kinds of software. The Proteomics Standards Initiative (PSI) established a standardized Extensible Markup Language (XML) representation for raw data interchange in mass spectrometry (MS) in 2008¹, called “mzML”, further building upon concepts defined in mzData and mzXML². mzML is now the pervasive format for interchange and deposition of raw MS proteomics and metabolomics data³. However, in the design of “mzML”, like all text-based XML file formats, numeric spectrum data should be converted into text strings using Base64 encoding⁴. Although the numeric data can be Zlib⁵ compressed before encoding, the sizes of mzML files are still 4 to 18-fold higher than vendor formats. That is, the memory of a typical desktop computer cannot handle the analysis of even a single proteomics file. Research projects on MS data compression have sprung up, to address difficulties arise in usage of mzML format for large-volume data analysis.

One approach for biological sampled, high-throughput data analysis is random access file reading, allowing specific data to be efficiently extracted, thus there is no need to read an entire file into memory. HDF5⁶(Hierarchical Data Format version 5) is a binary format developed by the National Center for Supercomputing Applications (NCSA) for the storage and organization of large amount of data. Mz5⁵ and Toffee⁷ are both developed on HDF5 technology. HDF5 allows multidimensional arrays of data elements of a specific type (e.g., integer, floating point, characters, strings, or a collection of these organized as compound types), bringing an average file size reduction of 54% and increases linear read and write speeds 3–4-fold according to the comparison of mz5 and mzML. Another similar database-based solution is mzDB⁸, which is developed upon the lightweight SQLite relational database technology. In comparison with XML formats, mzDB saves 25% of storage space. In addition, data access times are

cut in half or less when compared to mzML, counting on the data access mode. However, one of the bottlenecks of these database-based format is that data written in mz5 or mzDB precludes a Java implementation using the corresponding Java application programming interface (API) as compound structures are extremely slow to access with API.

Different from mz5, mzDB, and Toffee, Numpress⁹ and MassComp¹⁰ are encoding schemes for mzML, focusing on a novel algorithm to compress the binary data in the mzML file before Base64 encoding. Numpress was described to compress mzML file size by around 61%, even approximately 86% if additionally Zlib compressed. Though Numpress have better performance in compression than database-based formats, there is no flexible random file reading scheme provided in the design of Numpress. Not alone, so is MassComp.

Aird¹¹ is developed considering both efficient data reading and compression performance. It is a computation-oriented format targets for fast accessing and decoding time, providing controllable precision and multiple index strategies to reconstruct the mass spectrum file. For metadata, Aird uses JavaScript Object Notation (JSON) rather than XML to reach a lightweight metadata file with similar readability and extensibility as XML. Also, JSON has better performance in web-side development. That is JSON can better adapt to the cloud service support of MS data. For spectrometry data, Aird applies a novel compression method ZDPD¹¹ (Zlib, diff, pFor and delta). Floating-point mass-to charge ratio(m/z) array is converted to ordered integer array to use FastPfor¹² to compress. FastPfor is a library with integer compression schemes. It is broadly applicable to the compression of arrays of 32-bit integers. The library exploits the Single Instruction Multiple Data (SIMD)¹² instructions to achieve faster compression and decompression. After additionally Zlib compressing, Aird can reduce 53% of its volume when using 1 part per million(ppm) as the precision requirement, but only takes 33% of the time for decoding, compared to using Zlib only.

On the basis of ZDPD, we developed Stack-ZDPD(SZDPD) to achieve higher compression ratio. Considering that pFor compresses smaller and more repetitive integer arrays better, m/z arrays of several spectrums are joined before compression. Storage volume of m/z array reduces at the cost of the addition of tag array, where each m/z corresponds to a tag of a spectrum for decoding. Taken together, when appropriate layers of spectrums are stacked, the overall compression rate can be improved.

Results

We firstly analyze the effect of the number of spectra in a stack on the compression rate of m/z array applied SZDPD. Given that tag of 2^n spectrums require n bits for storage, MS data of the first 5 MS2 blocks in File 8(shown in Table 1.) is used to test SZDPD of different n , and the comparison result is shown in Figure 1. Overall, as n increases, SZDPD performs better first and then worse. As for its improvement relative to ZDPD, the pattern is the same. The area marked with red shadow in Figure 1.A should be paid extra attention for the sum of the size of m/z and tag array decreases when n goes from 7 to 8. The compression ratio comes to the minimum when n equals to 8, mainly benefits from the

reduction of tag array size, which is marked with the blue rectangle in Figure 1.B. Merging m/z arrays results in smaller deltas, and in turn leads to a higher compression ratio for pFor, which can be seen from the decreasing size of m/z array as number of spectra in a stack grows. When n is equal to 0, a stack contains one spectrum, which we can regard as compressing with ZDPD. In Figure 1.C, decoding time is little changed as n grows. In analyzing software widely used in field of MS like OpenSWATH¹³, MZMine¹⁴, XCMS¹⁵ and so on, extracted ion chromatogram (XIC) is heavily used during calculation. Though encoding time is markedly increased, we tend to focus more on decompression time, for the decoding frequency is much higher than encoding.

Table 1

The raw MS data files used in validation, together with MS instrument and method information. Files 1–2 are from ref 16¹⁶, data files 3–4 are from ref 17¹⁷, 5 is from ref 18¹⁸, 6 is from ref 19¹⁹, 7 is from ref 20²⁰, 8–11 are from ref 21²¹ and finally 12 is from ref 22²².

File No.	MS Data File Name	Type	Vendor	Instrument
1	0530_BG_293T_1_SWATH_1	DIA	Sciex	Triple TOF5600
2	20180722_L929_test_DDA_1	DDA	Sciex	Triple TOF5600
3	QE-HFX-20190719_50cm_60min_OFe4_2	DIA	Thermo	Qbitrap Exactive
4	QE-HFX-20190719_50cm_60min_Fr1	DDA	Thermo	Qbitrap Exactive
5	S8184TPST_01	DIA	Thermo	Qbitrap Exactive
6	Negative_000333	DDA	Thermo	Qbitrap Exactive
7	SampleA_1	DDA	Sciex	Triple TOF6600
8	HYE110_TTOF6600_32fix_Igillet_I160308_001	DIA	Sciex	Triple TOF6600
9	HYE110_TTOF6600_64fix_Igillet_I160310_001	DIA	Sciex	Triple TOF6600
10	HYE124_TTOF5600_32fix_Igillet_L150206_001	DIA	Sciex	Triple TOF5600
11	HYE124_TTOF5600_64var_Igillet_L150206_007	DIA	Sciex	Triple TOF5600
12	Set 1_F2	DDA	Agilent	6550 iFunnel Q-TOF

Relative to vendor format, we compared the compression performance of mzML, ZDPD and SZDPD, and the result is shown in Figure 2. Whether data dependent acquisition (DDA) file or data independent acquisition (DIA) file, file sizes are significantly smaller when stored in the Aird format compared to mzML, even smaller than the vendor's format. Files applying SZDPD are further compressed on the basis of ZDPD, and the promotion is better in DIA files than DDA. In order to see the compression effect more clearly, we calculated the compression ratio of the new format compared to the vendor files (shown in Figure 2.B). Overall, compared with ZDPD, the compression performance of SZDPD is improved. In several files (file7, 8, 9), the promotion effect is more obvious. Given that the test files are from different databases, the main difference in the compression of different files is the instrument that generated the

data. We focus on how SZDPD improves ZDPD, and the results show that SZDPD is more suitable for SCIEX files than other vendors. Further, among SCIEX vendor files, the compression promotion of data produced by TOF6600 is better than TOF5600, and the compression rate can be increased to about 30% provided that ZDPD has a good record of compressing vendor files by about 9%.

Discussion

Applicable file characteristics for SZDPD

Since we compress the tag array by manipulating each digit to form a byte array and then applying Zlib, the duplication of each byte is highest when the tag is 8-bit, that is, a byte storage, which is a favorable characteristic for the Zlib algorithm. Based on this, we see that 2^8 spectra in a stack compress best in the test of number of spectra a stack contains. Given that, in our subsequent tests, we used the strategy of 256 spectra stacked.

We only test the files whose data acquisition modes are DDA and DIA, for the starting point of using SZDPD is based on the similarity of adjacent spectra. However, in other data acquisition modes like SRM, MRM and PRM, the similarity between adjacent MS1 and MS2 spectra is relatively small, thus there is no need to apply SZDPD. MS2 blocks in DDA files generally contains spectrums less than 256, therefore, we merely apply SZDPD to MS1 block for files in the DDA data acquisition mode.

Among the mainstream mass spectrometers that have been tested, files produce by SCIEX TOF6600 are the most suitable for application of SZDPD, followed by the SCIEX TOF5600 and Agilent 6550 iFunnel Q-TOF. Time Of Flight (TOF) technology generate MS data with more similar adjacent spectra, and TOF6600 has the performance to deliver more sensitive, larger scale quantitative data, leading to the smaller and more repeatable deltas between m/z. As a result, stack compression offers more significant benefit.

Superiority and development prospect of SZDPD

Cloud computing is growing in popularity, and MS data has been limited to large file sizes, leading to the difficulty to use the huge cloud server computing resources, especially data of proteomics. On the other hand, one of the important reasons why proteomics is difficult to build a Web-based analysis platform is that the huge data files lead to an unacceptable long analysis process, which is unfavorable to the Web-based analysis platform. Smaller files and faster data reading are the pursuits of many researchers who devote themselves to MS research. The application of ZDPD makes the mass spectral data greatly compressed, especially the data obtained by QE technique. By using SZDPD to compress the MS files, the compression level of mass spectrum data generated by TOF can reach about 30% while ensuring reasonable decompression time.

In the whole process of MS data from generation to analysis to results, information is gradually lost. What data needs to be stored and what data needs to be transmitted are interlinked. As a format intended

to reduce data storage footprint and optimize the speed of data analysis, Aird(SZDPD) has favorable compression performance and appropriate data reading speed. Although file size has been remarkably compressed, there is still room of optimization in the data transmission of mass spectrum files in the calculation process. More effort is needed to explore suitable data for various uses.

Methods

Stack-ZDPD algorithm description

A concise schematic diagram of the Stack-ZDPD algorithm is shown in Figure 3. Some characteristics of m/z array can be obtained from experimental mass spectrum data: (a) In DDA mode, the range of fragment mass is about 0~6000, and for SWATH/DIA, the range is 400~12000. Taken all, whether in metabolomics or proteomics, each m/z can be stored as integer type. (b) Numbers in m/z array are ordered for incrementation, one of the premises of using pFor library. (c) There are similar m/z in adjacent spectra, and deltas of merged array is relatively smaller and more repeated. Given that, the pFor compression algorithm performs better on merged arrays, providing room for compression optimization.

Data precision adjustment is achieved by enlarging m/z by different multiples. For example, 1 ppm (10 daltons to calculate) corresponds to m/z-accuracy of 4 decimal places (dp). Therefore, a floating-point m/z of 1 ppm is multiplied by 10^4 to store it as an integer. And for 0.1 ppm, the number is 10^5 .

MS-data structure adjustment

MS-Data structures of mzML, Aird(ZDPD) and Aird(Stack-ZDPD) are shown in Figure 4. For each spectrum, its scanning number and start position are stored in order of RT in metadata of mzML, along with MS-data arranged according to the original RT. In addition to the classical index strategy, we rearrange the spectra, and store the MS1 and MS2 spectra into MS1 and MS2 blocks respectively, according to the calculation requirements instead of RT. Further, Stack-ZDPD is applied for each MS-data block. Storage of the chromatogram and spectral data (scan start times, m/z, and intensities) is flexible and self-described in terms of floating-point precision and layout.

Material

To verify Aird(Stack-ZDPD) across a broad range of proteomics and metabolomics data sets generated in different laboratories, we select a wide variety of MS technologies and instruments from mainstream vendors. The data sets are depicted in Table 1. We test Aird(Stack-ZDPD) among different MS types, including SWATH-DIA and DDA, and from the major vendors including Thermo, Agilent, and Sciex.

Our implementation of Stack-ZDPD has been integrated on the followed two tools, available from <https://github.com/CSi-Studio/Aird-SDK> and <https://github.com/CSi-Studio/AirdPro>. (a) Aird SDK: The SDK of Aird data accessing for both Java and C# languages. This enables third-party bioinformatics tool developers to import and export data written in Aird simply using library available in Java. (b) Aird Pro: A

GUI client for Aird data converting. The vendor file reader API is from MSConvert. Hence, the proprietary raw vendor files can be directly converted into Aird(Stack-ZDPD) using the "Aird Pro" tool and MS data is conveniently accessed through Aird SDK.

Declarations

Author contributions

Jinyin Wang coded the compression and decompression algorithm in Aird SDK, performed test of stack number and file size analysis, prepared the figures and wrote the paper. Miaoshan Lu came up with the idea, coded the compression part in AirdPro and wrote the instructions. Ruiming Wang participated in compression algorithm coding. Shaowei An and Cong Xie filtered and collected the test datasets. Changbin Yu reviewed the paper. All authors read and approved the final manuscript.

Additional information

Competing interests

The authors declare no competing interests.

Availability

Aird SDK allows scholars to access MS data stored in Aird(SZDPD) efficiently, available at <https://github.com/CSi-Studio/Aird-SDK>

AirdPro can convert vendor files into Aird files, available at <https://github.com/CSi-Studio/AirdPro>

References

1. Deutsch, E. mzML: A single, unifying data format for mass spectrometer output. *Proteomics* **8**, 2776–2777 (2008).
2. Pedrioli, P. G. A. *et al.* A common open representation of mass spectrometry data and its application to proteomics research. *Nat. Biotechnol.* **22**, 1459–1466 (2004).
3. Martens, L. *et al.* mzML - A community standard for mass spectrometry data. *Mol. Cell. Proteomics* **10**, 1–7 (2011).
4. Josefsson, S. The Base16, Base32, and Base64 Data Encodings. *RFC 4648* 18 (2006).
5. Wilhelm, M., Kirchner, M., Steen, J. A. J. & Steen, H. mz5: Space- and time-efficient storage of mass spectrometry data sets. *Mol. Cell. Proteomics* **11**, 1–5 (2012).
6. Folk, M., Heber, G., Koziol, Q., Pourmal, E. & Robinson, D. An Overview of the HDF5 Technology Suite and Its Applications. In Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases. *ACM* 36–47 (2011) doi:10.1145/1966895.1966900.
7. Tully, B. Toffee – a highly efficient, lossless file format for DIA-MS. *Sci. Rep.* **10**, 1–13 (2020).

8. Bouyssié, D. *et al.* MzDB: A file format using multiple indexing strategies for the efficient analysis of large LC-MS/MS and SWATH-MS data sets. *Mol. Cell. Proteomics* **14**, 771–781 (2015).
9. Teleman, J. *et al.* Numerical compression schemes for proteomics mass spectrometry data. *Mol. Cell. Proteomics* **13**, 1537–1542 (2014).
10. Yang, R., Chen, X. & Ochoa, I. MassComp, a lossless compressor for mass spectrometry data. *BMC Bioinformatics* **20**, 1–10 (2019).
11. Miaoshan Lu, Shaowei An, Ruimin Wang, Jinyin Wang, C. Y. Aird: A computation-oriented mass spectrometry data format enables higher compression ratio and less decoding time. *bioRxiv* 2020.10.14.338921.
12. Lemire, D., Boytsov, L. & Kurz, N. SIMD compression and the intersection of sorted integers. *Softw. Pract. Exp.* **46**, 723–749 (2016).
13. Röst, H. L. *et al.* OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data. *Nat. Biotechnol.* **32**, 219–223 (2014).
14. Pluskal, T., Castillo, S., Villar-Briones, A. & Orešič, M. MZmine 2: Modular framework for processing, visualizing, and analyzing mass spectrometry-based molecular profile data. *BMC Bioinformatics* **11**, (2010).
15. Colin A. Smith, Elizabeth J. Want, Grace O’Maille, Ruben Abagyan, and G. S. XCMS: Processing Mass Spectrometry Data for Metabolite Profiling Using Nonlinear Peak Alignment, Matching, and Identification. *Anal. Chem.* **78**, 779–787 (2006).
16. Wang, D., Gan, G., Chen, X. & Zhong, C. Q. QuantPipe: A User-Friendly Pipeline Software Tool for DIA Data Analysis Based on the OpenSWATH-PyProphet-TRIC Workflow. *J. Proteome Res.* **20**, 1096–1102 (2021).
17. He, Y. *et al.* Data-Independent Acquisition Proteomics Unravels the Effects of Iron Ions on Coronatine Synthesis in *Pseudomonas syringae* pv. tomato DC3000. *Front. Microbiol.* **11**, 1–14 (2020).
18. Wang, Z. *et al.* Quantitative Proteomics Combined with Affinity MS Revealed the Molecular Mechanism of Ginsenoside Antitumor Effects. *J. Proteome Res.* **18**, 2100–2108 (2019).
19. Danne-Rasche, N., Rubenzucker, S. & Ahrends, R. Uncovering the complexity of the yeast lipidome by means of nLC/NSI-MS/MS. *Anal. Chim. Acta* **1140**, 199–209 (2020).
20. Li, Z. *et al.* Comprehensive evaluation of untargeted metabolomics data processing software in feature detection, quantification and discriminating marker selection. *Anal. Chim. Acta* **1029**, 50–57 (2018).
21. Navarro, P. *et al.* A multi-center study benchmarks software tools for label-free proteome quantification. **34**, 1130–1136 (2017).
22. Takashi, Y. *et al.* Activation of unliganded FGF receptor by extracellular phosphate potentiates proteolytic protection of FGF23 by its O-glycosylation. *Proc. Natl. Acad. Sci. U. S. A.* **166**, 11418–11427 (2019).

Figures

Figure 1

(A) Curve of MS data compression ratio-change along with n. The black line with empty circle is generated from size of data compressed with SZDPD divided by size of original arrays, and for red curve with circle, divided by size of ZDPD-compressed arrays. (B) The size of tag array and m/z array varies with n, respectively. Data compressed with SZDPD is composed of tag array and m/z array, and the change curves of the two arrays are represented by black line and red line respectively. (C) The growth curve of encoding and decoding time as n increases.

Figure 2

(A) File size of data stored with different format. For file 2, 4, 6, 7 and 12, the data acquisition mode is DDA, and the rest files are of SWATH/DIA mode. (B) Compression ratio of mzML, ZDPD and SZDPD. The ratio is calculated by $[\text{size}(\text{vendor}) - \text{size}(x)] / \text{size}(\text{vendor}) * 100\%$, where x is mzML, ZDPD and SZDPD respectively.

Figure 3

Schematic diagram of Stack-ZDPD compression principle. Floating-point m/z array is converted to ordered integer array, and then replace the later integer with delta of adjacent integers to use pFor library to compress the array. In principle of Stack-ZDPD, we merge several arrays at first and repeat the ZDPD process. Integer from spectrum 1 is in the gray cell, while integer from spectrum 2 is in the white cell. To restore the original arrays, corresponding spectrum tag should be stored. In the given example in the diagram, spectrum 1 is marked as "0", and spectrum 2, "1". The computer performs calculation in binary, thus tag of 2n spectrums require n bits for storage.

Figure 4

(A) MS-Data structures in data acquisition mode of DDA. In DDA mode, each MS1 spectra is originally followed by several relevant MS2 spectras. In data organization of Aird(ZDPD), MS1 spectras are gathered as MS1 block to accelerate the extracted ion chromatogram(XIC) calculation. According to Stack-ZDPD, each k spectras are combined before compression, so the output data is composed of several stacks, the merged spectrums. Each stack is stored along with a Zlib-compressed tag array for decoding. It should be noted that the number of spectra m_i in the MS2 block is often not very large, thus MS2 data keeps the structure in Aird(ZDPD). (B) MS-Data structures in SWATH/DIA mode. Different from DDA mode, each MS2 spectra in DIA is set to a certain m/z SWATH window and MS2 data in the same window is generally extracted for analysis. Therefore, MS2 data is stored as MS2 blocks according to its m/z SWATH window and spectrum number of each MS2 block is theoretically same as MS1 block. Further, data in each block is stored as stacks to achieve better compression performance.