Bioinformatics, 37(21), 2021, 3923–3925
doi: 10.1093/bioinformatics/btab636
Advance Access Publication Date: 3 September 2021
Applications Note

Genome analysis

mspack: efficient lossless and lossy mass spectrometry data compression

Felix Hanau [1], Hannes Röst[2] and Idoia Ochoa [3,4,*]

[1]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA, [2]Department of Molecular Genetics, Donnelly Center, University of Toronto, Toronto, ON M5S 3E1, Canada, [3]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA and and [4]Department of Electrical Engineering, University of Navarra, Tecnun, Donostia 20018, Spain

*To whom correspondence should be addressed.

Associate Editor: Peter Robinson

Received on May 3, 2021; editorial decision on August 29, 2021; revised on August 16, 2021; accepted on September 1, 2021

Abstract

Motivation: Mass spectrometry (MS) data, used for proteomics and metabolomics analyses, have seen considerable growth in the last years. Aiming at reducing the associated storage costs, dedicated compression algorithms for MS data have been proposed, such as *MassComp* and *MSNumpress*. However, these algorithms focus on either lossless or lossy compression, respectively, and do not exploit the additional redundancy existing across scans contained in a single file. We introduce *mspack*, a compression algorithm for MS data that exploits this additional redundancy and that supports both lossless and lossy compression, as well as the mzML and the legacy mzXML formats. mspack applies several preprocessing lossless transforms and optional lossy transforms with a configurable error, followed by the general purpose compressors *gzip* or *bsc* to achieve a higher compression ratio.

Results: We tested mspack on several datasets generated by commonly used MS instruments. When used with the bsc compression backend, mspack achieves on average 76% smaller file sizes for lossless compression and 94% smaller file sizes for lossy compression, as compared with the original files. Lossless mspack achieves 10–60% lower file sizes than MassComp, and lossy mspack compresses 36–60% better than the lossy MSNumpress, for the same error, while exhibiting comparable accuracy and running time.

Availability and implementation: mspack is implemented in C++ and freely available at https://github.com/fhanau/mspack under the Apache license.

Contact: idoia@illinois.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Mass spectrometry (MS) data have seen an increase in size and volume due to its importance in proteomics and metabolomics analyses. Proof of this is the several repositories that have been created in the last years, such as MassIVE (massive.ucsd.edu/), jPOSTrepo (repository.jpostdb.org), iProx (http://www.iprox.org/) or PRIDE (https://www.ebi.ac.uk/pride/), all of which are part of the ProteomeXchange (PX) consortium (Vizcaíno *et al.*, 2014). To facilitate the exchange of MS data and reduce the associated storage and bandwidth costs, availability of specialized compressors for these data is paramount.

Currently, MS data are generally stored either uncompressed or inefficiently compressed using general-purpose compressors, in the mzML (Hermjakob, 2006) and legacy mzXML (Pedrioli *et al.*, 2004) formats, which are vendor independent. These files contain, in addition to metadata related to the experiment, the information of several scans, with each scan mainly composed of (*m*/*z*)-intensity pairs. These pairs correspond to the mass to charge ratios and corresponding ion counts, respectively, and are represented as floating point values.

MSNumpress (Teleman *et al.*, 2014) and MassComp (Yang *et al.*, 2019) have been proposed for the compression of MS data in the mzML and mzXML format, respectively. Both algorithms focus mainly on the compression of the *m*/*z*-intensity pairs and compress significantly better than general purpose compressors. However, they only exploit some forms of redundancy in the data.

MSNumpress includes lossy transforms and a byte packing transform to reduce the size of each scan. It keeps the compressed scans within the XML data, making it easier for mzML tools to access the data but incurring a considerable negative impact on the compression ratio when gzip is applied (optionally) to the resulting file. MassComp applies several lossless transforms, and then uses arithmetic coding to improve compression, while separating XML metadata from scan data and using gzip as a post-processor.

We developed mspack, an algorithm tailored to MS data that supports both lossless and lossy compression. To ensure no degradation on the downstream analysis, the user can specify a maximum loss (relative or absolute) when lossy compressing the data. Contrary to previously proposed tools, mspack exploits redundancy between scans, significantly improving the compression ratio. In addition, while the provided implementation currently accepts mzXML and mzML formats, it could be easily adapted to other formats if they became available. We tested mspack on a wide range of MS data, including MS data generated by different vendors and containing single and double precision *m*/*z*-intensity pairs. In all cases, we showed that mspack with the bsc backend reduces the file size with respect to MSNumpress (for the same loss) and MassComp by 31% and 49% on average, respectively, while having a comparable running time.

2 Methods and experimental results

mspack first extracts and decodes the *m*/*z*-intensity pairs from the different scans, which are encoded in base64 in both the mzXML and mzML formats. It then applies a series of lossless transforms to make the data easier to compress, specifically, a bucket transform to reorder similar data across scans, a delta transform for the *m*/*z* data (note that *m*/*z* values are sorted in increasing order within a scan), and a byte splitting transform to exploit the floating-point format of the data. Optionally, lossy transforms using a configurable maximum absolute error for *m*/*z* data and a relative error for intensities can be applied. The transformed data is then concatenated with the XML metadata, followed by compression with bsc (https://github.com/IlyaGrebnov/libbsc/) or gzip. The gain of mspack comes from exploiting the redundancy present in the data (by means of the different transformations), such that similar data are stored together. In the case of lossy compression, there is an additional gain from reducing the precision of the *m*/*z*-intensity values (always within the allowed error), reducing their entropy and hence making them easier to compress. See Supplementary Data for a detailed description of the method.

We compared the performance of mspack to that of MSNumpress and MassComp on six mzML MS files ranging from 0.9 to 12.6 GB, generated with different instruments, and containing 64-bit *m*/*z* data and 32-bit intensity data (Supplementary Table S1). The files include both profile and centroid data as well as data with and without extra zero intensity values between non-zero values, to reflect a range of possible use cases (see Supplementary Section S2.1). For completeness, we also provide results when all data are converted to centroid (see Supplementary Section S2.4).

All results were gathered using a single thread on an Intel Xeon CPU.

For lossless compression only, we tested with the files converted to 32-bit mzXML for compatibility with MassComp. mspack with bsc is able to compress the original file by 76% on average (Supplementary Table S3). Compared to MassComp, it results in a file size reduction from about 10% to 60%. Similar results are obtained when compressing the *m*/*z*-intensity pairs only (Supplementary Table S4). mspack has an average compression speed of 16.05 MB/s using bsc while MassComp achieves 27.70 MB/s, but mspack is much faster at decompressing (Supplementary Tables S5 and S6). mspack with gzip achieves higher compression speed (27.67 MB/s) on par with MassComp, with a lower compression ratio than mspack with bsc but still better than MassComp.

For lossy compression, we tested both mspack and MSNumpress with a maximum absolute error of 10^{-4} (0.2 ppm at 500 *m*/*z*) for *m*/*z* data and a maximum relative error of 10^{-2} for intensities (see Supplementary Data for results with other errors), thus keeping compression inaccuracies below the level of expected instrument variation. We use MSNumpress with the Proteowizard suite (Chambers *et al.*, 2012), where we enabled the options to compress each scan with zlib and the entire file with gzip to improve the compression ratio. mspack compressed the data by 94% on average with the default lossy transforms and the given specified maximum errors (Supplementary Table S7). Compared to MSNumpress, file sizes were 36% to 60% smaller achieving the same accuracy.

MSNumpress compresses each scan separately, which enables random access unlike MassComp and mspack. However, this

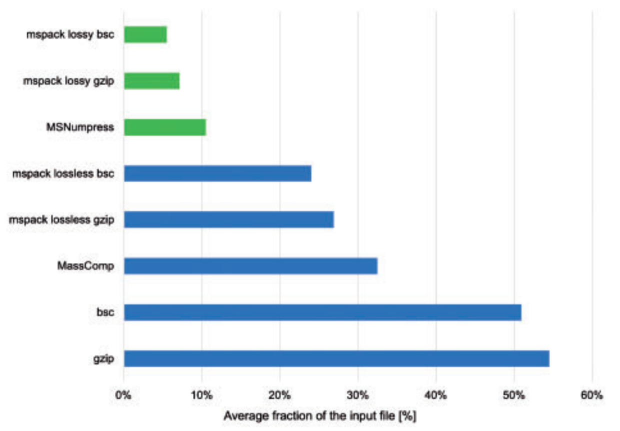

Fig. 1. Average compression performance of the tested compressors across all the considered MS files. For each compressor, we specify the average file size as the proportion of the uncompressed file size (compressed size divided by original file). For lossy compression (i.e. mspack lossy and MSNumpress, in green), we used an absolute error of 10^{-4} for *m*/*z* data and a relative error of 10^{-2} for intensity values. The lossless results (in blue) were measured using 32-bit mzXML while the lossy results were measured using 32-bit *m*/*z* data and 64-bit intensities in the mzML representation

negatively impacts the compression ratio, as the scan data is interleaved with the metadata prior to compression with gzip. When compressing only the *m*/*z*-intensity pairs, which allows to better compare the theoretical compression performance of the algorithms, mspack compresses the data 10–50% better than MSNumpress (Supplementary Table S8). mspack has an average encoding time performance of 34.7 MB/s with bsc (39.8 MB/s with gzip), being slightly faster than MSNumpress with 31.2 MB/s. For decoding, mspack with bsc is slower and with gzip slightly faster than MSNumpress (Supplementary Tables S9 and S10).

Figure 1 summarizes the results. For completeness, results with gzip and bsc are also included. In conclusion, we presented an algorithm that compresses current data by 94% (76% for lossless compression), substantially outperforming current state-of-the-art methods for lossy and lossless compression. These results will have a substantial impact on the field, increasing the ease with which data can be stored, exchanged and analyzed. It should be noted that the current implementation of mspack does not support random access, making it particularly suitable for long term storage. Nevertheless, mspack supports block-based compression and compression without the bucket transform, which can facilitate the support for random access in the future. In conclusion, we expect that our contribution will serve to substantially reduce costs for storage, facilitating proteomics workflows and analysis of data.

Funding

This work was partially funded by a Gipuzkoa Fellows award and a Ramon y Cajal grant.

Conflict of Interest: none declared.

Data availability

All the files used in this article can be found at https://doi.org/10.13012/B2IDB-1396774_V2.

References

Chambers,M.C. *et al.* (2012) A cross-platform toolkit for mass spectrometry and proteomics. *Nat. Biotechnol.*, **30**, 918–920.
Hermjakob,H. (2006) The HUPO proteomics standards initiative–overcoming the fragmentation of proteomics data. *Proteomics*, **6**, 34–38.

Pedrioli,P.G. *et al.* (2004) A common open representation of mass spectrometry data and its application to proteomics research. *Nat. Biotechnol.*, **22**, 1459–1466.

Teleman,J. *et al.* (2014) Numerical compression schemes for proteomics mass spectrometry data. *Mol. Cell. Proteomics*, **13**, 1537–1542.

Vizcaíno,J.A. *et al.* (2014) Proteomexchange provides globally coordinated proteomics data submission and dissemination. *Nat. Biotechnol.*, **32**, 223–226.

Yang,R. *et al.* (2019) Masscomp, a lossless compressor for mass spectrometry data. *BMC Bioinformatics*, **20**, 368.