BIOINFORMATICS APPLICATIONS NOTE Vol. 27 no. 8 2011, pages 1166–1167
doi:10.1093/bioinformatics/btr084

Gene expression

Advance Access publication February 16, 2011

libfbi: a C++ implementation for fast box intersection and application to sparse mass spectrometry data

Marc Kirchner[1,2,3,†,*], Buote Xu[1,2,†], Hanno Steen[1,2,3] and Judith A. J. Steen[1,4]

[1]Proteomics Center, Children's Hospital Boston, [2]Department of Pathology, Children's Hospital Boston, [3]Department of Pathology, Harvard Medical School and [4]Department of Neurobiology, Harvard Medical School and T. M. Kirby Neurobiology Center, Children's Hospital, Boston, MA 02115, USA

Associate Editor: Martin Bishop

ABSTRACT

Motivation: Algorithms for sparse data require fast search and subset selection capabilities for the determination of point neighborhoods. A natural data representation for such cases are space partitioning data structures. However, the associated range queries assume noise-free observations and cannot take into account observation-specific uncertainty estimates that are present in e.g. modern mass spectrometry data. In order to accommodate the inhomogeneous noise characteristics of sparse real-world datasets, point queries need to be reformulated in terms of box intersection queries, where box sizes correspond to uncertainty regions for each observation.

Results: This contribution introduces `libfbi`, a standard C++, header-only template implementation for fast box intersection in an arbitrary number of dimensions, with arbitrary data types in each dimension. The implementation is applied to a data aggregation task on state-of-the-art liquid chromatography/mass spectrometry data, where it shows excellent run time properties.

Availability: The library is available under an MIT license and can be downloaded from http://software.steenlab.org/libfbi.

Contact: marc.kirchner@childrens.harvard.edu

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

Received on October 29, 2010; revised on February 7, 2011; accepted on February 10, 2011

1 INTRODUCTION

Modern high-resolution liquid chromatography/mass spectrometry (LC/MS) datasets are a prominent example of data stored in a sparse representation: instead of storing vectors holding all sampled measurement values, dataset sizes are minimized by dropping zero measurements, thus abolishing underlying fixed sampling grids and their implicit neighborhood relations. This choice of representation has fundamental consequences for data processing algorithms.

For LC/MS, aggregation of raw measurements into (i) high mass accuracy centroid data; (ii) combination of centroids into extracted ion current (XIC) measurements; and (iii) determining XIC patterns that correspond to isotopic envelopes of analytes of interest, are standard preprocessing steps (Cox and Mann, 2008; Khan *et al.*, 2009).

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

Each of these steps requires fast neighborhood evaluation for hundreds of thousands of single measurements. A straightforward approach to this problem is the use of space partitioning data structures such as BSP trees, Octrees, R-trees or *kd*-trees and to evaluate neighborhood relations on the fly using data-dependent range queries (Khan *et al.*, 2009). However, real-world measurements are often subject to varying magnitudes of noise. Consequently, aggregation methods will deliver varying uncertainty estimates for e.g. calculated centroids and/or XICs. This context gives rise to a major conceptual reservation against simple range query approaches: although a range query is a natural representation for the detection of measurements that fall into the uncertainty bound of the point from which the query is issued, it cannot take into account the uncertainty of the points that should be returned by the query. Consequently, the range query assumes that the queried observations are noise free, and is bound to miss observations where points fall outside the query range but query and target uncertainty ranges overlap (Fig. 1).

The key to overcoming this limitation is to reformulate the classical range query in terms of a (potentially multi-dimensional) box intersection problem with point-specific box sizes. Observations and range queries both define a set of axis-parallel boxes, and the goal is to determine all box intersections between the sets.

This contribution introduces an implementation of a fast box intersection procedure termed `libfbi` and illustrates its application to state-of-the-art MS data.

2 METHOD AND IMPLEMENTATION

A box X is defined as the Cartesian product of half-open intervals $[l_1,u_1)\otimes[l_2,u_2)\otimes\cdots\otimes[l_D,u_D)$, where the l_d and u_d, $d\in\{1,\ldots,D\}$ are the lower and upper interval bounds in dimension d. The box intersection problem takes two sets of boxes $\mathcal{A}=\{A_i\}$, $i\in\{1,\ldots,M\}$ and $\mathcal{B}=\{B_j\}$, $j\in\{1,\ldots,N\}$ and determines an adjacency list holding the set of index pairs $\{(i,j)_k\}$, $k\in\{1,\ldots,K\}$ of intersecting boxes.

`libfbi` provides an implementation of the fast box intersection algorithm proposed in Zomorodian and Edelsbrunner (2000). The approach follows a divide and conquer scheme, iteratively separating the sets of boxes in every dimension based on implicitly constructed segment trees. Reaching the last dimension or a threshold θ in the number of boxes, the algorithm switches to a brute-force scanning procedure to avoid the comparatively large hidden constants of the segment tree. Such a hybrid approach yields $O(M+N)$ space complexity, $O((M+N)\log^D((M+N)/\theta)+K)$ time complexity for the partitioning and $O((N+M)(\log(\theta)+\theta'))$ for the scan, with $\theta'\in\{1,\ldots,\theta\}$.

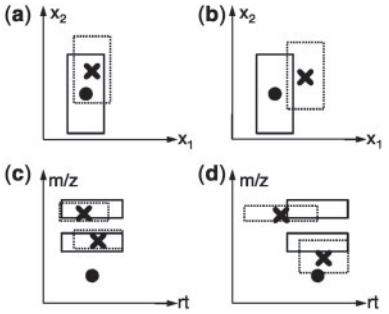

Fig. 1. Schematic illustrations of the box intersection method (**a** and **b**) and application to (pre-identification) liquid chromatography/mass spectrometry data (**c** and **d**). Points and uncertainty estimates are denoted by dashed rectangles with crosses at their center. Active range queries are shown as solid rectangles and circles. (a) The target point falls into the range query and is correctly detected by standard range queries as well as box intersection queries. (b) The target point lies outside the range query but the target point uncertainty and the query overlap. This query only succeeds for the box intersection approach. (c) Regular range query attempting to detect points that belong to an isotope pattern. Isotope candidates fall into subsequent range queries, and standard range queries are sufficient. (d) Isotope candidates are distorted. Overlapping uncertainty regions can only be detected using box intersection queries.

`libfbi` is a standard C++, template header-only implementation that makes heavy use of generic programming techniques and relies on C++0x variadic templates to support arbitrary numbers and types of dimensions. The user provides problem-specific functor classes to construct uncertainty and query intervals, and supplies suitable comparison operators, if necessary. The underlying data containers are only required to supply a forward iterator type, and their data are never copied.

3 APPLICATION

Data. LC/MS raw data were acquired from a HeLa cell lysate on an LTQ-Orbitrap (Thermo). Centroids were extracted using an in-house program and fed into the `libfbi` example application. See Supplementary Material for details.

Adjacency list construction for extracted ion current (XIC) determination. XIC construction determines groups of centroid measurements that belong to the same isotope peak (Cox and Mann, 2008). Algorithmically, this amounts to determining the connected components among the centroids, where two centroids $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ are connected if the range query for $\boldsymbol{c}_i$ intersects the uncertainty region of $\boldsymbol{c}_j$. We use `libfbi` to generate the adjacency list that serves as an input to the connected components algorithm. For illustration purposes, we ran `libfbi` with four different setups, combining two dimension ordering choices (m/z first versus retention time/scan number first) with two cutoffs $\theta_{\text{low}} = 250$ and $[\theta_{\text{high}} = 2 \times 10^6]$.

4 RESULTS

Figure 2 shows the run times for different thresholds θ and different dimension orderings on datasets of increasing size.

`libfbi` is applicable in practical settings. With average run times for $\sim 10^6$ points below 10 s, `libfbi` is well suited for adjacency list determination in MS data analysis settings and beyond (see Supplementary Material for detailed benchmark results).

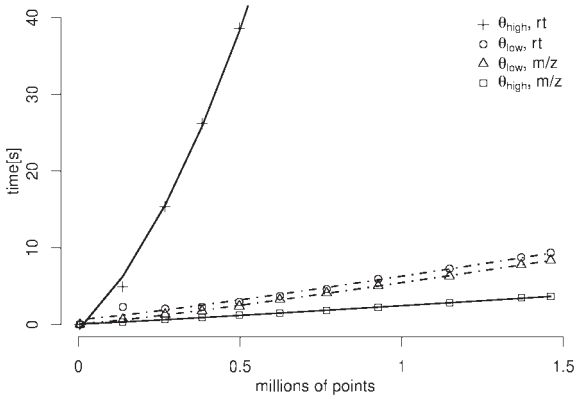

Fig. 2. `libfbi` run time analysis. The plot shows run times (in seconds) for a selection of pruning threshold θ for increasing dataset sizes and under different dimension ordering. This illustrates that `libfbi` is applicable in practical settings and that low cutoffs exhibit stable run time behavior.

Parameter selection. In `libfbi`, box dimensions are processed sequentially. The *m/z* dimension is much more discriminative, yields a smaller set of potential overlaps and hence leaves a smaller computational burden for the determination of retention time interval intersections. Hence, for θ_{high}, the `libfbi` setup with *m/z* in the first dimension runs faster (Fig. 2, crosses and squares) and the plot reveals the quadratic complexity of the underlying scanning procedure when selectivity in the first dimension is low. As a consequence, for large θ an adequate dimension ordering is mandatory. Conversely, choosing $\theta = \theta_{\text{low}}$ (Fig. 2, circles and triangles) eliminates the tremendous influence of dimension ordering. If optimization of θ is infeasible (e.g. due to significant variance in the data), low cutoffs enhance run time stability. This is the recommended *modus operandi*.

5 CONCLUSION

`libfbi` is a library for the computation of box intersections in an arbitrary number of dimensions. Application scenarios include LC/MS feature extraction, feature correspondence estimation, bounding volume determination and collision detection in geometric and image processing problems and more. `libfbi` is available from http://software.steenlab.org/libfbi.

ACKNOWLEDGEMENTS

The authors thank Dominic Winter for data acquisition.

Funding: M.K. gratefully acknowledges financial support by the Alexander von Humboldt-Foundation.

Conflict of Interest: none declared.

REFERENCES

Cox,J. and Mann,M. (2008) Maxquant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide protein quantification. *Nat. Biotechnol.*, **26**, 1367–1372.

Khan,Z. *et al*. (2009) Protein quantification across hundreds of experimental conditions. *Proc. Natl Acad. Sci. USA* , **106**, 15544–15548.

Zomorodian,A. and Edelsbrunner,H. (2000) Fast software for box intersections. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, ACM, New York, NY, USA, pp. 129–138.