ASTRONOMICAL DATA ANALYSIS SOFTWARE AND SYSTEMS XIV
ASP Conference Series, Vol. 347, 2005
P. L. Shopbell, M. C. Britton, and R. Ebert, eds.

An O($N \log M$) Algorithm for Catalogue Crossmatching

Drew Devereux, David J. Abel, Robert A. Power, and Peter R. Lamb
CSIRO ICT Centre, GPO Box 664 Canberra ACT 2601, Australia

Abstract. Catalogue cross-matching is an inherently costly operation. Our algorithm applies filter-refine and plane sweep techniques. Pre-processing consists of a sort by declination, and the active list is a queue indexed by a binary tree. The algorithm is O($N \log M$) in both I/O and processor costs, with only moderate memory requirements. Empirical assessment on catalogues of up to a billion records suggests that the algorithm performs at least an order of magnitude better than the techniques in current use.

1. Introduction

Catalogue cross matching is a fundamental operation in the virtual observatory. However, it is inherently costly, because the catalogues of interest are very large, and the matching operation is intrinsically a join. Current algorithms have costs that are quadratic with problem size and would take days to cross-match the very large catalogues being produced by current surveys.

The obvious approach to cross-matching is to match source records on the basis of positional coincidence. In this paper, we report a fast algorithm for determining the pairs of records, drawn from two catalogues, that are positionally coincident. The algorithm has preprocessing costs of O($N \log N + M \log M$), where N and M are the number of objects in the input catalogues. The matching operation proper has I/O costs of O($N+M$) and processor costs of O($(N+M) \log M$). An empirical assessment of the algorithm shows that, with modest computing equipment, matching of catalogues of a billion records can be performed in under six hours.

2. The Catalogue Cross-Matching Problem

We are given two source catalogues A and T, each with a large number of records in no particular sequence. We assume that the number of records is too large to be held in main memory. Each record specifies the apparent position of a source, taken to be a normally distributed random variable expressed as a mean location in right ascension and declination coordinates, and a standard deviation value expressed as a single angular distance. We assume that the catalogues have been standardised in their coordinate systems and epoch, so that the spatial locations are commensurable. Errors may vary between records in a catalogue.

A pair (a, t) of source records will be declared a match only if we are sufficiently confident that the true positions of a and t are the same. We can be $100\alpha\%$

confident that a and t are positionally coincident only if the angular distance between their means is no more than $z_{\alpha/2}\sqrt{\sigma_a^2 + \sigma_t^2}$.

Note that this problem definition permits the same record to occur in more than one candidate pair. Further processing may be required to further refine the set of matches, and possibly to select the best match for each record.

3. Outline of the Algorithm

Since computation of angular distance is an expensive operation, our problem definition leads to a test that is costly to perform. The key to efficient crossmatching is to perform this test as few times as possible. To achieve this, our algorithm applies two concepts. The first is to use the well-known filter-refine strategy from geospatial databases; the second is an algorithm based on the plane sweep metaphor from computational geometry.

3.1. Filter-Refine

In filter-refine strategies, we defer executing an expensive test by first performing a cheap "filter" test. The filter is designed to rapidly reject "obvious" non-matches at very low cost, but is guaranteed not to reject any true matches. Only those candidates that are accepted by the filter are subject to the full test. Our algorithm uses the filter-refine strategy by decomposing the full test into separate, and somewhat weakened, tests for right ascension and declination.

3.2. Plane Sweep

In plane sweep algorithms, a line is swept through the space, and an event is processed each time the line encounters an object of interest. Often the event processing serves to maintain an *active list*: a list of objects that are of current interest. Tests may then be restricted to members of the active list, which is typically much smaller than the full set of objects. Our algorithm adapts the plane sweep metaphor to the celestial sphere by sweeping a circle of constant declination through the space of records from catalogue T. An active list is maintained to contain those records from catalogue A that fall within the filter's declination bounds for the current record from catalogue T.

3.3. The Algorithm

As a prerequisite to sweeping by declination, both catalogues are sorted by declination. During sorting, we note the largest errors for each catalogue. We then sweep the space of records from catalogue T. As each record t is encountered, we compute upper and lower bounds on declination and right ascension for records that could match t, taking into account the catalogues' largest errors. We then update the active list to contain all and only those members of catalogue A whose mean positions fall within the declination bounds. Finally, we query the active list for all members whose mean positions fall within the right ascension bounds. For each returned record a, we perform the full test on the pair (a, t). For more detail, see Abel et al. (2004) and Devereux, Abel & Power (2004).

3.4. The Active List

The active list is a queue on declination, and a threaded binary tree on right ascension. This ensures that we can update and query the active list in $O(\log N)$ time.

The active list remains small enough to reside in main memory. For a catalogue of a billion objects and even if we assume errors of one arc minute, the active list will still only have a maximum of about 600,000 elements; the memory requirements for this are fairly modest.

4. Results

The initial sort is trivially $O(N \log N + M \log M)$. The sweep is $O(M + N)$ in I/O, and $O((N + M) \log M)$ for processing, where M is the size of the catalogue A, and N is the size of catalogue T.

Performance was empirically assessed by pairwise matching of six catalogues. Preprocessing costs are given in Table 1 and the costs of matching are given in Table 2. Benchmarking was performed on a Dual Pentium Xeon 2Ghz processor with 2Gb of main memory. The disk was five RAID5 storage arrays with 10K RPM UltraSCSI disks with a maximum transfer rate between the RAIDs and the server of 80Mb/s. For full details, see Power & Devereux (2004).

Catalogue	Times	
	elapsed	cpu
1XMM	0:02	0:01
SUMSS	0:02	0:03
Tycho2	0:34	0:33
2MASS	106:00	95:00
USNO A2	45:00	43:00
USNO B1	95:00	80:00

Table 1. Preprocessing Times (mm:ss)

	1XMM	SUMSS	Tycho2	2MASS	USNO A2	USNO B1
1XMM	0:00	0:01	0:04	12:00	13:00	27:00
SUMSS	0:00	0:01	0:03	9:00	11:00	20:00
Tycho2	0:03	0:03	0:08	14:00	15:00	33:00
2MASS	57:00	32:00	34:00	57:00	60:00	112:00
USNO A2	50:00	34:00	19:00	56:00	44:00	116:00
USNO B1	116:00	63:00	103:00	166:00	138:00	226:00

Table 2. Matching Times (mm:ss)

5. Related Work

Reported algorithms for catalogue crossmatching have been based either on exhaustive enumeration or spatial indexing. Malik et al. (2003) report an exhaus-

tive enumeration approach as the basis of the XMATCH operation in SkyQuery. The approach is clearly O(NM) in processor costs, and O(N+M) in I/O only if the catalogues can be held in main memory. Gray et al. (2004) report an exhaustive enumeration approach with zoning for the closely related problem of neighbour finding. Index-based joins have been reported by Kunzst, Szalay & Thakar (2001), Kalpakis et al. (2001), and Page (2003). Preprocessing costs depend on the index to be built, but are in general quite high. Matching costs are reasonably low.

6. Conclusions

We have reported an algorithm for catalogue crossmatching based on the application of a filter-refine strategy and a plane sweep algorithm. Our algorithm is logarithic in both I/O and processor costs, and has low main memory requirements. Empirical performance assessment shows solution times of around six hours elapsed for matching of a billion record catalogue with a half billion record catalogue. This appears to be a significant improvement on current algorithms. Tentative comparisons with the approaches of Gray et al. (2004) and of Page (2003) indicate that for catalogues of moderate size our algorithm is faster by at least an order of magnitude. Consideration of the orders of complexity suggest that the advantage will be higher still for large catalogues.

References

Abel, D. J., Devereux, D., Power, R. A. & Lamb, P. R. 2004, An O($N \log M$) Algorithm for Catalogue Matching, CSIRO ICT Centre Technical Report TR-04/1846

Devereux, D., Abel, D. J. & Power, R. A. 2004, Notes on the Implementation of Catalogue Cross Matching, CSIRO ICT Centre Technical Report TR-04/1847

Gray, J., Szalay, A. S., Thakar, A. R., Fekete, G. F., O'Mullane, W., Heber, G., Rots, A. H. 2004, There Goes the Neighborhood: Relational Algebra for Spatial Data Search, Microsoft Research Technical Report 2004-32

Kunszt, P. Z., Szalay, A. S., Thakar, A. R. 2001, in Mining the Sky: Proc. of the MPA/ESO/MPE workshop, 631

Kalpakis, K., Riggs, M., Pasad, M., Puttagunta, V., & Behnke, J. 2001, in ASP Conf. Ser., Vol. 238, ADASS X, ed. F. R. Harnden, Jr., F. A. Primini, & H. E. Payne (San Francisco: ASP), 133

Malik, T., Szalay, A. S., Budavari, T. & Thakar, A. R. 2003, SkyQuery: A Web Service Approach to Federate Databases. In Proc. CIDR'03

Page, C. G. 2003, in ASP Conf. Ser., Vol. 295, ADASS XII, ed. H. E. Payne, R. I. Jedrzejewski, & R. N. Hook (San Francisco: ASP), 39

Power, R. A. & Devereux, D. 2004, Benchmarking Catalogue Cross Matching, CSIRO ICT Centre Technical Report TR-04/1848