

Notes on the Implementation of Catalogue Cross Matching

CSIRO ICT Centre Technical Report TR-04/1847

Drew Devereux, David J. Abel and Robert A. Power
{drew.devereux, dave.abel, robert.power}@csiro.au

CSIRO ICT Centre
GPO Box 664 Canberra, ACT 2601 Australia

Abstract

One of the basic problems in the integration of astronomical catalogues is that of how to determine which source records from different catalogues actually refer to the same source, usually on the basis of spatial co-location. This *catalogue matching* problem is inherently costly to solve. In [1], we present an $O(N \log M)$ algorithm based on filter-refine and plane sweep techniques. This technical report documents the design and implementation details of that algorithm.

1 Introduction

In [1], we present an $O(N \log M)$ algorithm for cross-matching of astronomical source catalogues based on spatial location. This technical report documents the design and implementation details of that algorithm. For documentation of our benchmarking conditions and results, see [3].

2 Problem Definition

The catalogue matching problem is the problem of finding all pairs of source records from different catalogues that in fact refer to the same source. There are a number of ways to tackle this problem, but the most obvious, and most discriminative, is to identify pairs of records whose sources are spatially co-located. We will focus on this spatial catalogue matching problem.

There are many variants of the spatial catalogue matching problem. We require a general version: one that can be applied in all cases, and that can be easily tailored to specific situations and catalogues. The problem we have chosen to address is as follows.

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 in each catalogue has a unique object identifier, and a spatial location as described below. We assume that the catalogues have previously been standardised in their coordinate systems and epoch, so that the spatial locations are commensurable. Errors may vary between records in a catalogue.

We shall take the apparent spatial location of a source to be a normally distributed random variable, expressed as a mean location in right ascension and declination coordinates, and a standard deviation value provided as a single angular distance. Note that this assumption is valid for many but not all catalogues.

A pair (a, t) of source records will be declared a match if we cannot be sufficiently confident that the true locations of a and t are not the same. Essentially, this means that we accept a pair of records as matching if the angular distance between their mean locations is smaller than some threshold value defined by their distributions. The task of our algorithm is to identify the set of matching pairs of source records. In addition, we will form the sets of source records from each catalogue for which no match is found.

Formally, we denote the domain of object ids as **OID**, the domains of right ascension and declination as **RA** and **DEC** respectively, and the domain of angular errors as Σ . Then a source s is defined as

$$s = \{\text{oid}_s, \text{ra}_s, \text{dec}_s, \sigma_s\} \in \text{OID} \times \text{RA} \times \text{DEC} \times \Sigma$$

and a catalogue S of sources is just

$$S \subseteq \text{OID} \times \text{RA} \times \text{DEC} \times \Sigma$$

The catalogue matching problem then is to identify, given two catalogues A and T , the set $M \subseteq A \times T$ of matching sources, the set $\hat{A} \subseteq A$ of unmatched sources from A , and the set $\hat{T} \subseteq T$ of unmatched sources from T . That is:

$$\begin{aligned} M &: \{(a, t) : a \in A, t \in T, a \text{ matches } t\} \\ \hat{A} &: \{a \in A : \forall t \in T, (a, t) \notin M\} \\ \hat{T} &: \{t \in T : \forall a \in A, (a, t) \notin M\} \end{aligned}$$

2.1 Definition of a Match

We have said that a pair (a, t) of sources will be declared a match if and only if the angular distance between them is sufficiently small that we cannot rule out the possibility that they are spatially co-located. Let us formalize this condition. The angular distance between two points a and t on a sphere is

$$\cos^{-1} \left(\sin(\text{dec}_a) \sin(\text{dec}_t) + \cos(\text{dec}_a) \cos(\text{dec}_t) \cos(\text{ra}_a - \text{ra}_t) \right)$$

An equivalent formula that is both less costly to compute and less subject to rounding errors for short distances and small angles is the Haversine formula:

$$2 \sin^{-1} \sqrt{\sin^2 \left(\frac{\text{dec}_a - \text{dec}_t}{2} \right) + \cos(\text{dec}_a) \cos(\text{dec}_t) \sin^2 \left(\frac{\text{ra}_a - \text{ra}_t}{2} \right)} \quad (1)$$

We must test this distance against a threshold value determined by the distributions of a and t . This threshold is defined in the following way: using standard statistical notation, we define α so that our desired confidence is $(1 - \alpha)100\%$. That is, if we choose not to reject a candidate pair of records unless we are 95% sure that they do not match, then our α value will be $\alpha = 0.05$. We then denote z_α as the value that puts a probability of α in each tail of a normal distribution. That is, if $z \sim N(0, 1)$, then $P(z > z_\alpha) = \alpha$.

The questions whether two sources a and t are spatially co-located can be rephrased as the question whether the distance between their true locations is zero. Accordingly, we can express our test as a hypothesis about the distribution of the random variable $(a - t)$. If a and t are normally distributed with $a \sim N(\mu_a, \sigma_a^2)$ and $t \sim N(\mu_t, \sigma_t^2)$ respectively, then $(a - t)$ is also normally distributed, with $(a - t) \sim N(\mu_a - \mu_t, \sigma_a^2 + \sigma_t^2)$. In fact, even if a and t are not normally distributed, the central limit theorem indicates the $(a - t)$ will be approximately normally distributed. We are therefore not too heavily reliant on the assumption of normality of the distributions of a and t .

Our confidence in the hypothesis that a and t are not spatially co-located can be calculated as

$$\left| \frac{\mu_a - \mu_t}{\sqrt{\sigma_a^2 + \sigma_t^2}} \right|$$

which leads us to reject a candidate pair whenever

$$\left| \frac{\mu_a - \mu_t}{\sqrt{\sigma_a^2 + \sigma_t^2}} \right| > z_{\alpha/2}$$

This can be expressed as

$$|\mu_a - \mu_t| > z_{\alpha/2} \sqrt{\sigma_a^2 + \sigma_t^2}$$

The left hand side of this equation is just the distance between the two mean positions; thus this inequality states that our angular threshold for rejection of the pair (a, t) s should be

$$z_{\alpha/2} \sqrt{\sigma_a^2 + \sigma_t^2} \quad (2)$$

Bringing together (1) and (2), we say that a pair (a, t) of sources match if and only if

$$2 \sin^{-1} \sqrt{\sin^2\left(\frac{\text{dec}_a - \text{dec}_t}{2}\right) + \cos(\text{dec}_a) \cos(\text{dec}_t) \sin^2\left(\frac{\text{ra}_a - \text{ra}_t}{2}\right)} \leq z_{\alpha/2} \sqrt{\sigma_a^2 + \sigma_b^2} \quad (3)$$

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 that occurs in multiple matches.

3 Implementation

A brute force algorithm that applied the full test to every possible pair of sources from a pair of catalogues would scale as $O(N_A N_T)$ for catalogues with sizes N_A and N_T , and so would be prohibitively expensive to run with large catalogues. Instead we have developed an $O(N_T \log N_A)$ algorithm [1]. The design of the algorithm applies two concepts. The first is the well known filter-refine strategy of geospatial databases. The second is to apply a variation of the plane sweep algorithm to identify pairs of points that are near neighbours.

3.1 Filter-refine strategies

Filter-refine strategies [2] are applied in two steps. In the filter step, a weakened form of the problem is solved to rapidly generate a set of candidates. The candidates are guaranteed to include all pairs of records that satisfy the criteria of the full form of the problem, but may also include some *false hits* that do not satisfy the full criteria. Thus, the filter step rapidly discards most but not all of the non-matching pairs, so that many fewer pairs need to be tested further. In the refinement step, the remaining candidates are tested as satisfying the full constraints of the problem. Choice of the weakened form must strike a balance between the cost of generating a set of candidates versus the costs of diagnosing the false hits.

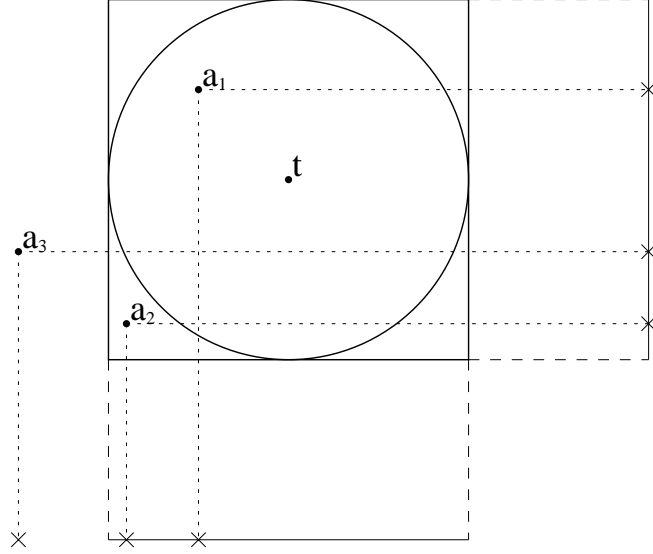


Figure 1: A Bounding Box Filter

It is much cheaper to test a point for containment in a circle's bounding box than in the circle itself. Here, a_1 is accepted; a_3 is rejected; a_2 is a “false hit” – it is accepted by the filter but will fail the full test.

One common approach to defining a weakened form of problems similar to ours is the use of bounding boxes. For example, a complex test for intersection of two complex shapes may be filtered by a simple test for intersection of their bounding boxes. In our case, we can avoid computation of the angular distance by rephrasing our test into a test whether one of our sources is contained inside a circle whose centre is the other point and whose radius is our threshold. The computationally expensive test of angular distance is then replaced by two computationally trivial tests: a test whether the nominal right ascension of our source falls within the right ascension range of our bounding box; and an identical test in the declination dimension. The tradeoff is that the circle takes up only $100(\frac{\pi}{4})\%$ of the box's area, so that on average $100(1 - \frac{\pi}{4})\%$ of the reported matches will be false hits.

3.2 Constructing the bounding box

The bounding box of a spherical circle s with its centre at (ra_s, dec_s) and radius θ_s is a 4-tuple

$$\{ra_low_s, ra_high_s, dec_low_s, dec_high_s\}$$

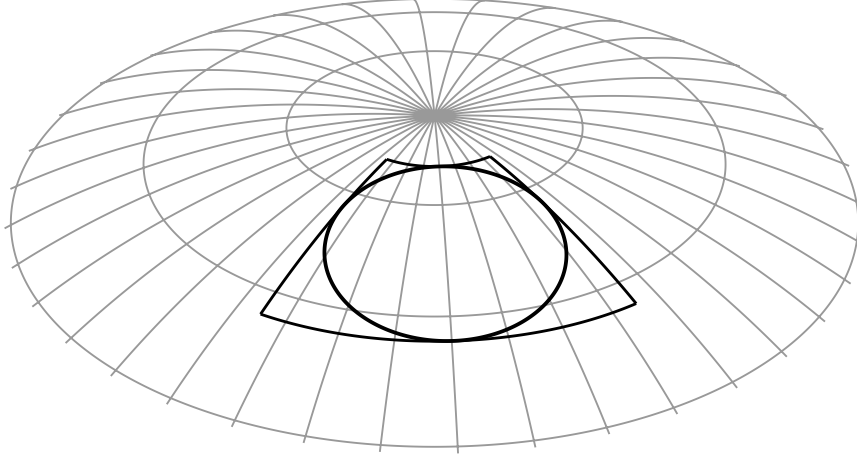


Figure 2: A bounding box in right ascension and declination will appear trapezoidal near a pole.

where ra_low_s and ra_high_s are respectively the lower and upper bounds of the circle in the right ascension dimension, and dec_low_s and dec_high_s are respectively the lower and upper bounds in declination. Note that when a bounding box crosses the prime meridian (the line $\text{ra} = 0$) the value of ra_low_s will be greater than the value of ra_high_s .

It is important to recognise that the bounding box cannot be assumed to have a rectangle appearance. Near the equator, bounding boxes will appear rectangular, but nearer the poles they will appear increasingly trapezoidal. For error circles that overlap a pole, the bounding box will be a circle centred on the pole. However, we are unconcerned with the geometrical interpretation of our bounding box; what is important is the following properties:

- bounding boxes may be computed at low cost;
- points are easily checked for containment in a bounding box; and
- whenever a circle contains a point, its bounding box necessarily contains the point.

This last point ensures that no valid candidate will be rejected by our filter.

Values for dec_low_s and dec_high_s are easily derived. For dec_low_s we simply subtract the circle's radius from its centre declination; that is, $\text{dec_low}_s = \text{dec}_s - \theta_s$. For dec_high_s we add the radius instead. A minor adjustment must be made for error circles that contain a pole. When an error circle contains the point $\text{dec} = 90^\circ$, the computed dec_high_s value

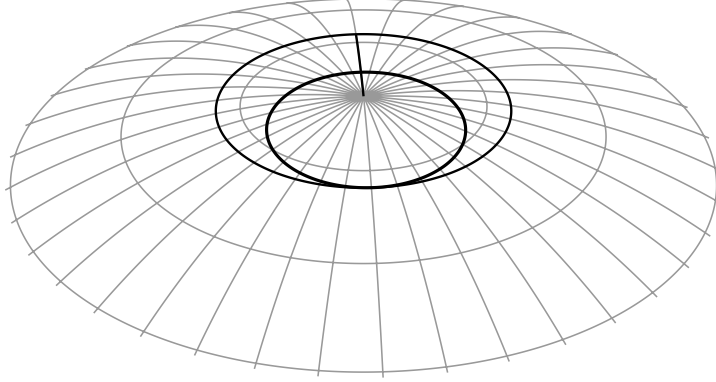


Figure 3: When a bounding box overlaps a pole, it defines a circular area.

will exceed 90° , and must be set to 90° . Similarly, when an error circle contains the point $\text{dec} = -90^\circ$, the dec_high_s value must be set to -90 . In both these special cases the error circle necessarily spans the entire range of right ascensions, so the ra_low_s and ra_high_s values are set to 0° and 360° respectively.

Values for ra_low_s and ra_high_s are not as trivial to compute. The distance of a point from a line of constant right ascension must be measured along the shortest path, which is a great circle. Except at the equator, lines of constant declination are not great circles, and so the points on a spherical circle of lowest and highest right ascension generally will not have the same declinations as the source's nominal location. Consequently, simply adding or subtracting the radius will not yield a correct boundary value (see Figure 4). The correct approach for computing ra_low_s and ra_high_s values is to first compute the correction

$$\Delta\text{ra}_s = \sin^{-1} \frac{\sin \theta_s}{\cos \text{dec}_s} \quad (4)$$

To obtain an ra_low_s value, this Δra_s , instead of θ_s , is subtracted from the source's nominal right ascension. Similarly, an ra_high_s value is obtained by adding Δra_s to the source's nominal right ascension. Naturally these calculations must be modulo 360.

3.3 The containment test

As stated earlier, testing the nominal location of source a for containment in the bounding box of source t is extremely computationally efficient in comparison to containment in a circle. In the declination dimension, we

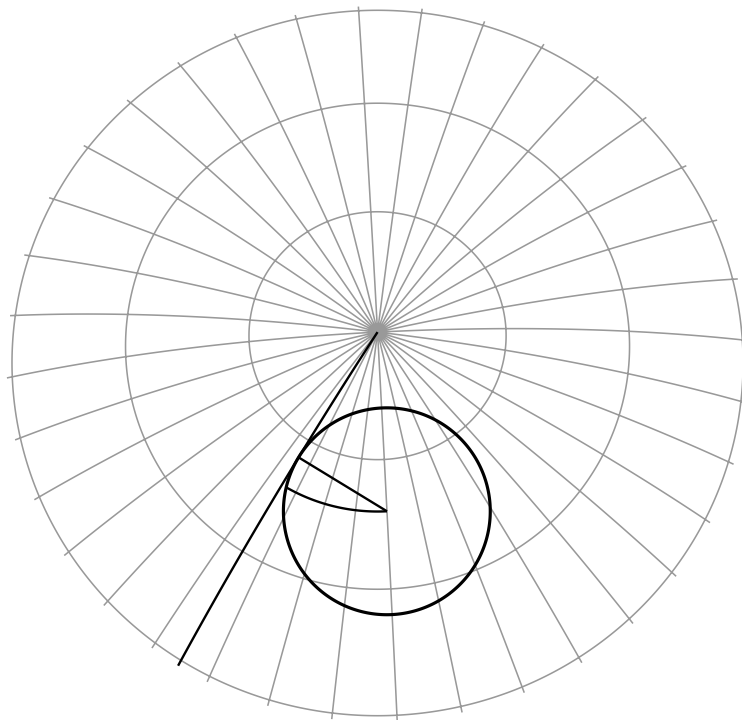


Figure 4: A line of constant right ascension is generally not the shortest path between two points, and so should not be used to measure distance. Here we see that a line of constant right ascension cannot be used to find the right ascension bounds of a spherical circle.

simply require the declination of the point to fall within the declination range of the bounding box. That is, we require that $\text{dec_low}_t \leq \text{dec}_a$ and $\text{dec}_a \leq \text{dec_high}_t$. Similar conditions hold for the right ascension dimension, but some care must be taken to account for bounding boxes that straddle the prime meridian. For bounding boxes that straddle the prime meridian, we require $\text{ra_high}_t \leq \text{ra}_a$ and $\text{ra}_a \leq \text{ra_low}_t$; for all the other bounding boxes, we require $\text{ra}_a \geq \text{ra_low}_t$ and $\text{ra}_a \leq \text{ra_high}_t$. These conditions can be conveniently combined into the single condition

$$(\text{ra_low}_t \leq \text{ra}_a) = (\text{ra}_a \leq \text{ra_high}_t) = (\text{ra_low}_t \leq \text{ra_high}_t)$$

3.4 Defining our filter-refine strategy

We can now formally define our filter refine strategy. Given two sources $a \in A$ and $t \in T$, our filter will accept the pair (a, t) as a candidate for further testing if and only if a is contained in the bounding box of a spherical circle whose centre is $(\text{ra}_t, \text{dec}_t)$ and whose radius is $z_{\alpha/2} \sqrt{\sigma_a^2 + \sigma_t^2}$. That is, we require the following three conditions to hold:

$$\begin{aligned} \text{dec_low}_t &\leq \text{dec}_a \\ \text{dec}_a &\leq \text{dec_high}_t \\ (\text{ra_low}_t \leq \text{ra}_a) &= (\text{ra}_a \leq \text{ra_high}_t) = (\text{ra_low}_t \leq \text{ra_high}_t) \end{aligned}$$

where

$$\text{dec_low}_t = \text{dec}_t - z_{\alpha/2} \sqrt{\sigma_a^2 + \sigma_t^2} \quad (5)$$

$$\text{dec_high}_t = \text{dec}_t + z_{\alpha/2} \sqrt{\sigma_a^2 + \sigma_t^2} \quad (6)$$

$$\text{ra_low}_t = \text{ra}_t - \sin^{-1} \left(\frac{\sin(z_{\alpha/2} \sqrt{\sigma_a^2 + \sigma_t^2})}{\cos(\text{dec}_t)} \right) \pmod{360} \quad (7)$$

$$\text{ra_high}_t = \text{ra}_t + \sin^{-1} \left(\frac{\sin(z_{\alpha/2} \sqrt{\sigma_a^2 + \sigma_t^2})}{\cos(\text{dec}_t)} \right) \pmod{360} \quad (8)$$

Note that many of these definitions share subterms, so checking the filter conditions is much cheaper than performance of the full angular separation test. Yet any pair (a, t) of points rejected by the filter would be rejected by the full test. Use of the filter will therefore cause the rejection of many false matches at low cost. The remaining candidates are then refined by

application of the full test. In applying the full test, we note that (3) can be rearranged to the more efficiently computed test

$$\sin^2\left(\frac{\text{dec}_a - \text{dec}_t}{2}\right) + \cos(\text{dec}_a)\cos(\text{dec}_t)\sin^2\left(\frac{\text{ra}_a - \text{ra}_t}{2}\right) \leq \sin^2\left(\frac{z_{\alpha/2}\sqrt{\sigma_a^2 + \sigma_b^2}}{2}\right) \quad (9)$$

3.5 The plane sweep approach

Although the use of a filter-refine strategy can substantially reduce the per-test cost of identifying matches, any algorithm that tests every possible source pair must scale as $O(N_A N_T)$. To avoid this, we adopt as our filter step an algorithm that ensures we only test sources against their near neighbours. Our algorithm is based on the plane sweep approach [4].

Plane sweep algorithms were originally introduced for solving a family of computational geometry problems, one of which is closely related to our bounding box containment problem. In the plane sweep approach, a line is swept through the space, generating events whenever it encounters an entity of interest (such as the side of a bounding box). These events are then handled in an appropriate manner. One of the standard event handling metaphors is the maintenance of an *active list*, which keeps track of a subset of entities such as the set of bounding boxes currently intersected by the sweep line.

To illustrate, consider the classical computational geometry problem of identifying, given a set of points and a set of rectangles, the set of all point-rectangle containments. This is somewhat similar to our problem, but it differs in some important ways. The standard plane sweep approach is to sweep a line through the space, and hence through each rectangle. Whenever the line enters a rectangle, that rectangle is added to the active list; whenever the line exits a rectangle, the rectangle is removed from the active list. Thus at all times the active list contains all rectangles currently intersected by the sweep line. In addition, whenever the line encounters a point, it is tested for containment against each of the rectangles currently in the active list. For most problem instances, the number of rectangles in the active list is very much smaller than the total number of rectangles, so only a very small proportion of the possible containment tests are ever performed. Moreover, this event handling strategy ensures that a point is only ever tested for containment in a rectangle if containment holds in the dimension along which the sweep line is travelling. Accordingly, the containment test need only check for containment in the other dimension. Careful construction of the active list data structure can ensure that this test is efficient.

3.6 A plane sweep algorithm for catalogue matching

The example above is a good illustration of the plane sweep concept, but it cannot be applied to our problem in its current form for a number of reasons. Firstly, there is the obvious and important fact that we are sweeping a spherical surface rather than a plane. Secondly, our bounding boxes cannot be treated as rectangles as in the example above, because their size varies with the attributes of the point being tested. Finally, we must address the high I/O costs associated with the example version. In this section, we construct a plane sweep algorithm for our catalogue matching problem.

3.6.1 Orientation of the sweep line

Since we will be sweeping a line through our space, the first design decision we must make is the orientation of the sweep line; that is, whether to sweep our line along the right ascension axis or along the declination axis. In the former case, the sweep line is a line of constant longitude from the south pole to the north pole, which sweeps from the prime meridian (right ascension = 0°) (or some other arbitrary right ascension) around the sphere until it reaches its starting point. In the latter case, the sweep line is a circle of constant latitude that starts as a point at the south pole, and grows as it moves up the sphere until it reaches maximum size at the equator (declination = 0°), then shrinks as it moves further up the sphere until it terminates at the north pole.

The more obvious approach is to sweep by right ascension, and this has the advantage that the sweep line remains the same length at all times, which means that the active list will be about the same size at all times. When sweeping by declination, the sweep line varies in size, and hence so does the active list. There is a slight loss of efficiency associated with this size variation.

However, there are a number of problems with sweeping by right ascension that outweigh this minor disadvantage. Of minor importance is the fact that our bounding boxes may straddle the prime meridian (or any other starting point for a sweep by right ascension). Thus the sweep line will initially intersect these boxes despite not having encountered an intersection event. It is not difficult to handle these special cases, and the additional processing costs are negligible, but it does lack elegance.

A more important problem is caused by the fact that bounding boxes very near the pole may have very large right ascension ranges; in the extreme case where an error circle contains a pole, the bounding box will range over the full 360° of right ascension (see Figures 2 and 3). If we sweep by right ascension, these rectangles will spend a long time on the active list and be

subject to many comparisons; if we sweep by declination, the time they spend on the active list remains small, and so many fewer comparisons are made. Moreover, in a sweep by declination these problem bounding boxes are handled while the sweep line is extremely small, which tends to cancel out the difficulties caused by their large size in right ascension. The conclusion is that sweeping by declination is substantially more efficient than sweeping by right ascension. This has been empirically confirmed.

3.6.2 Defining the events

Recall that our bounding boxes, as currently defined, vary in size with the attributes of the point against which they are being tested. This makes it impossible to construct a well-defined set of events for our bounding boxes. We will need to slightly alter the definition of a bounding box to make our problem amenable to a plane sweep approach. This redefinition must take into account an additional issue:

The sweeping of a line through the space is a geometrical metaphor that really implies little more than the accessing of events in a well-defined order. It follows that our plane sweep algorithm will require us to sort our events before we begin the plane sweep itself. This requirement places some substantial constraints on the nature of the events that we handle in our sweep. Specifically, note that if we treated the entering and exiting of a bounding box as events, then we would have to sort on these attributes. This would require reading our data, constructing `dec_low` and `dec_high` values, and then sorting them into ascending order. A preliminary investigation of this option indicated that this approach has unacceptably high I/O costs.

Both of the above issues can be resolved by defining our bounding box and events so that we can sort each of our catalogues on their nominal declination value. Our approach is as follows: Firstly, we sort both catalogues on their nominal declination values, and while we are doing so, we note their maximum standard deviations.

An event is triggered whenever the sweep line encounters a source (that is, the nominal location of a source) t from catalogue T . On encountering such a source, we add to the active list all sources from catalogue A that conceivably could, on the basis of declination, match t . That is, we compute a “worst case” `dec_hight` value for t ’s bounding box, and add to the active list all sources from A whose nominal location is less in declination than this worst case value. To compute this worst case `dec_hight`, we use (6), but in place of a specific σ_a value, we use the maximum standard deviation of the catalogue A . If $\max\sigma_A$ is the maximum standard deviation of the A , then

our `dec_hight` value is

$$\text{dec_high}_t = \text{dec}_t + z_{\alpha/2} \sqrt{\text{max}\sigma_A^2 + \sigma_t^2} \quad (10)$$

and we add to the active list all remaining points $a \in A$ such as $\text{dec}_a \leq \text{dec_high}_t$.

The next event handling step is to remove from the active list those sources that can no longer match t . By the same logic as above, we should remove all sources up to a declination of `dec_lowt`, where

$$\text{dec_low}_t = \text{dec}_t - z_{\alpha/2} \sqrt{\text{max}\sigma_A^2 + \sigma_t^2}$$

We must be very careful, however, when it comes to deleting sources from the active list, because it is possible for a not yet encountered source from T with a larger standard deviation than t to match a source from the active catalogue that t cannot (see Figure 5). Once we have deleted a source from the active list, we cannot add it again, so we must be conservative in our deletion policy. To account for this possibility, we use the maximum standard deviation $\text{max}\sigma_T$ of T in place of σ_t ; that is, we remove from the active list all sources whose declination is less than `dec_lowt`, where

$$\text{dec_low}_t = \text{dec}_t - z_{\alpha/2} \sqrt{\text{max}\sigma_A^2 + \text{max}\sigma_T^2} \quad (11)$$

The result of this event handling is that the active list is maintained so as to contain every source from A whose nominal location is contained within the declination range of the error circle centred on t . The active list may, however, contain a small proportion of additional sources from A that do not quite meet this criterion, because the bounding box that we use is slightly larger than the strictly minimum bounding box of the error circle.

The only remaining item of event handling is the containment test itself. Once the active list has been updated to contain all sources from the active catalogue that match the current test source in declination, we must query the active list for its sources that also match in right ascension.

3.6.3 Querying the active list

To individually test each source on the active list for intersection with the test point would be unnecessarily costly. Instead, we maintain the active list in a data structure that allows us to efficiently obtain all sources a whose ra_a is between `ra_lowt` and `ra_hight` (taking into account prime meridian effects). As before, `ra_lowt` and `ra_hight` are not defined independently of

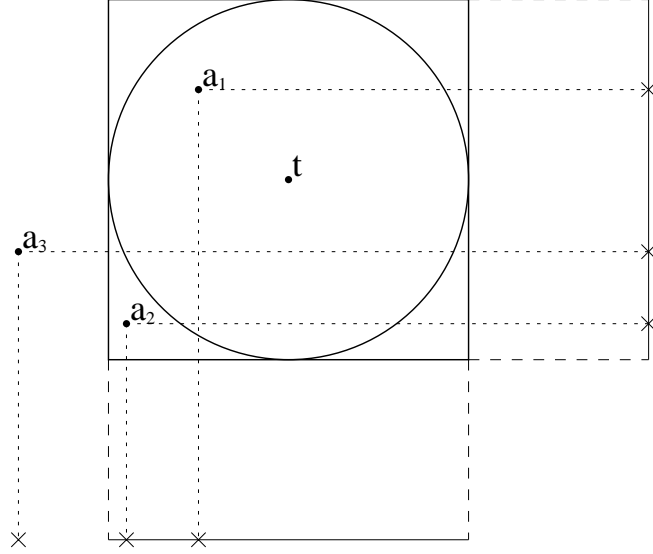


Figure 5: Care must be taken to delete points from the active list only when the current and all subsequent test points can no longer match them. Here, point a cannot match point t_1 , but it would be a mistake to delete it, since the not yet encountered point t_2 matches a .

the particular active point between tested, so we use worst case values in which σ_a is replaced by $\max\sigma_A$. That is,

$$\text{ra_low}_t = \text{ra}_t - \sin^{-1} \left(\frac{\sin(z_{\alpha/2} \sqrt{\max\sigma_A^2 + \sigma_t^2})}{\cos(\text{dec}_t)} \right) \pmod{360} \quad (12)$$

$$\text{ra_low}_t = \text{ra}_t + \sin^{-1} \left(\frac{\sin(z_{\alpha/2} \sqrt{\max\sigma_A^2 + \sigma_t^2})}{\cos(\text{dec}_t)} \right) \pmod{360} \quad (13)$$

3.6.4 A data structure for the active list

The data structure underlying the active list must provide efficient operations for insertion of sources; deletion of all sources whose declination is less than a given value; and determination of all sources within a given range of right ascensions. To achieve this, the data structure maintains its sources in both ascending order of declination and ascending order of right ascension.

Since sources are inserted into the active list in ascending order of declination, inserted sources may be simply appended to the end of the declination ordering. Similarly, deletion of all sources whose declination is less than a

given value is achieved by repeatedly deleting sources from the start of the declination ordering until the first source is greater than or equal to the given value. Therefore the data structure maintaining the declination ordering is a queue: it provides efficient operations for appending to the end and deleting from the front, but not for random access to internal sources. We have implemented this as a doubly-linked list with head and tail pointers.

The requirements for storage in right ascension ordering are somewhat different. Insertion and deletion of sources is unordered on right ascension, and so may occur at any location in the ordering. Identification of all sources in a given range implies location of range boundaries within the ordering, so that all sources between the range boundaries may be reported. Since we require random access to any source in the right ascension ordering, we implement this ordering with a binary tree. The necessity of efficiently visiting every source between two range boundaries implies that the binary tree must be threaded. Since insertion and deletion of sources is unordered on right ascension, the binary tree will, on average, remain balanced, so there is no need for a data structure that actively maintains the tree in a balanced state. A simple threaded binary tree suffices.

3.6.5 The algorithm

For convenience, we repeat here the algorithm presented in [1].

Given two catalogues A and T , and assuming that A will be passed through the active list and T will provide test sources, the steps of the algorithm are:

1. Preprocess the catalogues A and T by sorting them into ascending order of declination; during preprocessing, keep track of the maximum errors in each.
2. For each source t in T
 - Delete from the active list all members that cannot match t or any subsequent source from T , because their declinations are less than dec_low_t as defined in (11); report as unmatched any deleted member that has not been matched to a source from T .
 - Add to the active list all sources from A that might match t , because their declinations are less than or equal to dec_high_t as defined in (10);
 - Report all pairs (a, t) where a is a member of the active list whose right ascension falls between

`ra_lowt` and `ra_hight`, as defined in (12) and (13) respectively, and taking into account prime meridian effects; if nothing is reported, then report `t unmatched`.

3. Terminate by flushing any remaining members of the active list, reporting as unmatched any deleted member that has not been matched to a source from T .

3.6.6 Refinements for limited-range catalogues

Many of the catalogues that are available or likely to become available in the VO are not all-sky surveys. Some have quite limited ranges. If we review our algorithm from the point of view of this use case, we see a number of inefficiencies that may be eliminated by some minor refinements.

Firstly, in declination intervals covered only by the active catalogue, many sources from the active catalogue will be added to the active list, only to be deleted again without ever being the object of a containment test. This adds substantially to the costs of the algorithm, but it is easily addressed by a minor change: whenever the next source from the test catalogue is too far away to be capable of matching the next source from the active catalogue, the active list is flushed and sources from the active catalogue are discarded until a source is encountered that is not too far away to match the next test point. We then add this source to the active list and continue in the usual way. A special case is when the test catalogue is exhausted: we flush the active list and discard all remaining sources in the active catalogue.

A similar situation occurs in declination intervals covered only by the test catalogue. Here, sources may be repeatedly tested against an empty active list, causing poor performance. Instead, we discard test sources until we encounter one that causes the next source from the active catalogue to be added to the active list, and then proceed as usual.

Preliminary empirical results indicate that the first of these refinements results in a 20% speedup for our example limited-range catalogues, with the second yielding a further 20% speedup. The result overall is an 36% speedup in these special cases. Importantly, these changes have no effect on the performance of the algorithm on all-sky catalogues.

4 Results

In I/O costs, the algorithm is trivially $O(N_A \log N_A + N_T \log N_T)$ for the preprocessing stage and $O(N_A + N_T)$ for the matching itself. For processor

costs for the matching, the algorithm is $O(N_T \log N_A)$ through the access and maintenance costs of the binary tree. For full empirical results, see [3].

The expected length of the active list is $O(N_A)$ and varies with the cosine of the declination of the current object from B. Assuming a uniform distribution of sources about the celestial sphere, the active list will be largest when the sweep line is largest; that is, at the equator. This can be estimated by the number of sources from the active catalogue whose declinations lie in a strip of width $2\sqrt{\max\sigma_A^2 + \max\sigma_T^2}$. The area of this strip may be calculated using surface-of-revolution integration techniques, yielding an area of $4\pi \sin(\sqrt{\max\sigma_A^2 + \max\sigma_T^2})$. As a proportion of the total surface area of the sphere, this area is:

$$\sin(\sqrt{\max\sigma_A^2 + \max\sigma_T^2})$$

and the expected number of objects from A whose nominal locations lie in this strip is then:

$$N_A \sin(\sqrt{\max\sigma_A^2 + \max\sigma_T^2})$$

For a catalogue of 10^9 objects and with representative values of 1 arcminute for both $\max\sigma_A$ and $\max\sigma_T$, the active list would have around 580000 elements. This confirms that the active list can be maintained as a main-memory data structure for the catalogues of sizes currently available or envisaged.

5 Conclusions

In [1], we reported an algorithm for catalogue matching that is logarithmic in both I/O and processor costs and low main-memory requirements, achieved by applying filter-and-refine and plane sweep concepts from geospatial database and computational geometry. In this technical report, we have documented the design decisions and implementation details of this algorithm.

References

- [1] David J. Abel, Drew Devereux, Robert A. Power, and Peter R. Lamb. An $O(N \log M)$ algorithm for catalogue matching. Technical Report TR-04/1846, CSIRO ICT Centre, Canberra, Australia, 2004.
- [2] R. H. Güting. An introduction to spatial database systems. *VLDB Journal*, 3:357–399, 1994.

- [3] Robert A. Power and Drew Devereux. Benchmarking catalogue cross matching. Technical Report TR-04/1848, CSIRO ICT Centre, Canberra, Australia, 2004.
- [4] F. Preparata and M. Shamos. *Computational geometry: an introduction*. Springer Verlag, New York, 1985.