

**An  $O(N \log M)$  Algorithm for Catalogue Matching**  
CSIRO ICT Centre Technical Report TR-04/1846

**David J. Abel, Drew Devereux, Robert. A. Power and Peter R. Lamb**  
CSIRO ICT Centre  
GPO Box 664  
Canberra, ACT 2601  
Australia  
**{dave.abel, drew.devereux, robert.power, peter.lamb}@csiro.au**

**ABSTRACT**

The catalogue matching problem is a ubiquitous problem in data fusion in Virtual Observatories. It deals with reporting the locationally coincident pairs of objects in two astronomical databases where locations are imprecise. Our algorithm applies the filter-and-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 and has low-main memory requirements. Empirical assessment on matching problems of up to 1.5 billion objects suggests the algorithm has performance at least an order of magnitude better than the techniques in current use.

## **1 INTRODUCTION**

One of the central motivations for Virtual Observatories is that integrating data from two or more catalogues will provide richer or more extensive data sets that can be explored using data mining techniques such as classification to detect new types of bodies or to find more instances of rare or unusual types of bodies. The fundamental data fusion operation is to identify the objects appearing in both catalogues on the basis of equivalence of their spatial references. Catalogue matching can be required either to the merge of a number of catalogues to form a new catalogue to be used for many purposes or as a join within a search spanning a number of catalogues [1].

The operation is inherently costly, because the catalogues of interest are very large and the matching operation is intrinsically a join. The SuperCOSMOS catalogue of over a billion bodies is an early representative of the large catalogues now being generated by sky surveys [2]. Extrapolation of the solution times reported for smaller data sets indicates that the current algorithms with costs that are quadratic with problem size would have solution times of days for merging catalogues of this size. There is clearly merit in investigating faster algorithms.

In this paper, we report an algorithm to determine the pairs of objects, drawn from two catalogues, that are spatially equivalent. Often the set of pairs would be subjected to further tests of consistency in red shift, their emissions profile and so on. We assume that spatial indexes are not available. Our algorithm has pre-processing 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  $O(N\log M)$ .

We begin by formally describing the problem and then present our algorithm as based on the filter-and-refine strategy and on plane-sweep concepts. An empirical assessment of the algorithm shows that, with modest computing equipment, matching of catalogues of a billion objects can be performed in under 6 hours. We relate our algorithm to previous work in astronomy and other areas, and conclude with an analysis of the scope of applicability of our algorithm.

## 2 The Catalogue Matching Problem

There are many possible variants of the catalogue matching problem due to differences in the precise methodologies applied. We consider here only the spatial filter step of catalogue matching. This determines pairs of objects whose locations are essentially coincident. The object pairs from the spatial filter step would then be subjected to validation tests that check consistency of the other attributes of each pair of objects. To address the problem as generally as possible, and to establish some terminology, we begin with a formal statement of a core form of the problem.

We are given two catalogues  $B$  and  $C$  that have  $N$  and  $M$  objects (bodies) respectively. Each record (tuple) in  $B$  and  $C$  has a unique object identifier. The spatial location of an object is a normally distributed random variable described by a mean location in right ascension and declination coordinates, and a representation of variance. More fully, the location of an object is described by an ellipsoid of arbitrary orientation, which in turn can be represented by the orientation of its major axis and the lengths of its major and minor axes. The ellipsoid represents the area within which the object falls within a 95% confidence level. The lengths of the major and minor axes then are the standard deviations for the imprecision of the location within the local coordinate system of the ellipse. For simplicity of presentation, we will present our algorithm in terms of circles rather than ellipsoids. The modifications to deal with ellipsoids are straightforward, but lead to longer expressions. We will allow, however, the standard errors to vary between objects in a catalogue. That is, we have two catalogues  $B$  and  $C$  defined as

$$\begin{aligned} B &: \{b[\text{oid}, \text{ra}, \text{dec}, \sigma]\} \\ C &: \{c[\text{oid}, \text{ra}, \text{dec}, \sigma]\} \end{aligned}$$

Note the use of standard notation so that  $b[\text{ra}]$ , for example, refers to the attribute  $\text{ra}$  for an object instance from  $B$ .

Two objects will be accepted as spatially equivalent if the angular distance between them is less than a threshold expressed in terms of their standard errors. We can represent this test generally by a function that calculates the angular separation of a pair of objects, relates the separation to the standard errors for the two objects and returns a Boolean value if the separation is less than some specified threshold  $p$ . We denote this function by

$$t(b[ra], b[dec], b[\sigma], c[ra], c[dec], c[\sigma], p) \rightarrow \text{bool}$$

The catalogue matching problem then is to form the sets  $D$  (the pairs of objects from  $B$  and  $C$  that are accepted as matches),  $E$  (the objects in  $B$  that are not in  $D$ ) and  $F$  (the objects in  $C$  that are not in  $D$ ). That is,

$$D = \{d: (b,c) \mid t(b[ra], b[dec], b[\sigma], c[ra], c[dec], c[\sigma], p) \rightarrow \text{TRUE for all } b, c\}$$

$$E = \{b \mid t(b[ra], b[dec], b[\sigma], c[ra], c[dec], c[\sigma], p) \rightarrow \text{FALSE for all } b\}$$

$$F = \{c \mid t(c[ra], b[dec], b[\sigma], c[ra], c[dec], c[\sigma], p) \rightarrow \text{FALSE for all } c\}$$

We assume that the source catalogues are presented in no known sequence, that there are no pre-existing indexes on the catalogues, and that the catalogues are too large to be held in main memory. We also assume that the catalogues have previously been standardised in their coordinate systems and epoch.

### 3 ALGORITHM OUTLINE

The design of the algorithm applies two concepts. The first is to use the well-known filter-and-refine strategy of geospatial database. The second is to apply a variation of the plane sweep algorithm to identify pairs of points that are near-neighbours.

#### Filter and Refine

Filter-and-refine strategies [3] have two steps. In the filter step, a weakened form of the problem is solved to generate rapidly a set of candidates, typically aiming to minimise I/O costs. The candidates are guaranteed to include all pairs of objects (hits) satisfying the criteria of the full form of the problem but can also include some objects (false drops) not satisfying the full criteria. In the refinement step, the candidates are tested as satisfying the full constraints of the problem. Choice of the weakened form then strikes a balance between the costs of generation of a set of candidates and the costs of diagnosing the false drops.

The full form of the problem is to report all pairs of objects whose angular separation is less than a threshold expressed in terms of the standard errors of the two points and a required confidence level  $z$ :

$$\cos^{-1}(\sin(b[dec])\sin(c[dec]) + \cos(b[dec])\cos(c[dec])\cos(b[ra]-c[ra])) \leq z(b[\sigma]^2 + c[\sigma]^2)^{1/2}$$

(This expression for the angular separation in fact leads to high inaccuracies for low declinations, and the equivalent Haversine formula is preferable. For further details, see [6].)

This full form can be rephrased as the problem of reporting all pairs of objects such that one of the objects is contained within a circle of radius  $z(b[\sigma]^2 + c[\sigma]^2)^{1/2}$  centred on the other object. Our weakened form of the problem is to report all pairs of objects such that one of the objects is contained within the bounding box of the other object's circle. The bounding box is defined by the limits of the circle in right ascension and declination.

Taking into account the necessary correction to right ascension values for declination effects, the accepted candidate pairs are the pairs (b, c) such that

$$|b[\text{ra}] - c[\text{ra}]| \leq \sin^{-1}(\sin(z(b[\sigma]^2 + c[\sigma]^2)^{1/2}) / \cos(b[\text{dec}])) \quad (1)$$

and

$$|b[\text{dec}] - c[\text{dec}]| \leq z(b[\sigma]^2 + c[\sigma]^2)^{1/2} \quad (2)$$

## Evaluating the Candidates

The algorithm to identify candidate pairs follows the basic concepts of the plane sweep algorithm [4] by structuring processing around a regular set of events. The event in this case is a search for the objects from C that match a specific object b from B. The set of events is made regular by processing the objects from B in ascending sequence by declination. (We will note later the significance of sweeping by declination rather than right ascension.) To ensure that the search within each event is executed efficiently, we maintain an active list of the objects from C that might match the current object from B or further objects from B, within a maintenance policy that restricts the size of the active list to be available when dealing with any member of B. As we will see, this can be handled by a sequential traversal of C.

The algorithm, in outline, is:

- 1 Pre-process the catalogues B and C, by sorting them into ascending sequence by declination;
- 2 For each object from B,
  - a. Delete from the active list any members that cannot match the current member of B or any further members of B, on the basis of declination;
  - b. Insert into the active list any members of C that might match the current object from B on the basis of declination;
  - c. Search within the active list for matches against b;
- 3 Terminate, by flushing any remaining members of the active list.

Maintenance of the active list must ensure that it contains all members of C that satisfy the declination constraint on matching (2) for the current object from B. That is, the insertion discipline must ensure that it contains all members of C such that

$$b[\text{dec}] - z(b[\sigma]^2 + c[\sigma]^2)^{1/2} \leq c[\text{dec}] \leq b[\text{dec}] + z(b[\sigma]^2 + c[\sigma]^2)^{1/2}$$

The deletion discipline restricts the size of the active list by dropping any members that can no longer match a member of B. A naïve approach would be to delete any members for which

$$c[\text{dec}] < b[\text{dec}] - z(b[\sigma]^2 + c[\sigma]^2)^{1/2}$$

However, because the imprecisions are not uniform, there could be a member  $b'$  of B still to be encountered for which

$$b'[\text{dec}] - z(b'[\sigma]^2 + c[\sigma]^2)^{1/2} < b[\text{dec}] - z(b[\sigma]^2 + c[\sigma]^2)^{1/2}$$

We therefore modify our membership of the active list to be

$$b[\text{dec}] - z(\max(b[\sigma]^2 + \max(c[\sigma]^2))^{1/2} < c[\text{dec}] < b[\text{dec}] + z(b[\sigma]^2 + \max(c[\sigma]^2))^{1/2}$$

The search of the active list in Step 2c is a range search on right ascension, to find those members of the active list that also satisfy Constraint (1). We require the range to cover all members of the active list such that

$$b[\text{ra}] - z(\sigma_b'^2 + \sigma_c'^2)^{1/2} < c[\text{ra}] < b[\text{ra}] + z(\sigma_b'^2 + \sigma_c'^2)^{1/2}$$

where

$$\begin{aligned} \sigma_b' &= \sin^{-1}(\sin(b[\sigma])/\cos(b[\text{dec}])) \\ \sigma_c' &= \sin^{-1}(\sin(c[\sigma])/\cos(c[\text{dec}])) \end{aligned}$$

## The Data Structure for the Active List

As deletion and insertion both occur in ascending order by declination, the active list can be implemented as a FIFO (first in, first out) queue with head and tail pointers. To support access by a lower bound on right ascension and traversal to the upper bound, a threaded binary tree on right ascension is used. The specific actions are to identify the active list member with the largest right ascension less than the lower bound of the range and then traverse the tree to the upper bound.

Analysis of the expected size of the active list shows that it is small and can be held in a main-memory data structure. The expected length of the active list is  $O(M)$  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  $2 * (\max(b[\sigma]) + \max(c[\sigma]) / 60$  (assuming that  $b[\sigma]$  and  $c[\sigma]$  values are expressed in arcminutes). The area of this strip may be calculated as  $4\pi \sin(\sigma)$ , where  $\sigma = (\max(b[\sigma]) + \max(c[\sigma])/60$ . As a proportion of the total surface area of the sphere, this area is:

$$\sin((\max(b[\sigma]) + \max(c[\sigma]))/60)$$

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

$$M \sin((\max(b[\sigma]) + \max(c[\sigma]))/60)$$

For a catalogue of  $10^9$  objects and with representative values of 1 arcminute for both  $\max(b[\sigma])$  and  $\max(c[\sigma])$ , the active list would have around 580000 elements. Allowing eight bytes for each of the record fields and four bytes for each of the list pointers, we calculate the space required for storage of the list at about 20Mb, with the binary tree index requiring less than 7 Mb. The requirement for 27Mb of main memory is fairly moderate in the context of current systems; this confirms that the active list can be maintained as a main-memory data structure for the catalogues of sizes currently available or envisaged.

Sweeping by declination is more efficient than sweeping by right ascension. In terms of our algorithm, sweeping by right ascension, starting at  $ra = 0$  (for example), would require holding in the active list the objects that could match objects on the other side of the line  $ra = 0$ . This would include the objects with small right ascensions and with declinations placing the objects near a pole and so with ellipses with large ranges in right ascension.

#### 4 Analysis of Performance

In I/O costs, the algorithm is trivially  $O(N \log N + M \log M)$  for the preprocessing stage and  $O(N+M)$  for the matching itself. For processor costs for the matching, the algorithm is  $O(N \log M)$  through the access and maintenance costs of the binary tree.

To perform empirical performance assessment, the algorithm has been implemented in a form notionally aimed at cases where matching is to be performed essentially as a standalone operation, i.e. the input catalogues are in their distribution forms, and they are not required at the completion of the matching operation. We measured the costs of both the preprocessing and the matching operation proper.

To allow a clear assessment of the effects of catalogue size in terms of number of objects, during the pre-processing for performance evaluation, we stripped attributes other than the object id, declination, right ascension and imprecisions. Our pre-processing program performs a main-memory sort, creates a binary file and determines the greatest

imprecision of any object in the catalogue. The program is customized to the format of the distribution files.

For the tests reported below, the pre-processing and catalogue matching programs were implemented using g++ under Debian Linux 2.4.20 and run on a Dual Pentium Xeon 2Ghz processor with 2 Gbytes of main memory. The disk was 5 RAID5 storage arrays with 10K RPM UltraSCSI disks with a maximum transfer rate between the RAID5s to the server of 80 Mb/s. For further details see [5].

Performance was assessed by pairwise matching of 6 catalogues and by matching each catalogue with itself. Summary information on these catalogues is given in Table 1. The smaller catalogues, 1XMM, SUMSS, Tycho2, are distributed as a single text file. The USNO catalogues are distributed as a set of files that are disjoint by declination. It is then only necessary to sort each file by declination. The catalogue matching program reads the set of files in sequence. 2MASS is also partitioned, but the partitioning creates files of equal size, with the exception of the last, and adjacent partitions overlap in their declination ranges. We resolve this by sorting each file by declination during preprocessing, and reading from two files during the cross matching, selecting the lower (by declination) record for action.

Preprocessing costs are given in Table 2 and the costs of matching in Table 3. Table 3 shows solution times for (as an example) adopting Tycho2 as the catalogue to be passed through the active list in a match against 2MASS and for adopting 2MASS as the catalogue to be passed through the active list. The rows of the table show the catalogue passed through the active list. For example, matching Tycho2 against 2MASS with Tycho2 passing through the active list has a solution cost of 14 minutes.

**Table 1**  
**Test Catalogues**

<b>Catalogue</b>	<b>N records</b>	<b>Max Imprecision (Degrees)</b>
1XMM [7]	56,711	0.015615
SUMSS [8]	134,870	0.006167
Tycho2 [9]	2,539,913	0.000051
2MASS [10]	470,992,970	0.000336
USNO A2 [11]	526,280,881	0.000056
USNO B1 [11]	1,045,175,762	0.000278

**Table2**  
**Pre-processing Costs**  
**Times given as mm:ss**

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 3**  
**Matching costs as elapsed time. Times given as mm:ss.**

	1XMM	SUMSS	Tycho2	2MASS	USNO A2	USNO B1
1XMM	0:01	0:01	0:04	13:55	14:59	30:51
SUMSS	0:01	0:01	0:03	8:36	10:02	19:06
Tycho2	0:04	0:03	0:07	14:31	16:02	32:36
2MASS	67:13	33:03	49:38	92:21	90:43	148:41
USNO A2	63:13	36:15	24:23	81:39	47:17	136:14
USNO B1	252:16	108:25	199:31	238:43	200:13	281:29

**Table 4**  
**Maximum Lengths of Active List**

	1XMM	SUMSS	Tycho2	2MASS	USNO A2	USNO B1
1XMM	342	305	299	300	300	300
SUMSS	221	164	137	137	137	137
Tycho2	1206	476	19	41	21	46
2MASS	226766	108385	7429	9602	7485	9494
USNO A2	308395	136247	2037	8516	2339	7922
USNO B1	468456	252001	14363	20683	14492	20012



## 5 Related Work

Reported algorithms for the catalogue matching problem have been based on exhaustive enumeration and by index-based joins. Exhaustive enumeration (the comparison of all possible pairs of objects) has been reported, for example by Malik et al [13] as the basis for the XMATCH operation in SkyQuery. This approach is clearly  $O(NM)$  in processor performance and  $O(N+M)$  in I/O if the catalogues can be held in main memory. The approach can be readily parallelised by binning or zoning to restrict pair-wise comparisons by subregions [13,17].

Index-based joins have been reported by Kunzst [12], Page [14] and Kalpakis et al [15]. Kunzst et al suggest a nested loops strategy. For each object in one of the catalogues, a search is made, using an index on HTM keys, for objects with keys corresponding to the same HTM cell or neighbouring cells. Page has performed performance evaluations of matches based on one-dimensional keys such as HTM and HEALPix [16,18] and with R-trees. The preprocessing times to load data and generate the R-tree indexes for two data sets of 5M objects (2MASS) and 3.6M objects (a subset of USNO B1) were 617 seconds and 442 seconds respectively. The match operation itself was 283 seconds.

The catalogue matching problem is closely similar to the neighbour-finding problem posed by Gray et al [17]. This deals with finding all pairs of objects, from a single catalogues that are within a specified distance of each other. Gray reports an approach using exhaustive enumeration with zoning with a performance of 2800 objects per second on a test problem of 154000 objects. The neighbour-finding problem is present in the Computational Geometry literature as the fixed-radius all neighbours problem. Optimal algorithms with  $O(N\log N + k)$  performance, where  $k$  is the number of pairs found, have been reported applying techniques with high pre-processing costs to generate a degraded- $\delta$  grid [18] or a Voronoi tessellation. [19,20].

Any comparison of performance data must be tentative. The data for other algorithms is typically for a single problem only, and the implementation environments differ widely. Comparison of the performance reported here and those of the methods of Gray et al and of Page for problems of comparable size indicates that our algorithm is faster by at least an order of magnitude for moderate-sized problems, and consideration of the orders of complexity suggests that the advantage would be higher for large problems. However, it is unclear if implementation within a database management system kernel is advantageous or disadvantageous in terms of raw performance. Certainly there are collateral benefits in a database implementation in the database system's ability to optimise complex operations involving a join and a selection (for example).

## 6. Conclusions

We have 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. Empirical performance assessment shows solution times of around 6 hours elapsed to

match two catalogues, one of a billion objects, the other half a billion. This appears to be a significant improvement over current algorithms.

This performance tends to confirm the feasibility of some key operations for Virtual Observatories. It suggests that, within current computing technology, it is readily possible to integrate very large catalogues. The logarithmic performance and the low main-memory requirements of our algorithm indicate a capacity to handle catalogues much larger than those used in the tests. The performance also has ramifications for searches across a Virtual Observatory for objects with certain characteristics as provided by, for example, SkyQuery. The assessments show that join operations of the sizes likely to be present in matching a set of candidates developed in prior steps with a catalogue can be performed in 10 seconds or less. This makes it feasible to perform searches on all of the sky through queries joining multiple databases to integrate, for example, data across the electromagnetic spectrum.

There are opportunities for further development of the algorithm. Because the algorithm is processor-bound, parallelisation using the zoning approach of Gray et al [17] is likely to reduce solution times significantly. The algorithm is also structured around a full serial scan of the catalogues, and this is inefficient where the sizes of the data sets are significantly different. A case in point is within search across multiple databases where the list of candidates could be small compared to a catalogue.

## References

- [1] Budavári, T., Malik, T., Szalay, A. S., Thakar, A. R., & Gray, J. 2003, in ASP Conf. Ser., Vol. 295 Astronomical Data Analysis Software and Systems XII, eds. H. E. Payne, R. I. Jedrzejewski, & R. N. Hook (San Francisco: ASP), 31
- [2] <http://surveys.roe.ac.uk/ssa/>
- [3] R.H. Güting, An Introduction to Spatial Database Systems. *VLDB Journal* 3 (1994), 357-399.
- [4] Preparata, F. and Shamos, M., Computational Geometry, an Introduction, Springer, New York, 1985..
- [5] Power, R., Devereux, D., Benchmarking Catalogue Cross Matching, TR-04/1848, CSIRO ICT Centre, Canberra, Australia, 2004.
- [6] Devereux, D, Abel, D.J, Power, R.A, and Lamb, P.R., Notes on the Implementation of Catalogue Cross Matching. TR-04/1847, CSIRO ICT Centre, Perth, Australia, 2004.
- [7] XMM-Newton Survey Science Centre, *The First XMM-Newton Serendipitous Source Catalogue*, 2003, <http://xmmssc-www.star.le.ac.uk/>

- [8] Bock, D. C.-J., Large, M.I., and Sadler, E.M., SUMSS: a wide-field radio imaging survey of the Southern Sky. I. Science Goals, survey design, and instrumentation, *The Astronomical Journal*, 117: 1578-1593, 1999.
- [9] Høg, E.; Fabricius, C.; Makarov, V. V.; Urban, S.; Corbin, T.; Wycoff, G.; Bastian, U.; Schwekendiek, P.; Wicenec, A., *The Tycho-2 catalogue of the 2.5 million brightest stars*, *Astronomy & Astrophysics*, v.355, p.L27-L30, 2000.
- [10] <http://www.ipac.caltech.edu/2mass/>
- [11] <http://ftp.nofs.navy.mil/projects/pmm/catalogs.html>
- [12] Kunszt, P.Z., Szalay, A.S., Thakar, A.R., The Hierarchical Triangular Mesh, in *Mining the Sky: Proc. of the MPA/ESO/MPE workshop*, Garching, A.J.Banday, S. Zaroubi, M. Bartelmann (ed.); (Springer-Verlag Berlin Heidelberg), 631-637 (2001).
- [13] Malik, T., Szalay, A. S., Budavári, T., Thakar, A.R., 2002 CIDR '03, 17, SkyQuery: A Webservice Approach to Federate Databases.
- [14] Page, C.G. A new way of joining source catalogs using a relational database management system, ADASS XII, ASP Conference Series, Vol 295, 39-42 2003, H.E. Payne, R.I. Jedrzejewski and R.N. Hopok (eds)
- [15] Kalpakis, K., Riggs, M., Pasad, M., Puttagunta, V., & Behnke, J. 2001, in ASP Conf. Ser., Vol. 238, E. Payne (San Francisco: ASP), 133-137.
- [16] Analysis Issues for Large CMB Data Sets, Krzysztof M. Górski, Eric Hivon, Benjamin D. Wandelt (1999), in *Proceedings of the MPA/ESO Cosmology Conference "Evolution of Large-Scale Structure"*, eds. A.J. Banday, R.S. Sheth and L. Da Costa, PrintPartners Ipskamp, NL, pp. 37-42
- [17] Jim Gray; Alexander S. Szalay; Aniruddha R. Thakar; Gyorgy Fekete Fekete; William O'Mullane; Gerd Heber; Arnold H. Rots, ***There Goes the Neighborhood: Relational Algebra for Spatial Data Search***, Microsoft Research Technical Report MSR-TR-2004-32, April 2004, 11 p.
- [18] O'Mullane W., Banday A.J., Gorski K.M., Kunszt P., Szalay A.S., 2001, in *Mining the sky*, proceedings of the MPA/ESO/MPE workshop held at Garching, Germany, 31 July - 4 August 2000. edited by A.J. Banday, S.Zaroubi, and M. Bartelmann. (Heidelberg: Springer-Verlag), 2001, p. 638.
- [19] M.T. Dickerson and D. Eppstein, Algorithms for proximity problems in higher dimensions, *Computational Geometry, Theory and Applications* 5 (1996), pp 277-291
- [20] M.T. Dickerson and R.L. Drysdale, Fixed radius search problem for points and segments, *Information Processing Letters* 35 (1990), 269-273