Algorithms for Large-Scale Astronomical Problems

Bin Fu

January 2011

CMU-CS-11-XXX

Computer Science Department

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

Keywords: Distributed Algorithms, Data Mining, eScience, Astronomy Applications

Abstract

Modern astronomical and cosmological datasets are getting larger and larger, including billions of astronomical objects and taking up terabytes of disk space. However, many classical astrophysics applications do not scale to such data volumes, which raise the question: Can we use modern computer science techniques to help astrophysicists analyze large datasets?

In order to answer the question, we have applied distributed computing techniques and developed algorithms to provide fast scalable solutions. In this report we introduce our initial works on three astrophysics applications:

- We have developed a distributed version of the Friends-of-Friends technique, which is a standard astronomical application for analyzing clusters of galaxies. The distributed procedure can process tens of billions of objects, which makes it sufficiently powerful for modern astronomical datasets and cosmological simulations.
- The computation of correlation functions is a standard cosmological application for analyzing the distribution of matter in the universe. We have studied several approaches to this problem and developed an approximation procedure based on a combination of these approaches, which scales to massive datasets.
- When astronomers analyze telescope images, they match the observed objects to the catalog. We have developed a matching procedure that maintains a catalog with billions of objects and processes millions of observed objects per second.

I propose to extend our existing solutions, as well as address several new problems.

Contents

1. Introduction

Massive scientific datasets have recently become available, which enable a data-driven perspective for scientists to conduct research. Those datasets in astrophysics, bioinformatics, seismology and many other fields can be in terabytes and sometimes even petabytes. In astrophysics, digital surveys such as Sloan Digital Sky Survey [Abazajian *et al.*, 09] as well as multiple cosmological simulations such as [Heitmann *et al.*, 08] and [DiMatteo *et al.*, 08], produce datasets with billions of objects, and the processing of these data require development of new scalable algorithms.

The purpose of this thesis work is to help astrophysicists analyze the available massive data, by creating better algorithms and applying distributed computing technique. Current solutions to some astrophysical problems cannot process big datasets. For instance, in order to solve the Friends-of-Friends problem (FoF) [Huchra and Geller, 82], which is one of the standard astrophysics tools for analyzing the structure of the universe, existing sequential programs would require impractically large memory, which goes beyond the capacity of modern desktops and even most supercomputers. We make use of modern distributed computing architectures and software, such as Map-Reduce [Dean and Ghemawat, 04] and BigTable [Chang *et al.*, 06] to develop fast and scalable solutions.

In Section 3, we describe our initial work on three astronomical problems: in Section 3.1, we provide a distributed algorithm for identifying clusters of galaxies, i.e. the Friends-of-Friends problem; next in Section 3.2, we describe our work on Correlation Functions, which is a measurement to examine the distances between astronomical objects, for which we give an efficient approximation algorithm; in Section 3.3, we describe a catalog of astronomical objects that supports indexing and fast retrieval of objects in spherical coordinates.

We have developed and tested these three applications. While they are efficient and scalable, there are still opportunities for several major extensions. For example, we can extend the Friends-of-Friends algorithm to a more general tool, which will consider not only the positions of astronomical objects but also their masses and velocities, which will the first part of the proposed dissertation research (Section 4.1). The second part of the proposed work is to solve several other astrophysics problems. Two of them are described in Section 4.2: Quasar Detection is a specific machine learning problem and I will apply multiple machine learning techniques into it; Object History Tracking is a large-scale data processing application that requires distributed database techniques.

2. Literature Review

Friends-of-Friends:

The Friends-of-Friends (FoF) problem [Huchra and Geller, 82] is a basic technique used to analyze large-scale astronomical structures, such as clusters and superclusters of galaxies. Researchers have developed multiple sequential FoF algorithms, including FOF from the University of Washington (http://www-hpcc.astro.washington.edu/tools/fof.html)

There are also parallel FoF algorithms, including pHOP [Liu *et al*., 03], Ntropy [Gardner *et al.*, 06], HaloWorld [Pfitzer and Salmon, 96] and Amiga Halo Finder (AHF) [Gill *et al.*, 04]. All these techniques use Message Passing Interface [MPI, 93] or communication between machines. However, they are designed for parallel distributed memory machines with low-latency network. They further require the whole dataset to fit into memory, which severely limits their scalability and make them inapplicable to modern datasets with billions of objects.

Recently, Kwon *et al.* [Kwon *et al.*, 09] proposed a distributed technique based on Dryad [Isard *et al.*, 07], an infrastructure for parallel and distributed programming from Microsoft Research. They have reported the results for datasets with up to 1 billion objects using a computer cluster with sixty-four cores.

Correlation Functions:

[Peebles, 80] gives the following definition: *Given a random galaxy in a location, the* ***correlation function*** *describes the probability that another galaxy will be found within a given distance.* It provides a method to test models with different settings about the universe, thus very important for theoretical cosmology.

A number of computer scientists have studied this problem. For instance, Gray and Moore [Gray and Moore, 00] used *kd*-tree to speed up the computation, which is slightly faster than the brute-force approach ($O(N^{5/3})$ vs. $O(N^2)$, where N is the number of objects). However, it is still impractically slow for massive datasets. Belussi and Faloutsos [Belussi and Faloutsos, 95] applied Fractal method to compute a rough approximate the related distribution. While their algorithm runs in linear time, it does not provide sufficient accuracy of most astronomical applications.

Distant Quasar Detection

The detection of quasars from multicolor imaging data dates back to Sandage and Wyndham [Sandage and Wyndham, 65]. Richards *et al.* identifies quasars in five color bands (U, G, R, I and Z). Using Non-Parametric Bayesian Classification together with fast Kernel Density Estimation [Gray and More, 03], their algorithms achieve 65-95% precisions and 70-95% recalls, varying on different datasets [Richards *et al*., 02] [Richards *et al*., 04] [Richards *et al*., 09].

Despite the high precision, their solution is less powerful to identify extra-bright objects (due to interlopers like white dwarfs and faint low-metallicity F-stars) and objects with high redshifts (below 50% precision to objects with $z>2.2$, where z stands for redshift) [Richards *et al*., 09].

3. Initial Results

We present the three techniques we have developed, which include identification of galaxy clusters (Section 3.1) analysis of distances between galaxies (Section 3.2) and indexing of astronomical objects (Section 3.3).

3.1 DiscFinder: Identification of Galaxy Clusters

3.1.1 Background

Astrophysicists use sky surveys and cosmological simulations to analyze large-scale astronomical structures. One of the standard steps is to find clusters of astronomical objects, such as galaxy groups, clusters and superclusters. We consider a basic problem, called Friends-of-Friends (FoF), which uses a simple criterion (only positions of astronomical objects) to determine whether they are gravitationally bound. It has proved effective in identifying galaxy clusters and has been widely used by astrophysicists.

The FoF algorithm takes one parameter: a *linking length*, denoted τ. Its input is a set of astronomical objects, where each object i is represented by its coordinates (x_i, y_i, z_i) in three-dimensional Cartesian coordinate space.

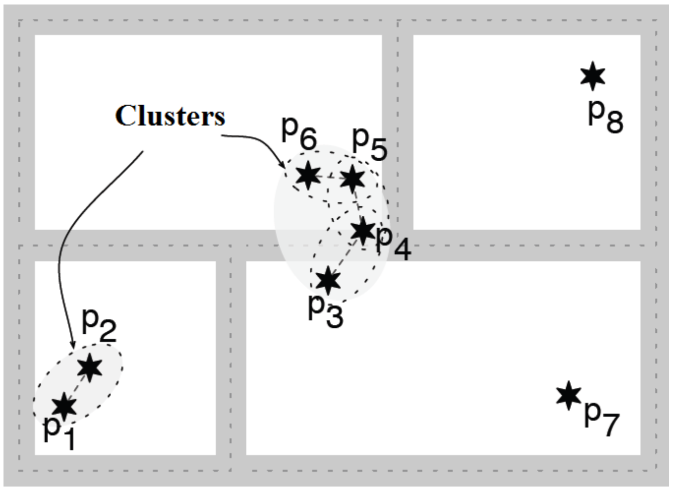


Figure 1. Example of Friends-of-Friends clusters and DiscFinder partition. In this example with eight objects, two clusters (p1, p2) and (p3, p4, p5, p6) are identified. Partition results provided by a *kd*-tree is also shown, where the space is divided into four regions. Shaded area represents the "shells" of regions, where each region is extended by $\tau/2$ on each side.

Two objects are considered "friends" if the distance between them is no greater than the linking length. The friendship between objects defines an undirected graph, and its connected components correspond to clusters of objects (Figure 1). For each object *i*, the FoF algorithm outputs the identifier of the cluster containing it.

3.1.2 Distributed algorithm

Sequential FoF procedures are inapplicable to the analysis of massive datasets because they require impractically time and memory. To improve the scalability of the FoF tools, we have developed a distributed version, called DiscFinder [Bin *et al.,* 10a], which is a Map-Reduce [Dean and Ghemawat, 04] wrapper that makes use of existing sequential FoF algorithms.

The underlying idea is to split an input dataset into several spatial regions, apply a sequential FoF algorithm to each region, and then merge the results from individual regions. DiscFinder manages these steps, including the load balancing in splitting phase and the merging phase.

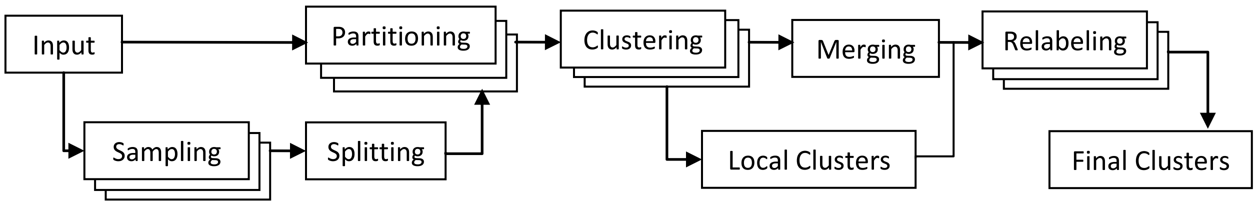


Figure 2. DiscFinder Pipeline

Figure 2 shows the DiscFinder pipeline. We use three Map-Reduce procedures (Sampling, Partitioning/Clustering, and Relabeling) to implement this distributed algorithm. Next we outline each step:

Sampling (parallel implementation, first MapReduce procedure) and **Splitting** (sequential): Astrophysics datasets are seldom uniformly distributed, so a naïve partition would cause load balancing problem, where each region has disparate number of objects. Although *kd*-tree data structure can create perfect partitions where each region contains exactly the same number of objects, it requires impractically memory (linear to the number of objects). So we first sample a small proportion of objects (typically 0.1% of all objects) in sampling phase, and construct a *kd*-tree using the sampled objects in splitting phase. This fast procedure results in a fairly balanced partition.

Partitioning (parallel, Map phase of the second MapReduce procedure): The *kd*-tree provides a spatial partitioning for the domain, which determines the region of each object. In order to handle the objects near region boundary, each region is extended by $\tau/2$ on each side. As a result, each region has a "shell" around it, which contains objects in its adjacent regions (see Figure 1).

Clustering (parallel, Reduce phase of the second MapReduce procedure): A sequential FoF procedure is invoked for each extended region. DiscFinder allows plugging in any sequential FoF implementations, such as FOF (http://www-hpcc.astro.washington.edu/tools/fof.html) or AFOF (http://www-hpcc.astro.washington.edu/tools/afof.html) from University of Washington. Cluster membership information of each object is outputted for each region. We refer these clusters as local clusters.

Merging (sequential): Output of the clustering step (object identifiers and their local cluster identifiers) is split into two separate sets: one for objects inside shells, one for other objects (Figure 3). Each in-shell object belongs to several regions, and thus it may be assigned to multiple local clusters within different regions. The purpose of merging is to combine the local clusters from different partitions that contain common objects.

It is achieved by the Union-Find procedure [Galler and Fisher, 64], which merges local clusters that have common objects. The time and space complexity of the Union-Find procedure are both near-linear. In our experiments the merging step takes only a small fraction (around 4%) of the total running time.

Relabeling (parallel, third MapReduce procedure): Since some local clusters are combined together in the merging phase, we use this step to reflect this effect. Relabeling step is a single pass over the output of the clustering step, where object from the combined clusters are relabeled to the same cluster.

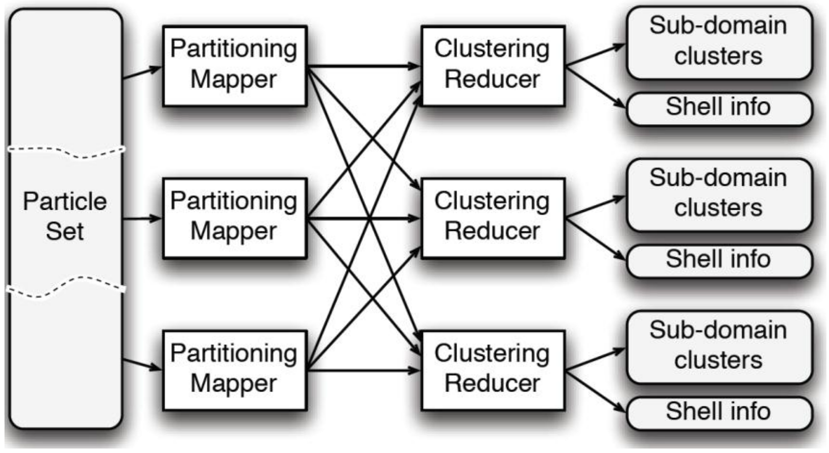


Figure 3. MapReduce framework for the partitioning and clustering phases in DiscFinder. Results of clustering phase are split to two parts: information of in-shell objects and the other objects (which formulate sub-domain clusters).

3.1.3 Data Sources and Computing Cluster

Data sources: For the experiments, we have used three cosmology simulation datasets provided by our astrophysicist collaborators: BHCosmo [DiMatteo *et al.*, 08], Coyote Universe [Heitmann *et al.*, 08] and DMKraken from McWilliams Center for Cosmology at Carnegie Mellon University. Each dataset has multiple "snapshots", which describe the simulated universe at different moments of time. These datasets are in the GADGET-2 format [Springel, 05], which is a binary format that represents multiple properties of cosmological objects including positions, velocities, accelerations and masses.

Name of the dataset	Num. object (millions)	Data size of each snapshot (GByte)	Number of snapshots	Total data size (GByte)
BHCosmo	20	1	22	22
Coyote Universe	1,100	32	20	640
DMKraken	14,700	500	28	14,000

Table 1. Data sources

Computing cluster: We have used a data-intensive cluster built in 2009 at Carnegie Mellon, called DISC/CLOUD (http://www2.pdl.cmu.edu/~twiki/cgi-bin/view/OpenCloud), which consists of 64 nodes. Each node has eight 2.8GHz CPU cores, 16GB memory and four 1TB SATA disks. The nodes are connected by a 10GigE network using Arista switches and QLogic adapters at the hosts. The cluster is configured with Linux (2.6.31), Hadoop (0.19.1) and Java (1.6).

3.1.4 Evaluation

Scalability

We have conducted a series of scalability experiments to determine how DiscFinder performs with different setups. The experiments have confirmed that it is effective for processing massive data, and can handle datasets much larger than the overall available memory.

In Figure 4, we show the results of strong scalability and weak scalability experiments. In the strong scalability experiments, the input size is constant while we gradually add computing resources. Figure 4(a) shows the dependency of the running time on the number of cores, both on logarithmic scale. The curves correspond to input sizes of 0.5, 1 and 14.7 billion objects. Linear scalability behaves a straight line with slope -1, indicating that running time decreases proportional to the number of cores. Our experiments show that, with small number of cores, running time actually suffers since each node takes on too much computing tasks; on the other side, with a large number of cores, each core performs too little work so framework overhead stands out.

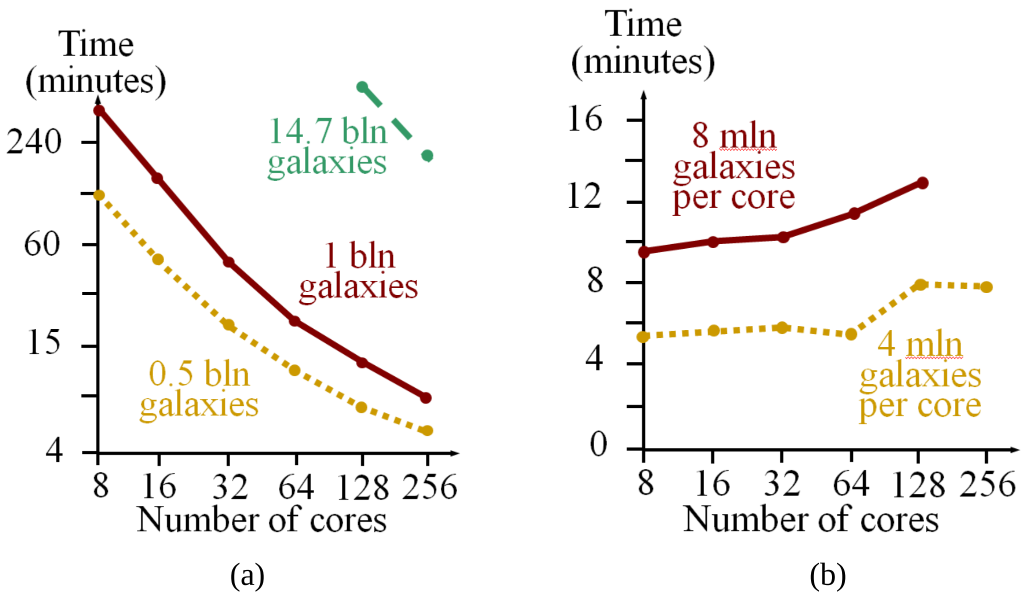


Figure 4. Strong scalability (a) and weak scalability (b)

In the weak scalability tests, each node is assigned the same amount of work; that is, the overall input size is proportional to the number of cores. In Figure 4(b), we plot the running time versus the number of cores. The two curves correspond to 4 and 8 million objects per core. Here only the horizontal axis is in logarithmic scale. In our experiments, both curves increase slightly, which is reasonable due to the framework overhead introduced by input/output and data movement.

Hadoop Performance

The original implementation of DiscFinder is very intuitive, but its running time far exceeds the initial expectation, because the framework spends significant time for data processing and other auxiliary processes. We have implemented a series of Hadoop-specific optimizations, which has resulted in an overall speedup by the factor of three in the experiment with 14.7 billion objects. Specifically, we have reduced the processing time from 16,609 seconds to 3,810 seconds.

3.2 DISC-Distance: Analysis of Distances between Galaxies

3.2.1 Background

In astrophysics, *Correlation Functions* (CF) is a measurement of the distribution of pairwise distances between astronomical objects. In this report, CF is defined as follows:

Given a set of object P = $\{p_1,p_2\ldots,p_N\}$ and a series of distance ranges $\{q_1,q_2\ldots,q_M\}$:

$q_1 = (d_0, d_1)$

$q_2 = (d_1, d_2)$

..

$q_M = (d_{M-1}, d_M)$

$d_i \in R$, $i = 0,1,...,M$

We want to calculate the number of pairs of objects whose distances fall in each of $q_i = (d_{i-1}, d_i)$, $i=1,2,\dots,M$:

$$CF((d_{i-1}, d_i)) = \#(p_u, p_v) \in P, \text{ s.t. } d_{i-1} \le d(p_u, p_v) \le d_i$$

Where d($\bullet$) represents the Euclidean distance between two objects.

In Figure 5, we show the correlation functions calculated on a dataset with 4.5 million object datasets, which is provided by our collaborators from McWilliams Center for Cosmology:

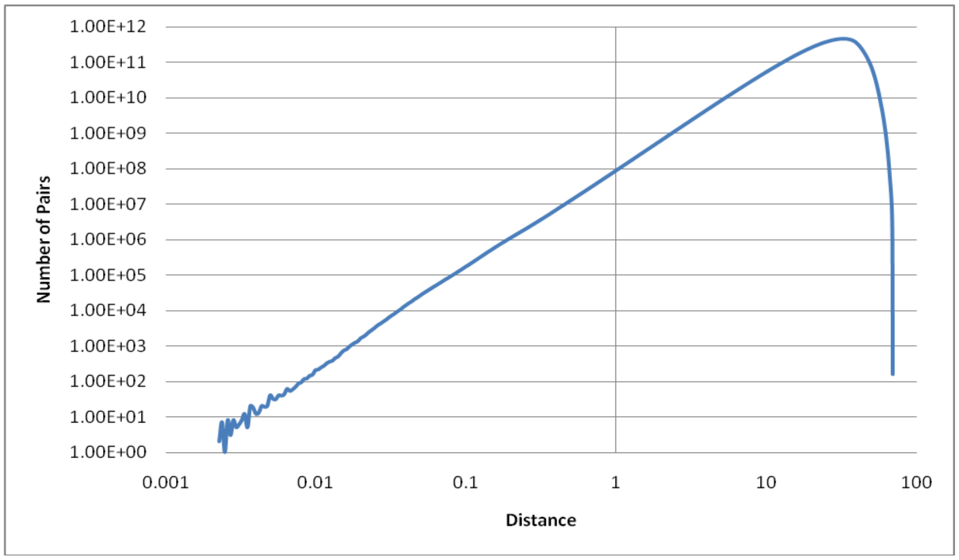


Figure 5. Correlation functions for a dataset with 4.5 million objects

3.2.2 Existing algorithms

To compute correlation functions, researchers have developed several exact and approximation algorithms. However, the existing exact algorithms are still impractically slow for large datasets, while the approximation techniques are impractically inaccurate [Bin *et al.*, 10b].

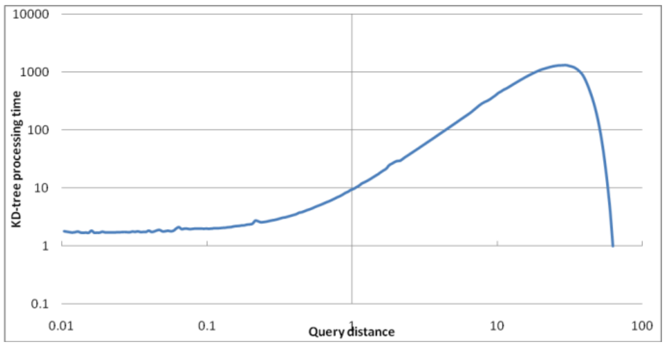
3.2.3 Proposed hybrid algorithms

We have proposed a hybrid algorithm to approximate correlation functions in [Bin *et al.*, 10b], by combining the *kd*-tree algorithm and a new sampling algorithm. The sampling algorithm randomly samples several subsets from the original dataset, and uses the sampled sets to approximate true results.

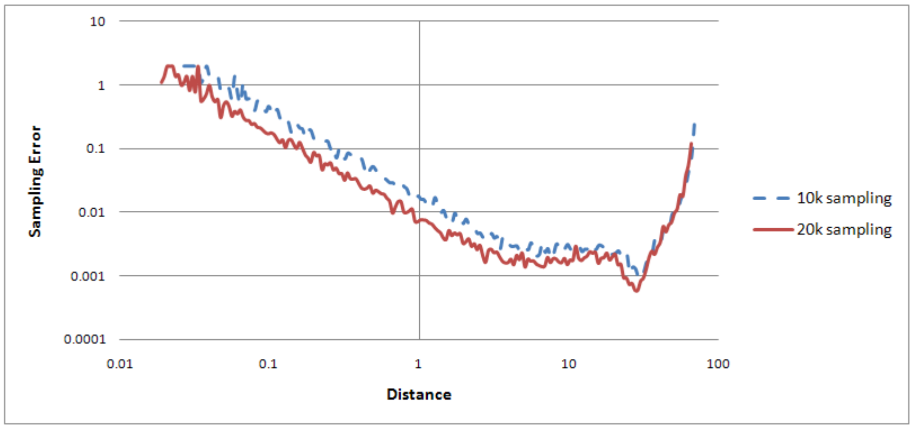
```
Sampling algorithm
Input: $p_1, p_2, \ldots, p_N$
       $q_1, q_2, \ldots, q_M$
       Number of sample objects S
Output: Mean and Standard Deviation of $c_1', c_2', \ldots, c_M'$
$c_k'$ is the estimated number of object-pair falling in $q_k$
T = 30; //T is the number of sampling iteration
Create a two dimensional array $a_{uv}$, $u = 1, 2, \ldots, M$;  $v=1, 2, \ldots, T$
for $i = 1$ to M
    for $j = i + 1$ to T
        $a_{ij} = 0$;
for t = 1 to T
    Randomly select S objects $u_1, u_2, \ldots, u_S$ from $p_1, p_2, \ldots, p_N$
    for $i = 1$ to S-1
        for $j = i + 1$ to S
            Find $q_k$ that $d(u_i, u_j)$ belongs to
            $a_{kt}$ ++;
    for $k = 1$ to M
        $a_{kt} = a_{kt} \bullet (N/S)^2$;
for $k$=1 to M
    $\mu(c_k') = (a_{k1} + a_{k2} + \ldots + a_{kT}) / T$
    $\sigma(c_k')^2 = ((a_{k1} - \mu(c_k'))^2 + (a_{k2} - \mu(c_k'))^2 + \ldots + (a_{kT} - \mu(c_k'))^2) / (T(T - 1))$
```

Our hybrid method combines the advantages of the *kd*-tree algorithm and the sampling algorithm. It applies *kd*-tree calculation at small and large range distances, because kd-tree is fast at both ends (Figure 6(a)); for range distances in the middle, sampling algorithm is invoked, where its approximation error is very small (Figure 6(b)).

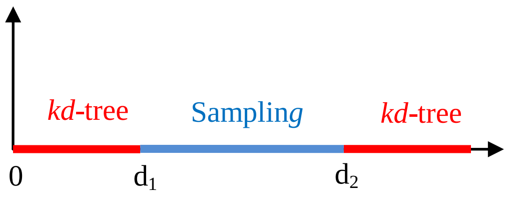
In Figure 7 we plot the running time of the brute-force algorithm, the *kd*-tree algorithm and our hybrid algorithm on the dataset with 4.5 million objects. The experiments are conducted on a single computer with two 2.66 GHz processors and 2GB memory. The hybrid method is an approximation algorithm. It takes an approximation error bound as input, which affects its running time. As shown in Figure 7, the hybrid method is faster than the existing exact algorithms, even when we require a very low error (0.1%).



(a) The running time.



(b) The approximation error.



(c) Hybrid approach.

Figure 6. The performance of the correlation functions computation. All experiments are conducted on an Intel Core2 Desktop with 2.66GHz CPU and 2GB memory. We show the running time of the *kd*-tree computation (a) as well as the approximation error of the sampling algorithm (b). The hybrid approach (c) is based on switching between the two algorithms based on the distance.

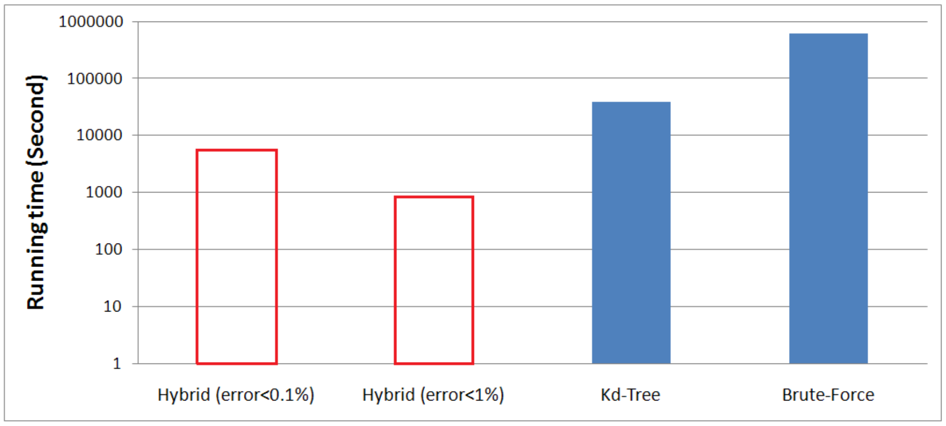


Figure 7. The time of computing the correlation functions on the dataset with 4.5 million object. We compare the performance of the hybrid algorithm (left) and that of the two exact algorithms (right).

3.3 SkyMap: Indexing of Astronomical Objects

3.3.1 Background

When astronomers analyze telescope images, they check whether newly observed objects appear in the catalog of known objects. Since positions of objects may change slightly due to atmospheric and optical distortions, astronomers need to retrieve close approximate matches. The current astronomical catalogs, e.g. Sloan Digital Sky Survey, contain about 230 million objects, and a straightforward linear matching would be impractically slow. We have developed a technique for indexing massive catalogs and fast matching of newly observed objects without the use of supercomputers.

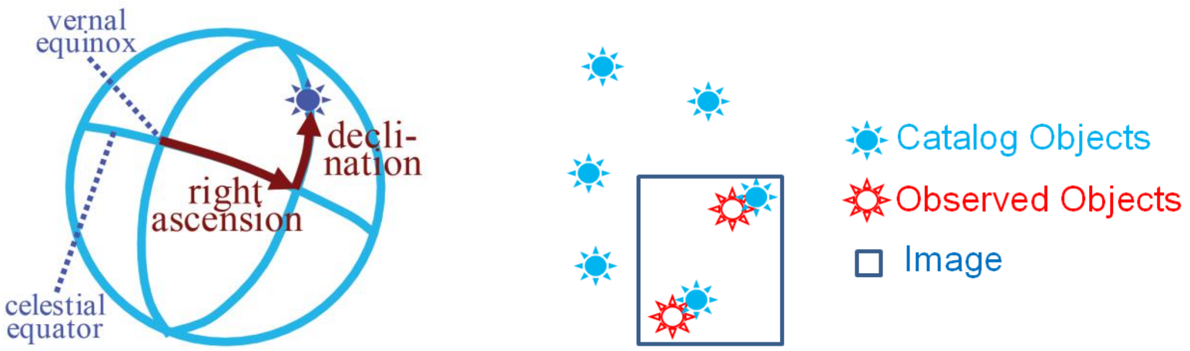
3.3.2 Problem

The astronomers describe the observed position of a celestial object by two values, which define its equatorial coordinates on the sky, called *Right Ascension* and *Declination*, which are similar to the longitude and latitude. Note that the catalogs usually do not include the data about the distance to an object, since it is not a directly observable value.

We have already acquired a catalog of the sky, which contains a list of objects observed previously. We have also obtained a number of newly arrived images. An image is typically a rectangular region of the sky, whose size is about 2.5×2.5 square degrees, with sides parallel to the celestial coordinate axes. An image may contain from a few hundred to a few tens of thousands objects, depending on the image size and telescope resolution. For each observed object *p*, we need to find a matching catalog object *q*, such that:

- p is the nearest object to q among the observed objects;
- q is the nearest object to p among the catalog objects; and
- The distance between p and q in two-dimensional spherical coordinates is at most 1 arc second, that is, 1/3600 degrees.

Since the modern catalogs are far too large for the memory of desktop computers, we have developed a technique for indexing catalogs on disk, which requires loading only a small contiguous part of the catalog into memory when processing a single image.



(a) (b)

Figure 8. (a) The location of an object is defined by its right ascension and declination. (b) An example to demonstrate the retrieval problem, which includes two newly observed objects that should be matched to two catalog objects.

3.3.3 Our solution

The developed system consists of two procedures: indexing procedure, which organized data on disk, and retrieval procedure, which identified the part of the catalog relevant to a given image, loads it into memory, and finds matches for all objects in the image.

Indexing:

Along the direction of right ascension, we divide the celestial sphere into strips, where each strip is 2 arcseconds wide, thus obtaining $(3600/2) \times 180 = 324{,}000$ strips. The objects within each strip are sorted by their right ascension. For each strip, we sort the list of its sorted objects as a separate file, thus obtaining 324,000 files.

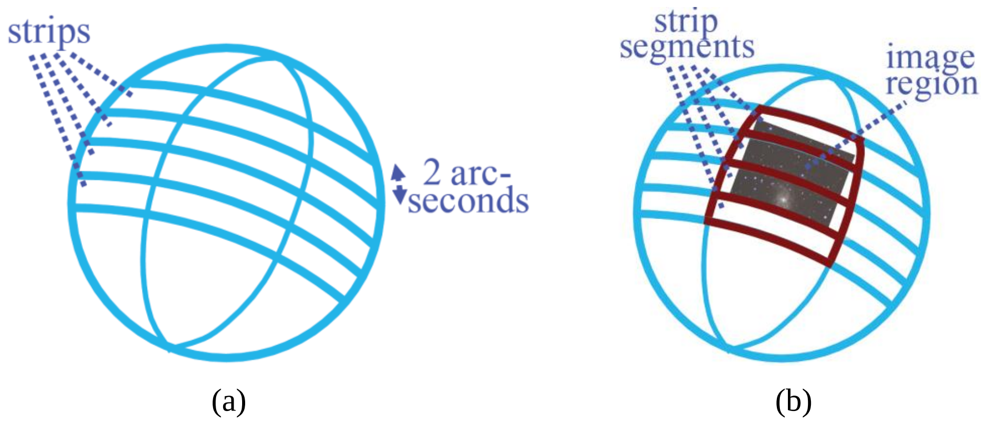


Figure 9. (a) Partition of the sky in equatorial coordinates. (b) Given an image, we first find its bounding box and identify the strip segments that cover the extended bounding box.

Retrieval:

Given an image, the first step is to calculate its bounding box. In order to find approximate matches, the bounding box needs to be extended by one arcsecond in each direction. We then retrieve the strip segments of the catalog that cover the extended box, by first identifying the related strips, and then finding the related strip segments. To select the related strips, we determine the declination range of the extended bounding box, and retrieve all strips that cover this range. Afterwards, since catalog objects are already sorted by their right ascension within each strip, we conduct binary search within each strip to locate a segment that covers the bounding box.

Consequently, for each image, only the relevant strip segments are retrieved and loaded to the memory. Typical image is small enough for a single computer to accommodate this information.

Finally, once necessary catalog objects available in memory, a simple and fast approximate matching procedure will be conducted to find match to each image object.

3.3.4 Experiments

We have conducted two sets of experiments to evaluate the running time of the retrieval procedure on a catalog of two billion objects. Experimental results come from desktop with 2.83GHz CPU and 16GB memory.

The first set of experiments, summarized in Figure 10(a), measures the time of loading the relevant part of the catalog into memory. For instance, if the image is 6 square degrees, which is a typical size of telescope images, the loading time is about half-minute. The running time grows linearly with the image size.

In the second set of experiments, we have evaluated the time to find the catalog match for each object in the image. In Figure 10(b), it is clear that this step is particularly fast. For instance, it takes less than one second to find matches for ten thousand objects in an image.

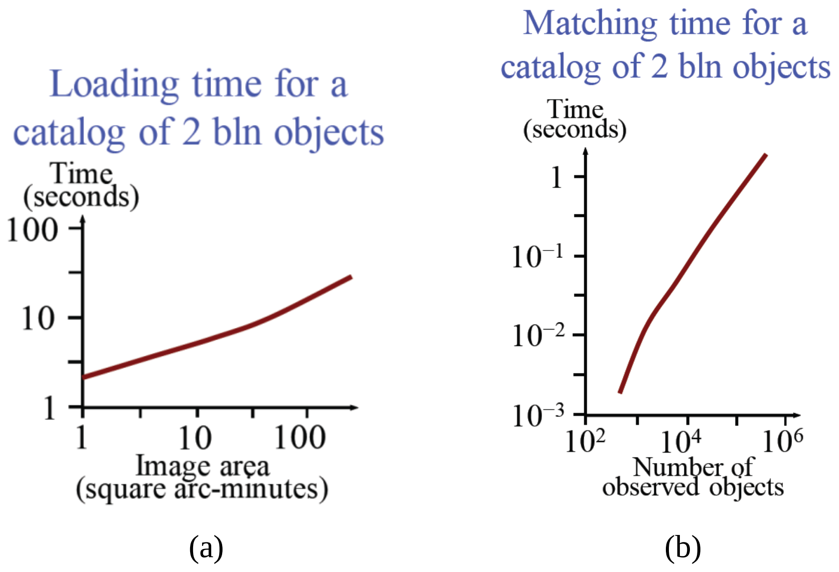


Figure 10. (a) The time of identifying the relevant part of the catalog and loading it into memory. (b) The time of finding matches after loading the catalog.

4. Proposed Work

I will now describe the proposed future work. In Section 4.1, I will discuss some potential extensions to each of my initial results. In Section 4.2, I will propose new astrophysics applications, which will be part of my thesis research.

4.1 Major Extensions

4.1.1 Identification of Galaxy Clusters

We have shown in Section 3.1 that DiscFinder scales to datasets with billions of objects; however, there are several open problems not addressed in the initial work, and we now propose to address these problems and make DiscFinder a more general tool.

In particular, the current version of DiscFinder is less efficient for a large linking length (τ). Each region is extended by $\tau/2$, which causes the drop in efficiency for large τ. Another problem of using a large linking length is the increase in the number of in-shell objects. Consequently the merging step may be a bottleneck: although the current Union-Find implementation has linear complexity, it is still a sequential algorithm with limitation at both time and memory. Currently in our experiments the linking length is typically set to 0.02% of the length of the diameter of the overall simulated universe, which is the typical value used by astronomers. We plan to extend the algorithm to allow significantly larger values without the loss of efficiency.

Also, current DiscFinder framework only works for FoF algorithms. There are more complex group finding algorithms that consider not only the positions of objects, but also their other attributes, such as mass and velocity. We plan to extend DiscFinder to support a distributed version of those algorithms.

4.1.2 Analysis of Distances between Galaxies

The problem in Section 3.2 is called *Two-Point Correlation Functions* (2PCF) because we examine distances between pairs of points. A related more challenging problem is *Three-Point Correlation Functions* (3PCF), where we examine distances among triples of points. A 3PCF takes on three distance ranges as input: (x_i, x_{i+1}), (y_j, y_{j+1}) and (z_k, z_{k+1}). Given a set of point $P = \{p_1, p_2, ..., p_N\}$:

$$\text{3PCF}((x_i, x_{i+1}), (y_j, y_{j+1}), (z_k, z_{k+1})) = \#(p_u, p_v, p_w \in P), \text{ s.t.}$$
$$x_i \le d(p_u, p_v) \le x_{i+1},\ y_j \le d(p_u, p_w) \le y_{j+1} \text{ and } z_k \le d(p_v, p_w) \le z_{k+1}$$

The amount of computation required for 3PCF is much greater than that for 2PCF. I plan to extend the developed hybrid algorithm to address the 3PCF problem.

Another direction is to improve the precision of the hybrid algorithm. Astrophysicists are in very accurate results, with the error below 0.1%

4.1.3 Indexing of Astronomical Objects

Although the running time of the developed matching algorithm is low, an even larger catalog and more massive image data may require further performance improvements. To address this problem, I propose to develop a distributed version of the matching algorithm. We will study the following two approaches to distributed matching.

- **Image Partition**. We may keep a copy of the whole catalog on multiple nodes and process different images on different nodes. This approach requires more data movement between nodes, but it can easily handle the load balancing.

- **Catalog Partition**. Alternatively, we may divide the catalog among nodes. Given a new image, the system will identify the related parts of the catalog and send it to the respective nodes. This approach will help to reduce data movement, but it may not always provide appropriate load balancing.

4.2 Other Problems

4.2.1 Distant Quasar Detection

We now describe another astrophysics problem, specifically, detection of distant quasars. We plan to apply various machine learning techniques into this problem.

Problem

A quasar is an unusually bright galaxy, which is visible from very large distances. Astronomers and cosmologists are interested in the properties of distant quasars because they provide information about remote regions of the universe, thus enables the study of the universe expansion.

An accurate identification of distant quasars is hard, because regular telescopes do not explicitly provide the distance information for an object, and quasars look similar to stars and regular galaxies. We will help astrophysicists better classify distant quasars. Specifically, given a dataset of astronomical objects, where only a small amount of them are labeled, we need to identify distant quasars in the unlabeled objects.

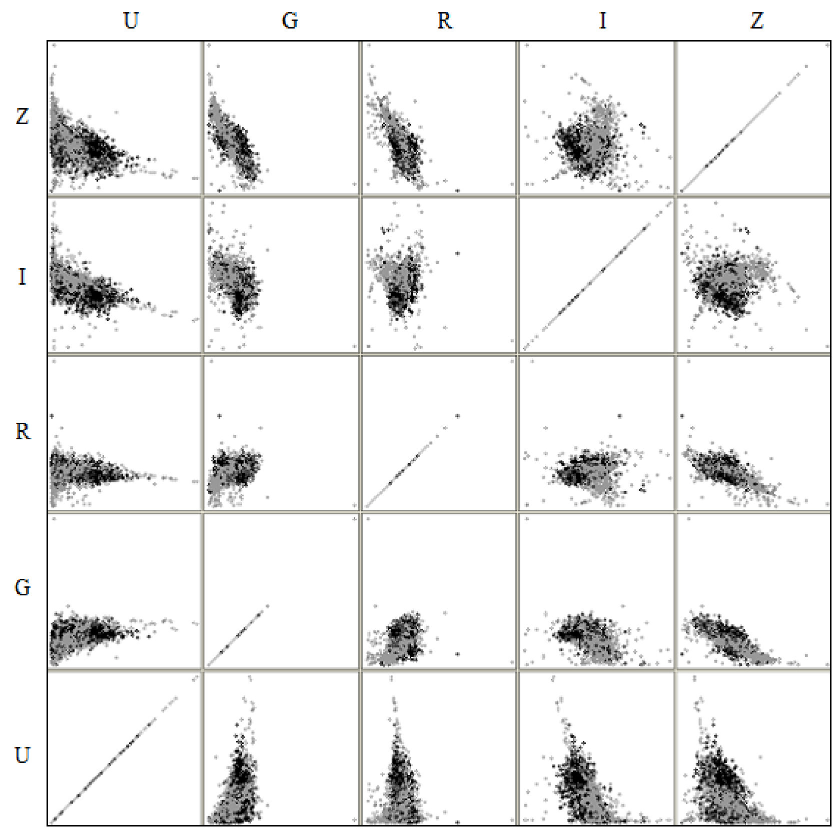


Figure 11. Distribution of quasars (black) and non-quasars (grey) in the space of five color bands (U, G, R, I and Z) on the sample dataset with 8,200 objects.

We will describe an astronomical object by five numeric values, which represent its brightness levels in five different color bands. In Figure 11, we show the distribution of objects on a sample dataset with 8,200 objects.

Proposed solutions

We propose to apply three machine learning techniques to this problem.

Supervised Learning: This is the base and simple case, where only labeled data is used to train a classifier. We will experiment with a variety of machine learning algorithms, including decision trees, support vector machines, *k*-means clustering, and nearest neighbors.

Semi-Supervised Learning: In this approach, a large amount of unlabeled data is used for training. [Zhu, 05] has shown that unlabeled data usually helps improve the classification precision. We will evaluate several semi-supervised algorithms, including self-training and Multiview training.

Active Learning: While we can request astronomers to acquire additional labels, it is a costly process. Specifically, the labeling of an astronomical object requires additional measurements, which cost several dollars per object. Active learning is used to handle this kind of situations: assuming user can provide a small amount of additional labels, active learning algorithm select appropriate objects of labeling [Settles, 10]. We plan to apply several active learning algorithms, such as uncertain sampling and query-by-committee algorithm.

4.2.2 Object History Tracking

Modern cosmological simulations generate massive data. For example, our collaborators from McWilliams Center for Cosmology produce a 3 terabytes per snapshot, with 27 snapshots per simulation. We propose to address an object history tracking problem: how to keep track of objects across snapshots. I will apply modern distributed database techniques, such as HBase [Chang *et al.*, 06], to store all 80 terabytes data and support fast and flexible query operations.

I will leave this as an optional task and consider working on it after finishing other proposed tasks.

4.3 Timeline

I plan to complete the proposed works according to the following tentative timeline.

- December 2010: Thesis proposal.
- January – April 2011: Distant quasar detection.
- May – August 2011: Improvements to the indexing of astronomical objects.
- September – December 2011: Improvements to the identification of galaxy clusters.
- January – April 2012: Improvements to the analysis of distances between objects.
- May – August 2012: Object history tracking.
- September – November 2012: Write the thesis.
- December 2012: Thesis defense.

Bibliography

[Abazajian *et al*., 09] Kevork N. Abazajian *et al*. The Seventh Data Release of the Sloan Digital Sky Survey. *The Astrophysical Journal Supplement Series*, 182(2), pages 543-558, 2009.

[Belussi and Faloutsos, 95] Alberto Belussi and Christos Faloutsos. Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension. In *Proceedings of the 21th International Conference on Very Large Data Bases*, pages 299-310, 1995.

[Bin *et al.*, 10a] Bin Fu, Kai Ren, Julio Lopez, Eugene Fink, and Garth Gibson. DiscFinder: A data-intensive scalable cluster finder for astrophysics. In *Proceedings of the 20th ACM International Symposium on High Performance Distributed Computing*, 2010.

[Bin *et al.*, 10b] Bin Fu, Eugene Fink, Julio Lopez and Garth Gibson. Disc-Distance: Fast Scalable Calculation of the Distribution of All-Pair Distance.

[Chang *et al.*, 06] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, 7:205-218, 2006.

[Dean and Ghemawat, 04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings the 6th Symposium on Operating System Design and Implemention*, pages 137–150, 2004.

[DiMatteo *et al.*, 08] Tiziana Di Matteo, Jörg Colberg, Volker Springel, Lars Hernquist, and Debora Sijacki. Direct cosmological simulations of the growth of black holes and galaxies. *The Astrophysical Journal*, 676(2), 2008.

[Galler and Fisher, 64] Bernard A. Galler and Michael J. Fischer. An improved equivalence algorithm. *Communications of the ACM*, 7(5):301–303, 1964.

[Gardner *et al.*, 06] Jeffrey P. Gardner, Andrew Connolly, and Cameron McBride. A Framework for Analyzing Massive Astrophyisical Datasets on a Distributed Grid. In *Proceedings of Astronomical Data Analysis Software and Systems XVI ASP Conference Series*, Volume 376-379, pages 69, 2006.

[Gray and Moore, 00] Alexander Gray and Andrew Moore. 'N-body' Problems in Statistical Learning. *Advances in Neural Information Processing Systems 13*, pages 521-527. MIT Press, 2000.

[Gray and More, 03] Alexander Gray and Andrew Moore. Nonparametric density estimation: Toward computational tractability. In *Proceedings of the Third SIAM International conference on Data Mining*, 2003.

[Gill *et al.,* 04] Stuart P.D. Gill, Alexander Knebe, and Brad Gibson. The evolution of substructure - I. A New Identification Method. *Monthly Notices of the Royal Astronomical Society*, 351(2):399–409, 2004

[Heitmann *et al.,* 08] Katrin Heitmann, Martin White, Christian Wagner, Salman Habib, and David Higdon. The Coyote Universe I: Precision Determination of the Nonlinear Matter Power Spectrum. *The Astrophysical Journal*, 715(1), 2008.

[Huchra and Geller, 82] John Huchra and Margaret J. Geller. Groups of galaxies. I – Nearby groups. *The Astrophysical Journal*, 257:423–437, 1982.

[Isard *et al.,* 07] Michael Isard , Mihai Budiu , Yuan Yu , Andrew Birrell , and Dennis Fetterly, Dryad: distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, pages 59-72, 2007.

[Kwon *et al.,* 09] YongChul Kwon, Dylan Nunley, Jeffrey P. Gardner, Magdalena Balazinska, Bill Howe and Sarah Loebman. Scalable clustering algorithm for n-body simulations in a shared-nothing cluster. Technical Report UW-CSE-09-06-01, University of Washington, June 2009.

[Liu *et al.*, 03] Ying Liu, Wei-keng Liao, and Alok Choudhary. Design and Evaluation of A Parallel HOP Clustering Algorithm for Cosmological Simulation. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, pages 82-89, 2003.

[MPI, 93] MPI Forum. MPI: A Message Passing Interface. In *Proceedings of the ACM/IEEE conference on Supercomputing*, pages 878-883, 1993.

[Peebles, 80] Jim Peebles. *The Large Scale Structure of the Universe*. Princeton University Press. 1980

[Pfitzer and Salmon, 96] David W. Pfitzner and John K. Salmon. Parallel Halo Finding in N-body Cosmology Simulations. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 208-213, 1996.

[Richards *et al*., 02] Gordon T. Richards et al., Spectroscopic target selection in the Sloan digital sky survey: The quasar sample. *The Astronomical Journal*, 123:2945, 2002.

[Richards *et al*., 04] Gordon T. Richards et al., Efficient photometric selection of quasars from the Sloan digital sky survey: 100,000 z<3 quasars from Data Release One. *The Astrophysics Journal Supplement Series*, 155:257-269, 2004.

[Richards *et al*., 09] Gordon T. Richards et al., Efficient photometric selection of quasars from the Sloan digital sky survey: II ~1,000,000 quasars from Data Release Six. *The Astrophysics Journal Supplement Series,* 180:67, 2009.

[Sandage and Wyndham, 65] Allan Sandage and John D. Wyndham. On the Optical Identification of Eleven New Quasi-Stellar Radio Sources. *The Astrophysics Journal,* 141:328-332, 1965.

[Settles, 10] Burr Settles. Active Learning Literature Survey. Computer Science Technical Report 1648, University of Wisconsin-Madison, 2010.

[Springel, 05] Volker Springel. The Cosmological Simulation Code GADGET-2. *Monthly Notices of the Royal Astronomical Society*, 364(4):1105–1134, 2005

[Zhu, 05] Xiaolin Zhu. Semi-Supervised Learning Literature Survey. Computer Science Technical Report 1530, University of Wisconsin-Madison, 2005.