ISRN TELCORDIA--2011-03+PR-0GARAU
**Prepared for: Office of Naval Research**

# Fast Multiscale Algorithms for Information Representation and Fusion

## Technical Progress Report No. 3

Devasis Bassu, Principal Investigator

Contract: N00014-10-C-0176

Telcordia Technologies

One Telcordia Drive

Piscataway, NJ 08854-4157

April 2011

| Report Documentation Page | *Form Approved* *OMB No. 0704-0188* |
|---|---|

| | | |
|---|---|---|
| 1. REPORT DATE **APR 2011** | 2. REPORT TYPE | 3. DATES COVERED **00-00-2011 to 00-00-2011** |
| 4. TITLE AND SUBTITLE **Fast Multiscale Algorithms For Information Representation And Fusion** | | 5a. CONTRACT NUMBER |
| | | 5b. GRANT NUMBER |
| | | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | | 5d. PROJECT NUMBER |
| | | 5e. TASK NUMBER |
| | | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Telcordia Technologies,One Telcordia Drive,Piscataway,NJ,08854** | | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | | 10. SPONSOR/MONITOR’S ACRONYM(S) |
| | | 11. SPONSOR/MONITOR’S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES

14. ABSTRACT
**In the third quarter of the work effort, we focused on the research and design of the randomized Approximate Nearest Neighbors (ANN) algorithm. This randomized variant of the ANN algorithm has theoretically proven improvements in the number of data dimensions that it can handle over existing algorithms and meets the theoretical lower bounds for computational complexity. Algorithm designs for computing the Randomized Approximate Nearest Neighbors (ANN) using randomized Fast Fourier Transform projections were completed. Fortran 95 interface for reusable randomized ANN routine has been defined and implemented. The randomized ANN implementation uses BLAS libraries via standardized interfaces to make optimal use of hardware resources (e.g., multiple cores, CPU cache) in addition to using the OpenMP standard (for parallel execution of code). Use of these standards enables the code to be built flexibly in a number of ways on various target platforms. Preliminary testing of the software is complete. Additional updating, fine tuning will be based on results from various experiments that will be conducted in the upcoming quarter. The project is currently on track ? in the upcoming quarter, we will focus on testing and conducting experiments for the randomized SVD and ANN algorithms. This also includes documentation and packaging efforts. No problems are currently anticipated.**

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **12** | |

## 1 Abstract

In the third quarter of the work effort, we focused on the research and design of the randomized Approximate Nearest Neighbors (ANN) algorithm. This randomized variant of the ANN algorithm has theoretically proven improvements in the number of data dimensions that it can handle over existing algorithms and meets the theoretical lower bounds for computational complexity. Algorithm designs for computing the Randomized Approximate Nearest Neighbors (ANN) using randomized Fast Fourier Transform projections were completed. Fortran 95 interface for reusable randomized ANN routine has been defined and implemented. The randomized ANN implementation uses BLAS libraries via standardized interfaces to make optimal use of hardware resources (e.g., multiple cores, CPU cache) in addition to using the OpenMP standard (for parallel execution of code). Use of these standards enables the code to be built flexibly in a number of ways on various target platforms. Preliminary testing of the software is complete. Additional updating, fine tuning will be based on results from various experiments that will be conducted in the upcoming quarter.

The project is currently on track – in the upcoming quarter, we will focus on testing and conducting experiments for the randomized SVD and ANN algorithms. This also includes documentation and packaging efforts. No problems are currently anticipated.

# Table of Contents

# 2 Summary

The focus for this quarter was on the research and development of the randomized Approximate Nearest Neighbors algorithm using random Fast Fourier Transform projections. Fortran 95 interface for reusable randomized ANN routine has been defined and implemented.

The implementation uses the Basic Linear Algebra Subprograms (BLAS) standard apart from the OpenMP standard (for parallel execution on multi-core/multiple CPUs). Preliminary tests were satisfactory – especially those involving real-world data in text retrieval applications. Further updates and fine-tuning will be based on testing and experiments conducted in the upcoming quarter.

The project is currently on track – we will focus on testing and conducting experiments for the randomized SVD and ANN algorithms in the next quarter. No problems are currently anticipated.

# 3 Introduction

The primary project effort over the last quarter focused on the design and development of fast, scalable Approximate Nearest Neighbor (ANN) algorithm using randomized Fast Fourier Transform projections (see [1]). The algorithm has been implemented as a reusable routine in Fortran 95 with well-defined interfaces. The implementation uses the standard Basic Linear Algebra Subroutines (BLAS) [2] interface and the OpenMP API [7]. This provides flexibility in terms of compiling the software on a variety of target platforms; exploiting availability of optimized BLAS libraries (see [3][4][5][6]) and availability of multiple cores/CPUs.

Preliminary tests have been conducted – additional testing and experimentation will be carried out in the upcoming quarter for both the randomized SVD and ANN algorithms resulting in possible improvement and fine-tuning of the software.

# 4 Methods, Assumptions and Procedures

## 4.1 Randomized Approximate Nearest Neighbor Algorithm

Our implementation of the randomized ANN involves the use of randomized projections using the Fast Fourier Transform (see [1] for details).

The algorithm may be broadly decomposed into three steps – a) a randomization step where the points in the input data set are subject to a random projection (to a lower dimensional space), b) construction of a tree data structure to partition the points using the medians of the first few coordinates, and finally c) refining the bins using supercharging for subsequent queries.

Note: Assume the input data set is of size $n$ and contains points of dimension $d$. For all three steps, BLAS operations are used wherever appropriate.

### 4.1.1 Randomization Step

In this step, we apply a pseudo-random orthogonal linear transformation matrix to each data point. This step is easily parallelizable since the random projection may be applied to each point independently. We choose a pseudo-random transformation matrix based on the Fast Fourier Transforms (FFT) which is less expensive to compute than say, drawing random numbers from a Gaussian distribution.

### 4.1.2 Tree Construction Step

Here, we construct a tree data structure of depth . Set the bin at level 0 to all the data points. For a given level $l$, use the median of the $l$-th coordinate to partition each bin at the previous level into two bins (based on whether the $l$-th coordinate of the selected data point is less-that-or-equal-to or greater-than the median) at the current level. A label comprising „L‟ or „G‟ may be then assigned to each leaf bin indicating the less-that/greater-than decision at each level of the tree (e.g., "LLGL").

For each leaf bin (at level $L$), compute the $k$ nearest neighbors by searching over the adjacent bins. Adjacent bins are defined as those with labels that are exactly unit edit distance away.

### 4.1.3 Supercharging Step

For each point, consider the set of all its current $k$-ANN and the $k$-ANNs for each of them. Then, reconstruct the $k$-ANN using this bigger set.

### 4.1.4 Query

The data structure constructed in the previous steps is used to determine the $k$-ANN for any given data point.

If the data point is member of the original data set, then the $k$-ANN set for that data point is simply retrieved from the data structure and returned.

If the data point is not a member of the original data set, then the random projection and the partitioning tree is used to find the appropriate leaf bin for the new data point. A *k*-NN search is performed on the smaller set of all the points in the bin as well as all their current *k*-ANNs to determine the *k*-ANN result set.

In practice, multiple rounds of the algorithm along with one or more supercharging iterations are found to provide the best results.

## 4.2 Deliverables / Milestones

| Date | Deliverables / Milestones | Status |
|---|---|---|
| Oct 2010 | Progress report for period 1, 1st quarter | ✓ |
| Jan 2011 | Progress report for period 1, 2nd quarter / complete randomized matrix decompositions task | ✓ |
| Apr 2011 | Progress report for period 1, 3rd quarter / complete approximate nearest neighbors task | ✓ |
| Jul 2011 | Progress report for period 1, 4th quarter / complete experiments – part 1 | |
| Oct 2011 | Progress report for period 2, 1st quarter | |
| Jan 2012 | Progress report for period 2, 2nd quarter / complete multiscale SVD task | |
| Apr 2012 | Progress report for period 2, 3rd quarter | |
| Jul 2012 | Progress report for period 2, 4th quarter / complete experiments – part 2 | |
| Oct 2012 | Progress report for period 3, 1st quarter | |
| Jan 2013 | Progress report for period 3, 2nd quarter / complete multiscale Heat Kernel task | |
| Apr 2013 | Progress report for period 3, 3rd quarter | |
| Jul 2013 | Final project report + software + documentation on CDROM / complete experiments – part 3 | |

## 5 Results and Discussion

We present some results from our preliminary testing of the randomized Approximate Nearest Neighbors algorithm to highlight the

a) accuracy of the randomized ANN, and
b) applicability to real-world text IR problems.

### 5.1 Test Setup

A data set containing 400 dimensional points of size 35,270 was constructed from analyzing text documents from eight different sources. Each point corresponds to a body of text (a.k.a. document). A standard task in Information Retrieval (IR) is to find similar documents given a query document. This requires computing the nearest neighbors subject to some threshold on the distance. In this test, we apply the randomized ANN algorithm to this problem. The number of nearest neighbors *k* was set to 30 with a distance threshold of 0.8. Further, each document vector has unit norm.

All tests were carried out on a machine with a Intel Core Duo E6550 2.33GHz 2-core CPU (total of 2 cores) with a total of 4 GB main memory. The OS was Ubuntu 10.10 64-bit SMP.

### 5.2 Test Results

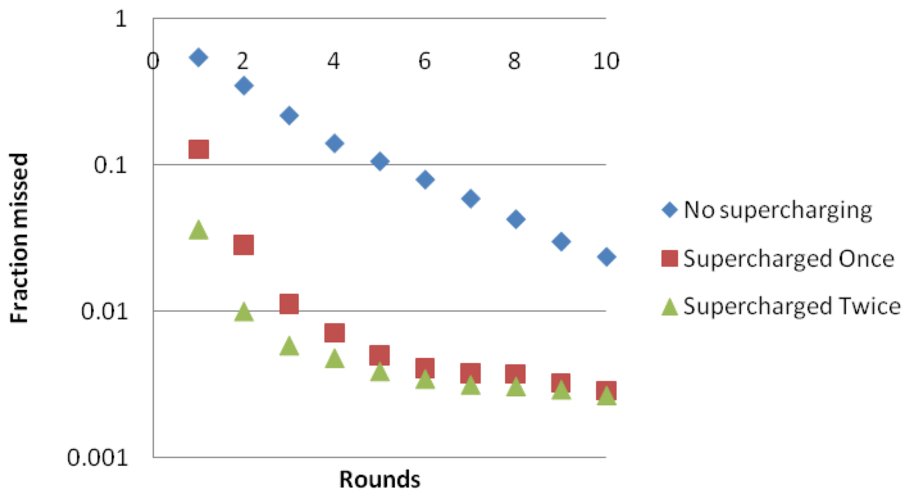

**Figure 1: Fraction missed for 30-ANN with distance threshold of 0.8**

Figure 1 shows the fraction missed (the baseline was the exact 30-NN for all the data points) in computing the 30-ANN using our algorithm for all the points in the data set. It highlights the performance boost obtained using supercharging. The error is about 0.5% which is within acceptable limits. Figure 2 and shows the corresponding run times. The time for computing the exact 30-NN for all the data points was recorded to be 4701 seconds.

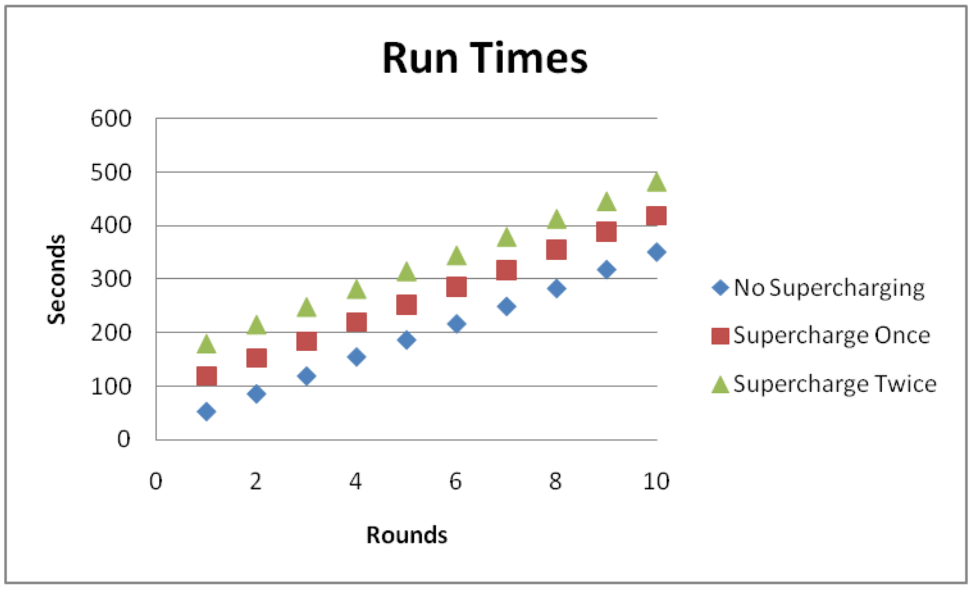

**Figure 2: Run times for 30-ANN with distance threshold of 0.8**

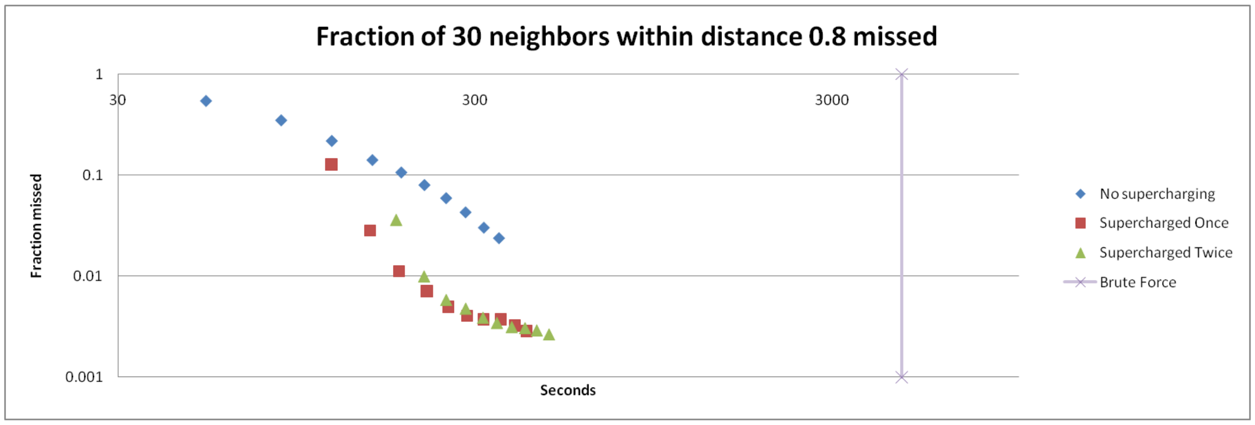

**Figure 3: Run times for 30-ANN with distance threshold of 0.8**

Figure 4 shows the effects of multiple rounds on the quality of ANN results with different distance thresholds (cut-offs). Overall, the algorithm shows stability with approximately 0.5% error for thresholds of interest with a performance gain of 90-94% over the exact *k*-NN algorithm. As expected, the error grows as the distance threshold is significantly increased.

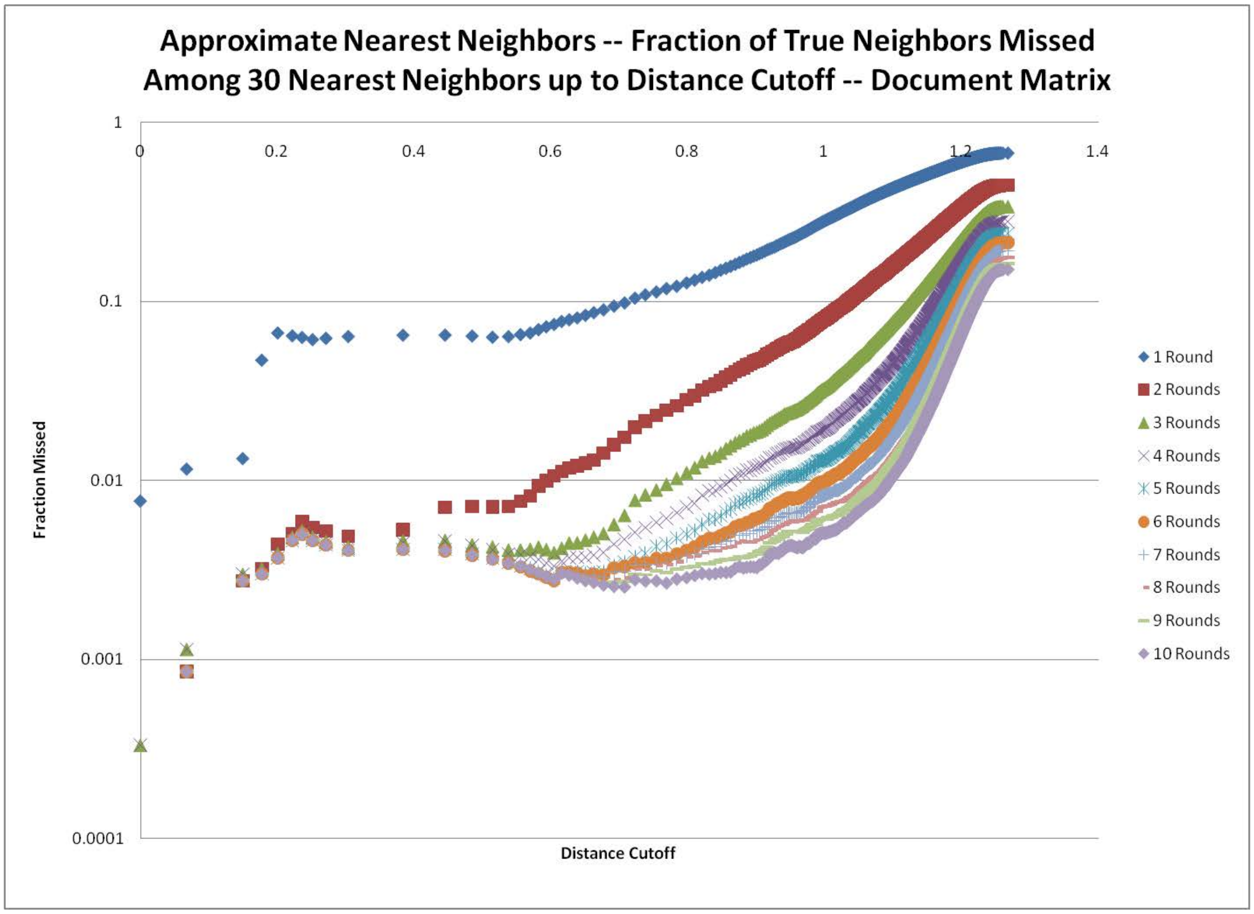

**Figure 4: Effect of number of rounds on fraction missed and distance cutoff**

## 6 Conclusions

The project is on track with the completion of design and implementation of the second algorithm (Randomized Approximate Nearest Neighbors). We will continue with further testing and application of the randomized SVD and ANN algorithms to a variety of real-world data sets in the next quarter.

No problems are currently anticipated.

## 7 References

[1] P. Jones, A. Osipov, V. Rokhlin, *A Randomized Approximate Nearest Neighbors Algorithm*, Research Report YALEU/DCS/RR-1434, Yale University, September 14, 2010
[2] Basic Linear Algebra Subprograms (BLAS), URL:http://www.netlib.org/blas
[3] GotoBLAS2, URL:http://www.tacc.utexas.edu/tacc-projects/gotoblas2/
[4] Automatically Tuned Linear Atlas Subroutines (ATLAS), URL:http://www.netlib.org/atlas
[5] Intel Math Kernel Library, URL:http://software.intel.com/en-us/articles/intel-mkl
[6] AMD Core Math Library (ACML), URL:http://developer.amd.com/cpu/Libraries/acml/Pages/default.aspx
[7] The OpenMP API Specification for Parallel Programming, URL:http://www.openmp.org