K-d Tree Search Algorithms for a Nearest Neighbor Gaussian Process Model

Alexander McKim[1], Andrew Finley[2]

[1]Clemson University, Clemson, SC, 29631

[2]MSU Department of Forestry, East Lansing, MI, 48824

1. Introduction

- Various statistical models for spatial data rely on some form of a nearest neighbor calculation among observed spatial locations
- A brute force solution to a nearest neighbor calculation is easy to implement, but is computationally impractical for large data sets
- Our focus is on efficient implementation of a statistical model called the Nearest Neighbor Gaussian Process (NNGP; Datta et al. 2016; Finley et al. 2017) that involves nearest neighbor searches for massive spatial data sets
- These implementations involve k-d trees, a structure commonly used to make nearest neighbor searches more efficient

2. Motivation

Our interest is in improving the constrained nearest neighbor search needed to implement a NNGP used in space and space-time regression models.

The NNGP provides a close approximation to a full Gaussian process (GP), which provides some ideal statistical interpolation properties but is computationally infeasible for large datasets.

Importantly the NNGP is constructed using a sparse representation of the GP's precision matrix among observed locations, and hence process parameters can be estimated in a fraction of the time needed to estimate the GP's process parameters.

NNGP is referred to as a sparsity-inducing Gaussian Process. This sparsity must be introduced in a specific manner to maintain a valid joint distribution, notably, an observation's neighbors must meet an ordering constraint; hence the need for the specialized search algorithms developed here.

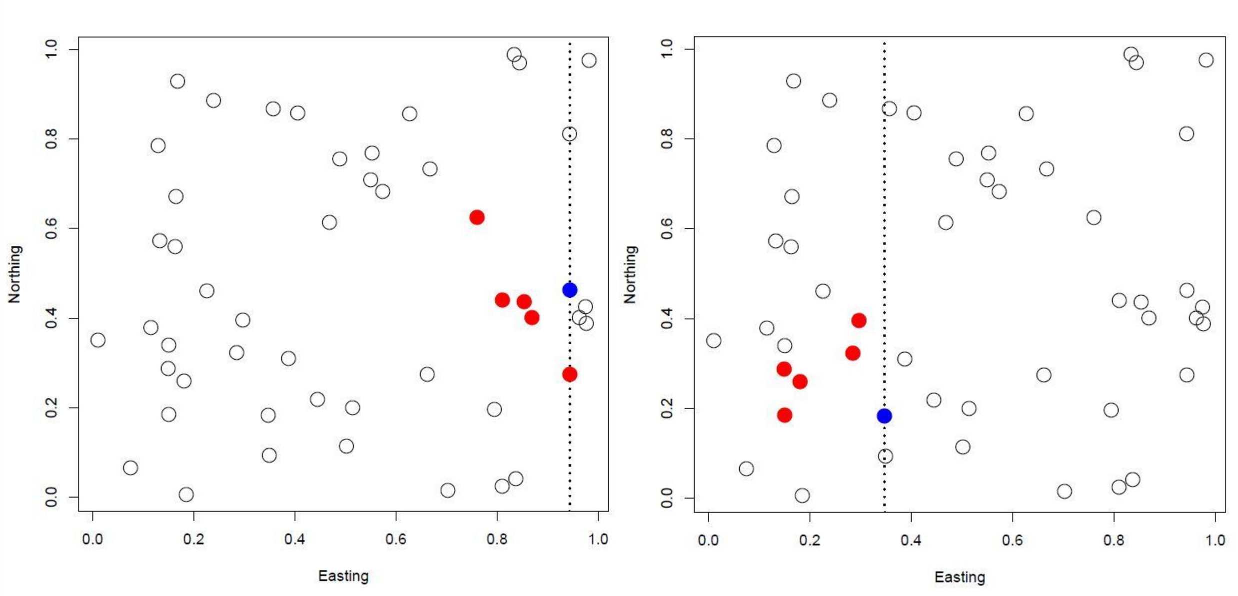


Figure 1: Nearest neighbors under x-axis ordering constraint. Only locations to the left of the given location are candidate neighbors. Left figure shows five neighbors (red point) for location 18 (blue point) among all observations (open circles). Right figure same set up but for location 45 (blue point).

3. Algorithm Design/Implementation

Proposed structures used in algorithms similar to classic k-d trees.

Four algorithms/structures implemented:

- 1. Serial incremental construction of classic k-d tree
- 2. Parallel search using an unbalanced tree
- 3. Parallel search with self-balancing technique
- 4. Parallel search with self-balancing and clustered observations into bucket

- Results are shown for first three techniques performed on randomly generated data sets of various sizes in different parallel settings
- All algorithms implemented in C++, with parallelization via OpenMP

4. Results

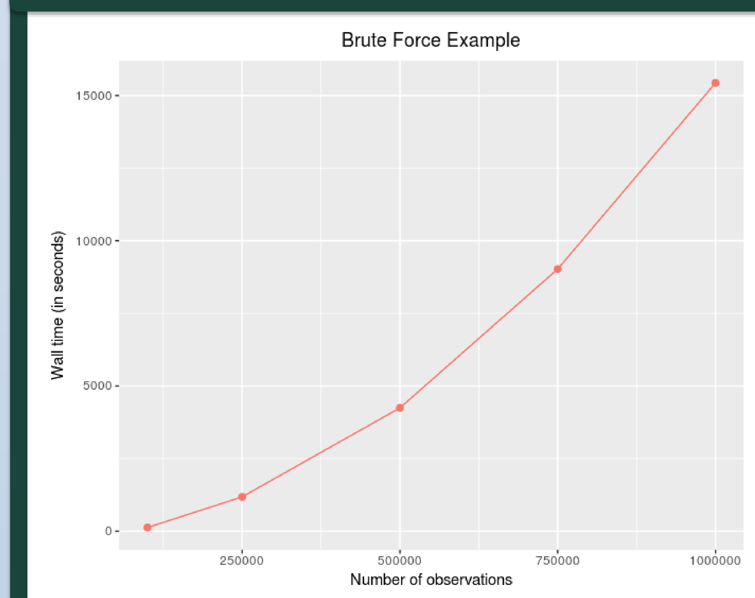


Figure 2: Brute force approach, 1 cpu and 15 neighbors.

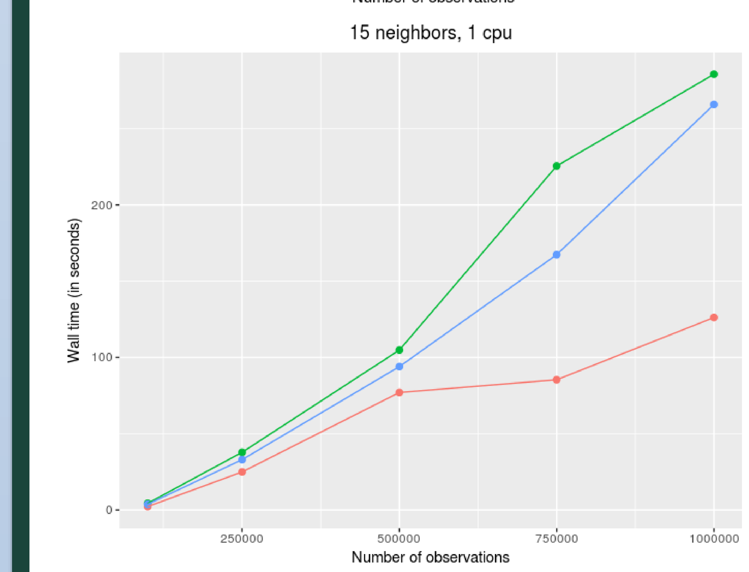


Figure 3: 1 cpu and 15 neighbors.

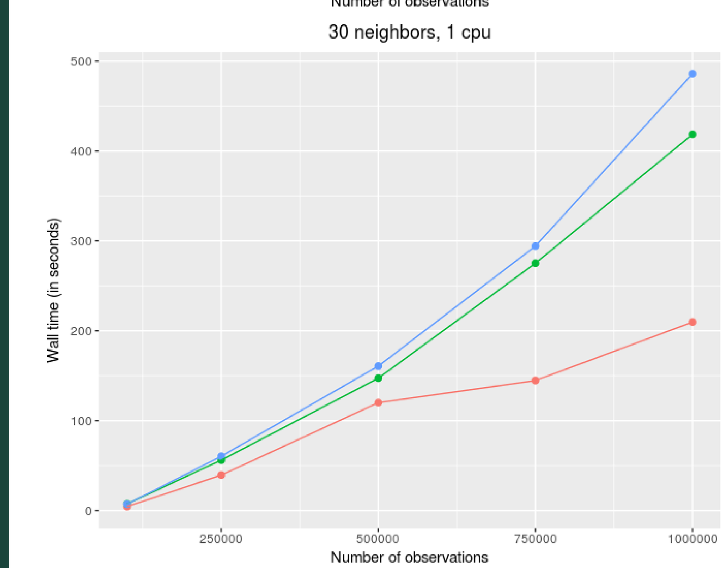


Figure 4: 1 cpu and 30 neighbors.

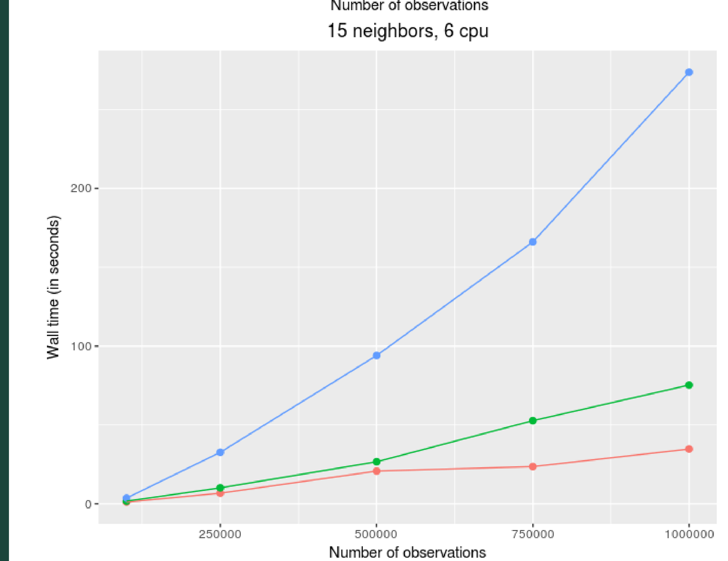


Figure 5: 6 cpu and 15 neighbors.

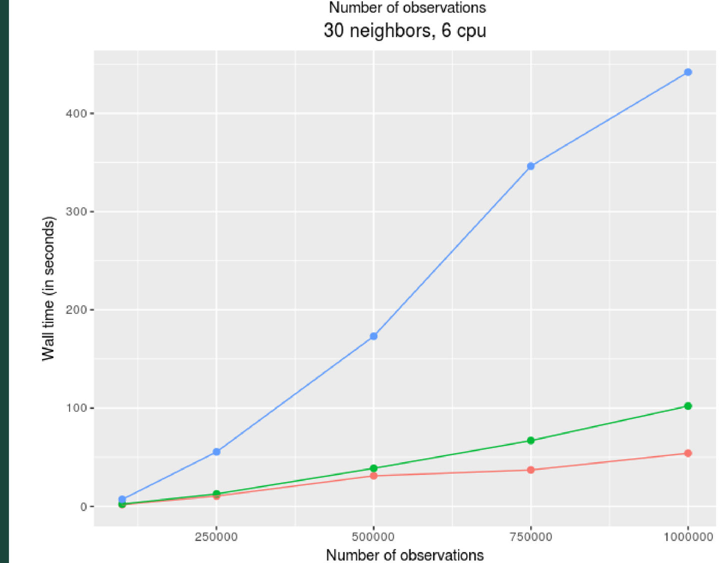
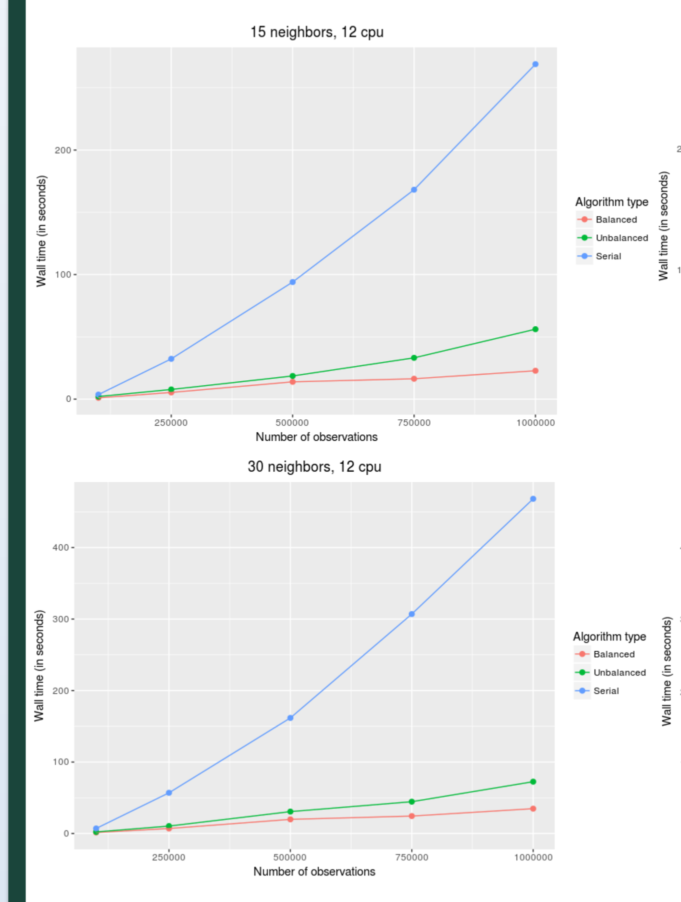


Figure 6: 6 cpus and 30 neighbors.

4. Results



Figures 7,8: 12 cpus, for 15 and 30 neighbors respectively.

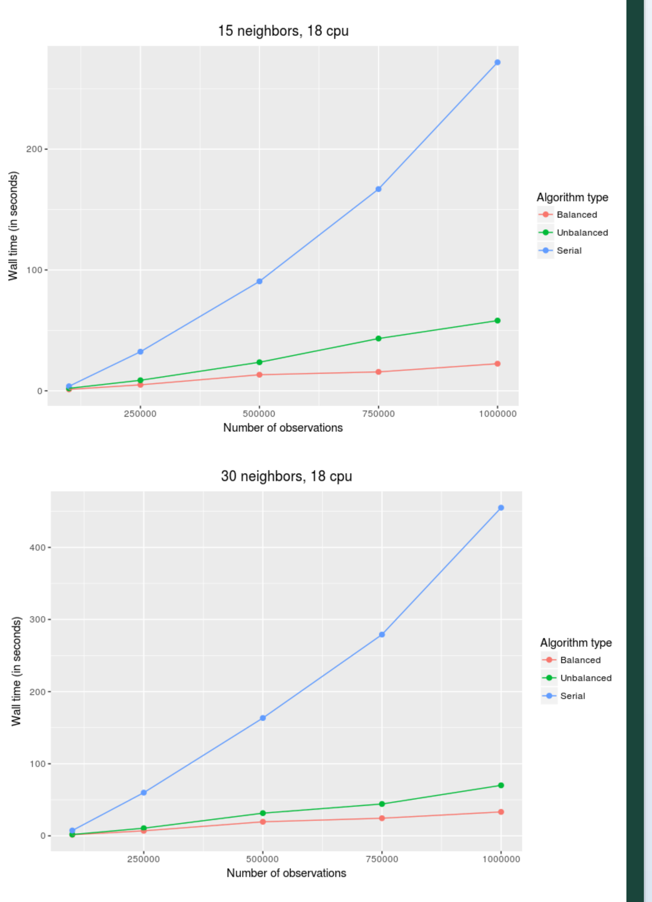


Figure 9,10: 18 cpus, for 15 and 30 neighbors respectively.

- Overall results show that the balanced approach is the fastest
- Serial can be faster than the unbalanced method if there is only 1 core

5 Discussion

- First three algorithms tested extensively and are significantly faster than brute force
- Fourth algorithm still has minor bugs, but clustering of "similar" points is correct
- Self-balancing technique speeds up significantly, but randomization can cause slower results (Figure 11)
- Currently, the unbalanced algorithm has been released in the spNNGP R package
- https://cran.rproject.org/web/packages/spNNGP/index.html

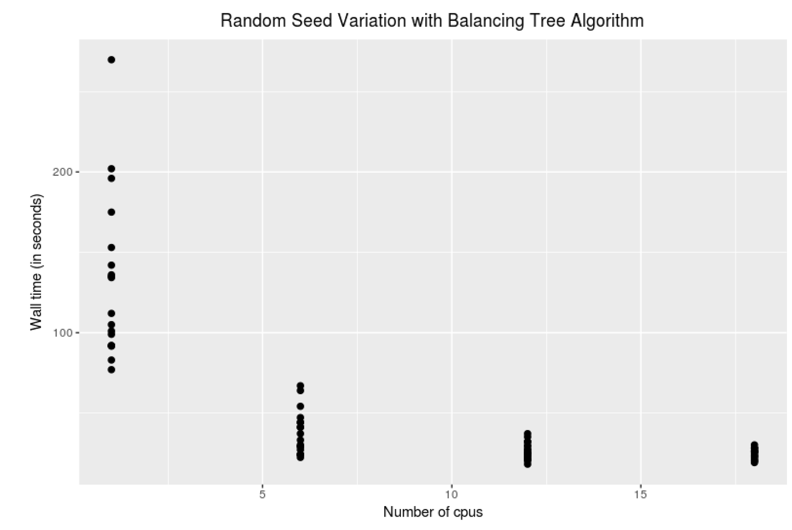


Figure 11: Search time variation by cpu. Random balancing decisions leads to greater variability, but overall fast runtime

6. Future Work

- Implement Balancing Algorithm with no randomness
- Investigate more pruning possibilities on the search
- Implement tree to work with more dimensions, particularly three dimensions
- Develop a faster way to cluster points in a bucket, and find an optimal bucket size for clustering, and for parallel computations

7. Acknowledgements and References

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. Journal of the American Statistical Association, 111:800-812.

Finley, A.O., A. Datta, B.C. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee (2017) Computing Bayesian Nearest-Neighbor Gaussian Process Models for Massive Spatial Data Sets