PSCAN: Parallel, density-based clustering of protein sequences

Joshua Brulé

Introduction

Traditional clustering algorithms typically scale poorly in protein clustering due to the expense of calculating precise similarity scores and the volume of data. Existing techniques, such as UCLUST or CD-HIT [1][2] use a greedy algorithm that identifies a representative sequence for each cluster, and assigns a new sequence to a cluster if it is sufficiently similar to the representative or assigns the new sequence as the representative sequence for a new cluster.

PSCAN is designed to explore the tractability of using a parallel implementation of a higher-quality (and more time/space expensive) clustering algorithm on a representative sample of the sequences to be clustered. After producing high-quality clusters, it is possible to calculate the most representative (medoid) sequence for each cluster, and perform a simple nearest-neighbor clustering to classify the rest of the sequences.

Methods

PSCAN is a parallel implementation of the density-based clustering algorithm DBSCAN*, using a global sequence alignment score as the distance measure.

DBSCAN [3] clusters points based on density reachability - a point is directly density reachable if it is within *epsilon* distance of at least *min-pts* points. A point p is density reachable to q if there is some sequence of points p_1, p_2, ... p_n=q where each point in the sequence is directly density reachable to the next. A cluster is the collection of all points that are all mutually density reachable. [NOTE 1]

DBSCAN requires O(n) region queries - for each point, the algorithm has to determine the collection of points within *epsilon* distance, for a total runtime of $O(n^2)$, executed serially. However, region queries are very amenable to parallelization. PSCAN uses fork/join parallelism to execute region queries, partitioning the collection of sequences into approximately *partition-size* sized subcollections, computing and filtering the distance scores in each partition and recursively combining the partitions into the final result. [NOTE 2] In addition, PSCAN optionally memoizes distance computations, CASing (compare-and-set) the distance value into the cache.

Note that the distance function does not have to satisfy all of the metric axioms - it only has to be symmetric. PSCAN uses the Needleman-Wunsch algorithm with affine gap penalty to calculate a similarity score for each pair of protein sequences, with a higher score corresponding to more closely related proteins. Under this definition of distance, "within epsilon distance" refers to two proteins having a similarity score of at least *epsilon*.

As described, PSCAN has a large number of free parameters. Defaults based on previous work were used wherever possible: *min-pts* = 4 (recommended by the original DBSCAN authors) [4]; substitution matrix = BLOSUM62, gap existence penalty = 11, gap extension penalty = 1 (NCBI BLAST defaults).

Test Data and Cluster Validation

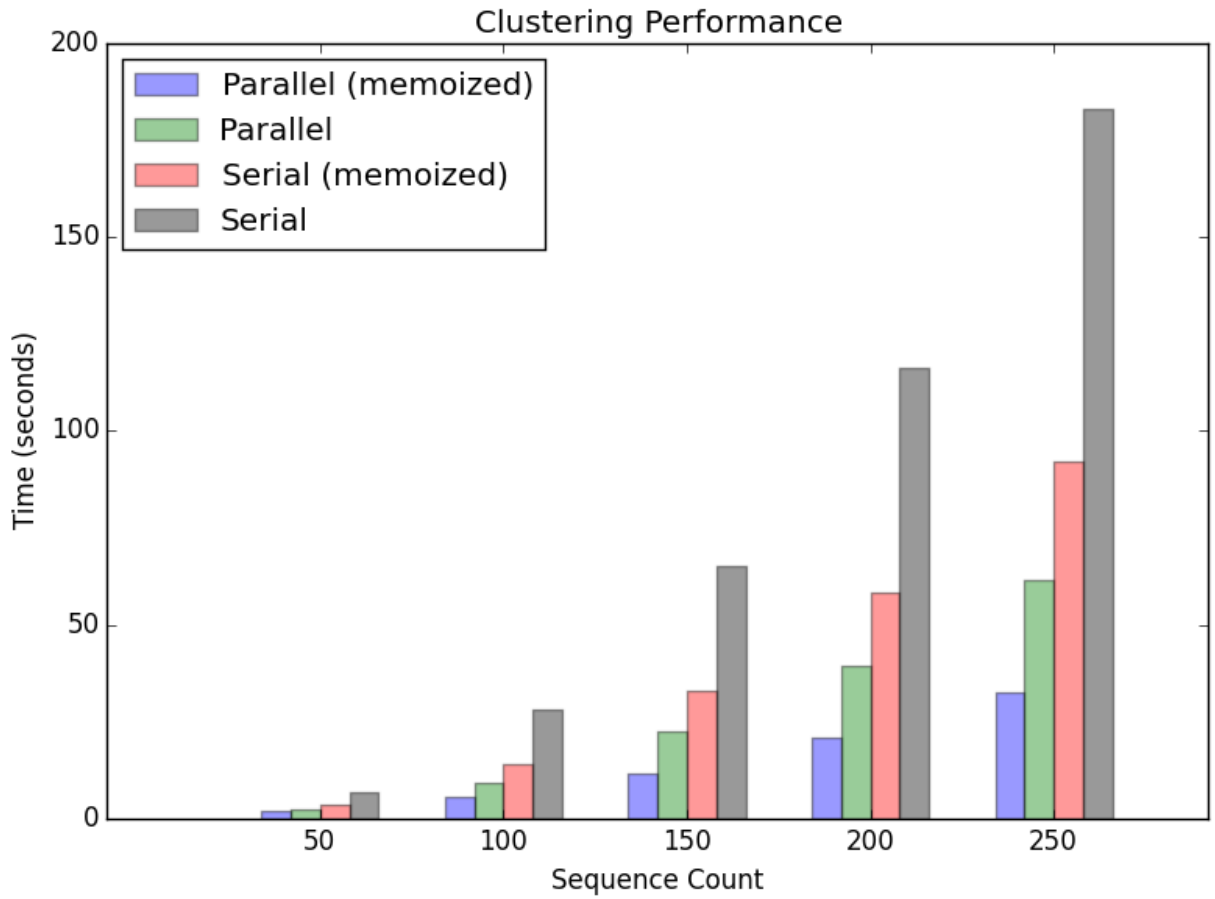
PSCAN was tested using data from the UniProt Reference Clusters [5], using 25 UniRef90 clusters of 10 (unique) sequences per cluster with sequence length between 247-257. [NOTE 3]

An appropriate *epsilon* was determined empirically. PSCAN began incorrectly classifying UniRef90_I0C2H0 and UniRef90_P0A0Q0 as the same cluster at *epsilon* ≈ 200 or less. PSCAN began to fail to cluster some sequences at *epsilon* ≈ 1000 or greater. Between these values, PSCAN correctly assigned all 250 proteins to their original UniRef90 clusters. Lacking any additional insight as to the interpretation of epsilon, the rest of the clustering validation was run with *epsilon* = 250 (approximately the average sequence length).

Partition sizes of 4, 8, 16, 32 and 64 were all tested, with no significant effect on runtime.

Using the medoid of each cluster to perform nearest-neighbor clustering, the entirety of the UniRef90_Q0A457 cluster (3,175 proteins) was classified correctly.

Performance



(On a 4 core Intel i7-3610 @2.30 GHz; 8GB RAM)

Discussion

Memoization consistently provided an approximately 2x speedup over a non-memoized DBSCAN. However, this comes at the expense of $O(n^2)$ additional memory. Parallelization on a 4 core machine provided approximately 3x speedup over a serial DBSCAN. As written, PSCAN should generalize to an arbitrary number of process *with shared main memory*. However this has not been tested.

These results suggest a general strategy of favoring parallelism over memoization. Nearest-neighbor classifying runs in linear time and constant space and non-memoized DBSCAN runs in linear space. A scalable strategy to cluster a large number of proteins would be to run DBSCAN on a sample of the protein sequences that can fit into main memory, followed by nearest-neighbor clustering on the remaining dataset, using the medoids of the DBSCAN clusters as representatives.

However, exact global alignment is far more precise (and disproportionately more expensive) than necessary, seeing as how DBSCAN produced the original UniProt clusters for a very wide range of *epsilon*. For this reason, the (theoretically) higher-quality clustering provided by DBSCAN seems unlikely to provide significant improvements in the cluster quality.

However, should a well-understood biologically significant similarity score for proteins be discovered, PSCAN's higher precision may warrant the additional run time. Since, as the number of processors available increases, PSCAN gains a near-linear speedup, density-based protein clustering may be quite valuable in the future.

Footnotes

[NOTE 1] In the original DBSCAN algorithm, density-reachability is not necessarily symmetric. It is possible for there to be "border" points, which are density reachable from some "core" point, but do not have *min_pts* neighbours within *epsilon* distance. This can result in the same border point being assigned to different clusters, depending on the order in which points are visited by the algorithm. PSCAN implements the DBSCAN* variation of DBSCAN, which guarantees a deterministic classification by treating border points as noise points.

[NOTE 2] PSCAN uses clojure.core.reducers for parallelism, built on Java's Fork/Join framework. Idle threads can "work-steal" tasks from other threads, minimizing the number and time of idle threads. [6] Under theoretically ideal conditions, with a sufficient number of processors available, a region query could be executed in O(log(n)) time.

[NOTE 3] The median sequence lengths for Archaea and Bacterial proteins, respectively. [7]

References

[1] Edgar, Robert C. Search and clustering orders of magnitude faster than BLAST
http://bioinformatics.oxfordjournals.org/content/26/19/2460

[2] Li, Weizhong, et al. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences
http://bioinformatics.oxfordjournals.org/content/22/13/1658.full

[3] Campello, Ricardo J. G. B., et al. Density-Based Clustering Based on Hierarchical Density Estimates
http://link.springer.com/chapter/10.1007%2F978-3-642-37456-2_14

[4] Ester, Martin, et al. A density-based algorithm for discovering clusters in large spatial databases with noise
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.71.1980

[5] The UniProt Reference Clusters (UniRef)
http://www.uniprot.org/uniref/?query=length%3A%5B247+TO+257%5D+AND+identity%3A0.9&sort=count

[6] Lea, Doug. A Java Fork/Join Framework
http://gee.cs.oswego.edu/dl/papers/fj.pdf

[7] Karlin, Samuel. Protein length in eukaryotic and prokaryotic proteomes
http://nar.oxfordjournals.org/content/33/10/3390.full