ENTHOUGHT

SEARCHING EFFICIENTLY THROUGH (GENOMIC) SEQUENCES WITH VANTAGE POINT TREES.

J. Vankerschaver; R. Kern; S.J. Kern; P. Zahemszky; C. Mueller; R. Cardwell

PRESENTED AT THE 69TH ANNUAL MEETING OF THE AMERICAN SOCIETY OF HUMAN GENETICS, OCTOBER 16, 2019, HOUSTON, TEXAS

Introduction

The current generation of sequencers produce massive quantities of genomic sequence data, and in order to gain new biological insights from this data effectively, researchers must be able to map these sequences to databases of known sequences quickly and accurately. Examples include determining the function of an unknown gene in a gene catalog, or assigning taxonomy information to a bacterial amplicon by finding the closest matching sequence in a taxonomic database such as RefSeq.

In terms of efficiency, standard tools such as BLAST (Basic Local Alignment Search Tool) provide fast lookups at the expense of some accuracy, whereas more accurate assignments can be obtained with tools such as glsearch or ggsearch (part of the fasta toolkit), performing either a local-to-local or global-to-global alignment. Ad-hoc computational approaches, such as organizing the target sequences in a k-dimensional tree, are able to increase throughput but sacrifice accuracy by imposing restrictions that are not appropriate for genomic sequences.

Here, we show that the best of both worlds (a lookup method that is accurate and fast) can be achieved by organizing the target sequences instead in a vantage-point tree data structure. Vantage point trees only require a notion of distance between sequences, which is naturally provided by global-to-global alignment. In practical applications, they deliver an order-of-magnitude improvement in lookup speed while maintaining the same accuracy as alignment-based methods.

Sequence alignment methods	Computational tool	Accuracy	Speed
Exact alignment + linear scan across database	glsearch / ggsearch	+++	-
Inexact alignment	BLAST	++	+
Exact alignment + K-dimensional trees	(non-ideal for bioinformatics)	+	++
Exact alignment + vantage-point trees	**vpsearch**	**+++**	**++**

BACKGROUND: VANTAGE POINT TREES

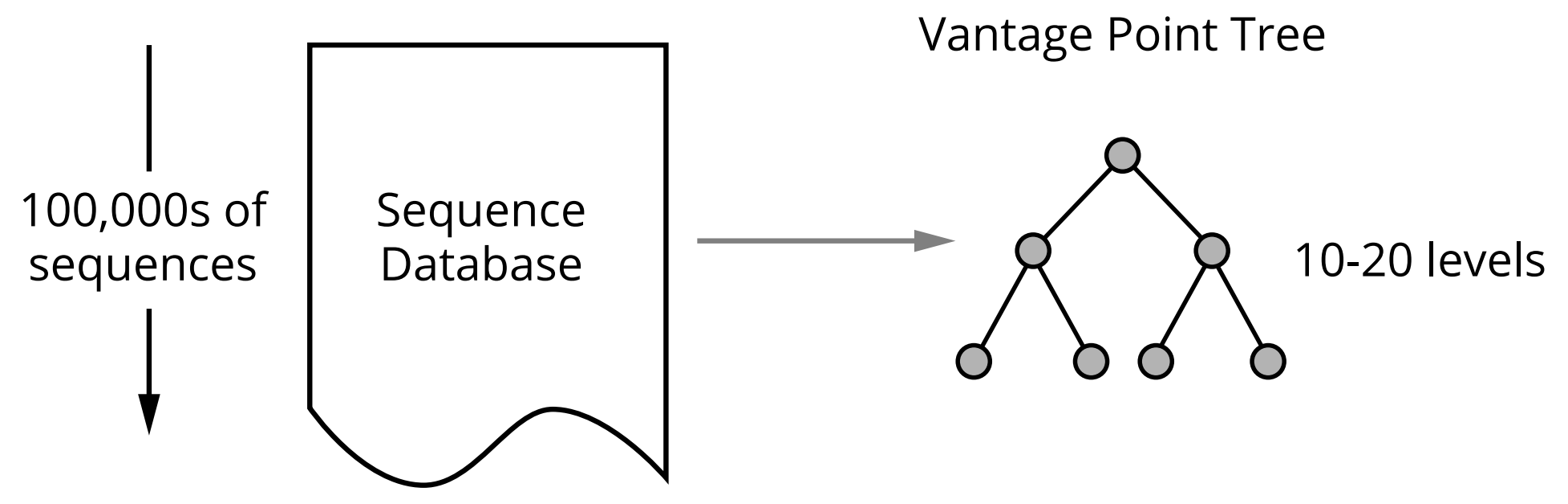
Vantage point trees were introduced independently by Uhlman (1991) and Yanilos (1993) as a tool for information retrieval. A distance function (or “metric”), encoding how (dis)similar the objects under consideration are, is the only requirement when using a vantage point tree. Vantage point trees have been used to find similar patches or patterns in images or to find similar melodies in music.

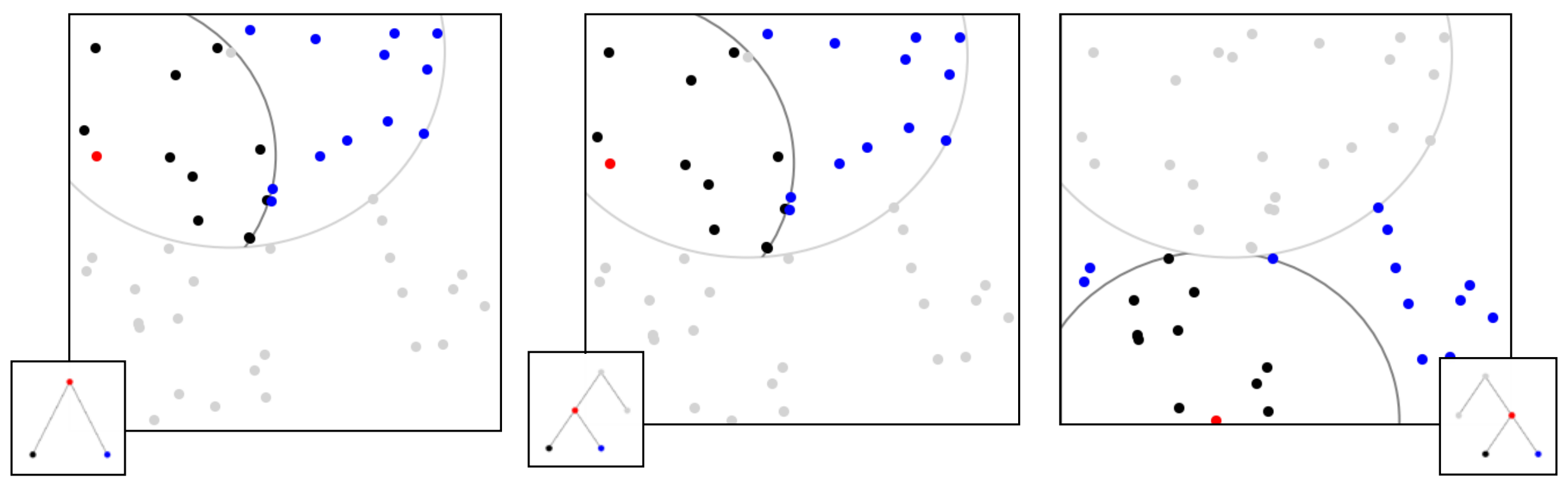
Methods: Building a vantage point tree

Vantage point trees represent a large database of genomic sequences (with thousands to millions of sequences) as a tree-like structure.

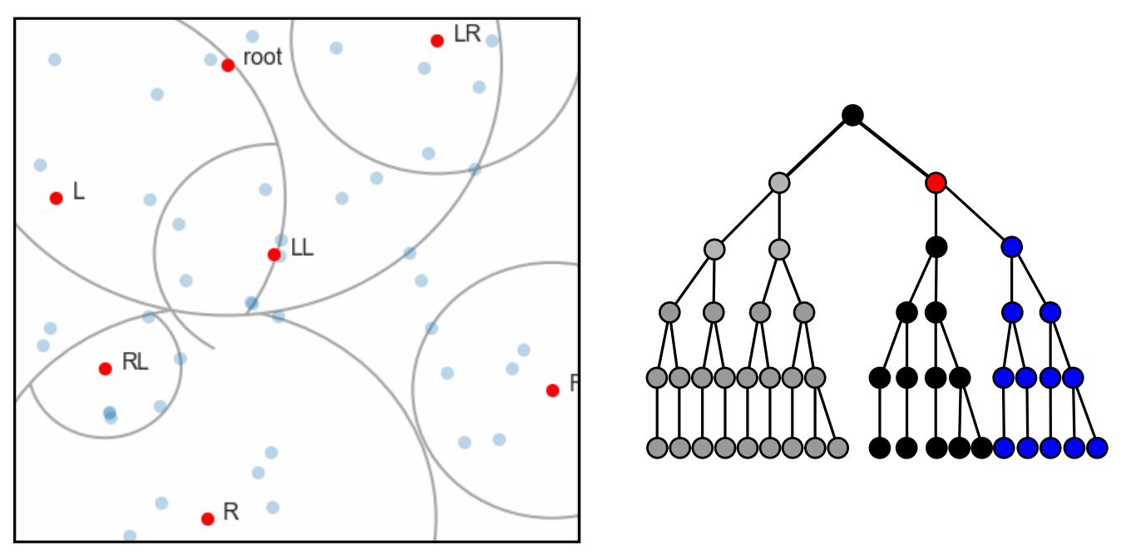
Building a vantage point tree is a one-time operation that takes about 10s for 100,0000 sequences. Once the tree is built, it can be re-used over and over again for different queries.

Each level in a vantage point tree is built by taking an arbitrary sequence and subdividing the remaining sequences into the 50% most similar and 50% most dissimilar, which go into the left and right subtrees, respectively.
.





1. Take a random “vantage point” (•)
2. Measure the alignment distance from each sequence to the vantage point
3. Divide into 50% closest sequences (•), 50% furthest sequences (•)
4. Apply steps 1-3 to the left and right subtrees
5. Recursively apply this process until no more points are left, resulting in a “tree”



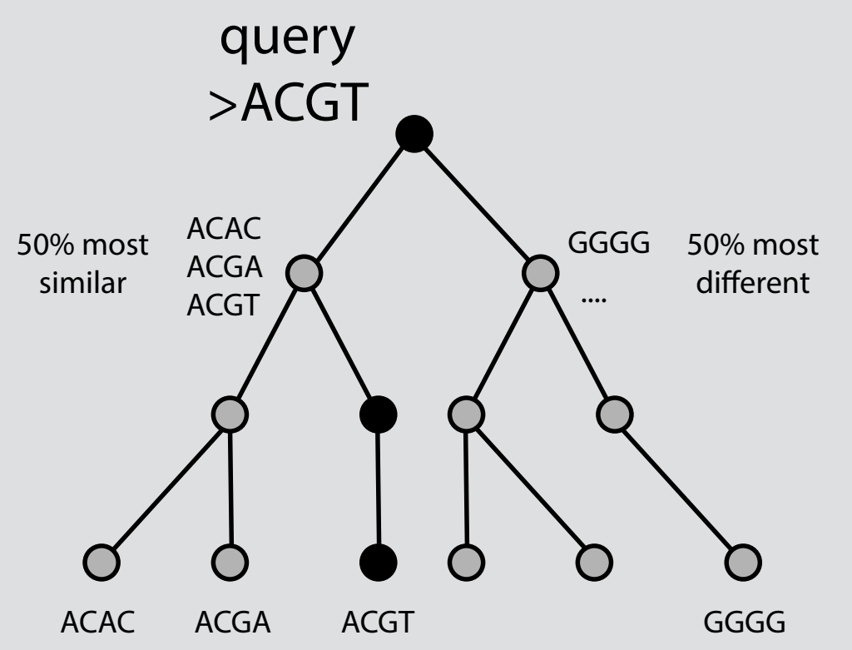
QUERYING USING A VANTAGE POINT TREE

Once a vantage point tree is built, queries can be looked up using a handful of comparisons:

- Compare the query with the root of the tree
- Take the left subtree if the query is similar, the subtree path if the query is dissimilar (occasionally both subtrees need to be explored)
- Repeat until only one sequence is left. This is the target.

For a database of 100,000 sequences, the vantage point tree is about 15 levels deep. This implies that a lookup needs about 15 comparisons to find the target sequence, rather than potentially thousands, as with a linear scan.

Our reference implementation takes up less than 500 lines of Python/Cython code and emits data in the standard Blast 8-column format, so that it can be used as a drop-in replacement for most standard lookup tools.



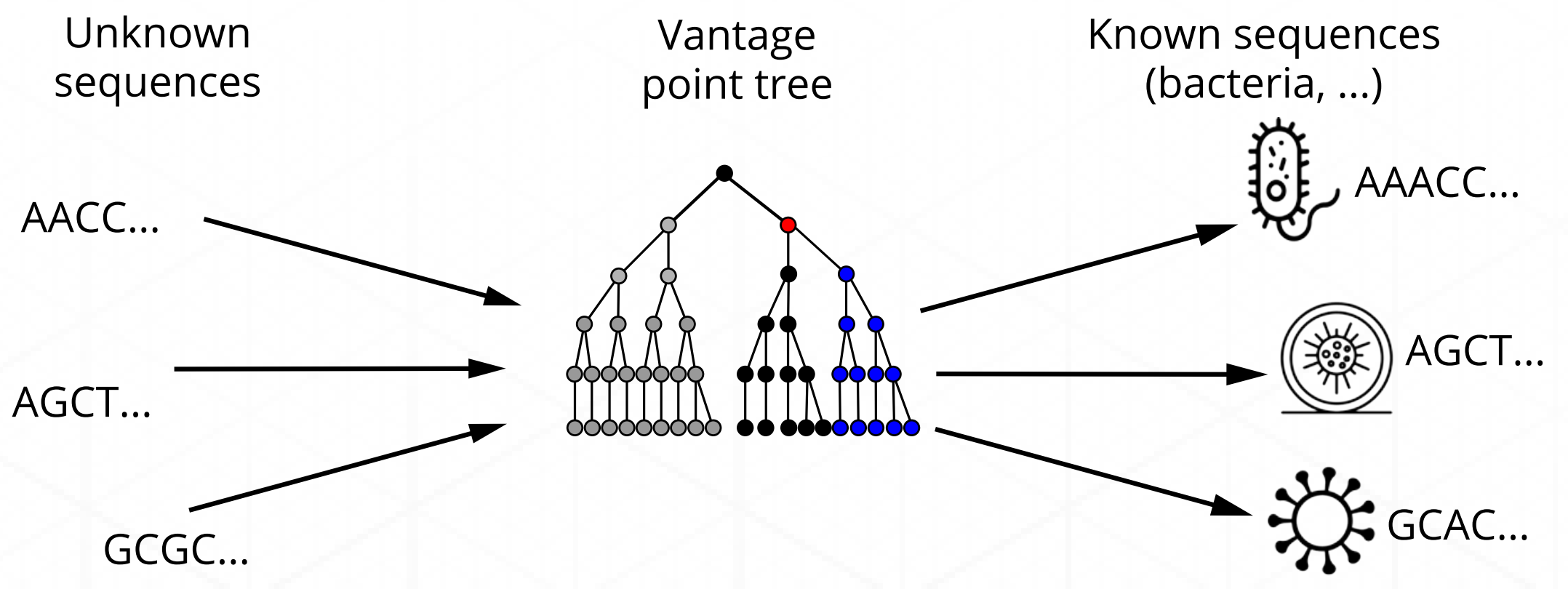
Results: Taxonomic assignments using vpsearch

Using a dataset comprised of the bacterial 16S rRNA gene, sequence analysis was performed and results compared. The paired-end reads from a single library of unknown composition targeting the v1-v2 hypervariable region were generated on an Illumina MiSeq platform with approximately 10,000 total reads and an average read length of 356 bp. Sequences were first prepared by removing eukaryotic (non-bacterial) and low-quality sequences (average quality less than 25). The remaining sequences were clustered into 129 operational taxonomic units (OTUs) representing putative species present in the sample, with the open source vsearch tool at 97% similarity.

Taxonomic assignment of the sequences was performed using either the vpsearch algorithm, our vantage point tree method, or ggsearch36 using a combined Ribosomal Database Project (RDP) and CORE (Ohio State University, oral microbiome) database, containing 117,010 labeled v1-v2 HVR sequences.

Of the 129 OTUs, vpsearch and ggsearch (part of the fasta-36.3.8.4 package) suggest 123 identical OTUs at the species level (>95% concordant). The six (6) OTU differences observed are due to how ambiguous nucleotides are scored by both algorithms. Adjustment of the scoring matrix is likely to alleviate differences. (Figure 1).

Additionally, the lookup time (in seconds) required to query a database was modeled using vpsearch and ggsearch. On a realistic database of 100,000 annotated sequences, unknown sequence lookup with the vantage point method was 0.1 second, while the exact alignment (ggsearch) method required approximately 1 second, a 10X improvement in speed. On the whole, vpsearch scales logarithmically with the size of the database (doubling the size of the database increases the query time by a constant amount) while ggsearch scales linearly (query time is proportional to the size of the database). (Figure 2).



Unknown genomic sequences are processed through vantage point tree method to identify bacterial taxonomy

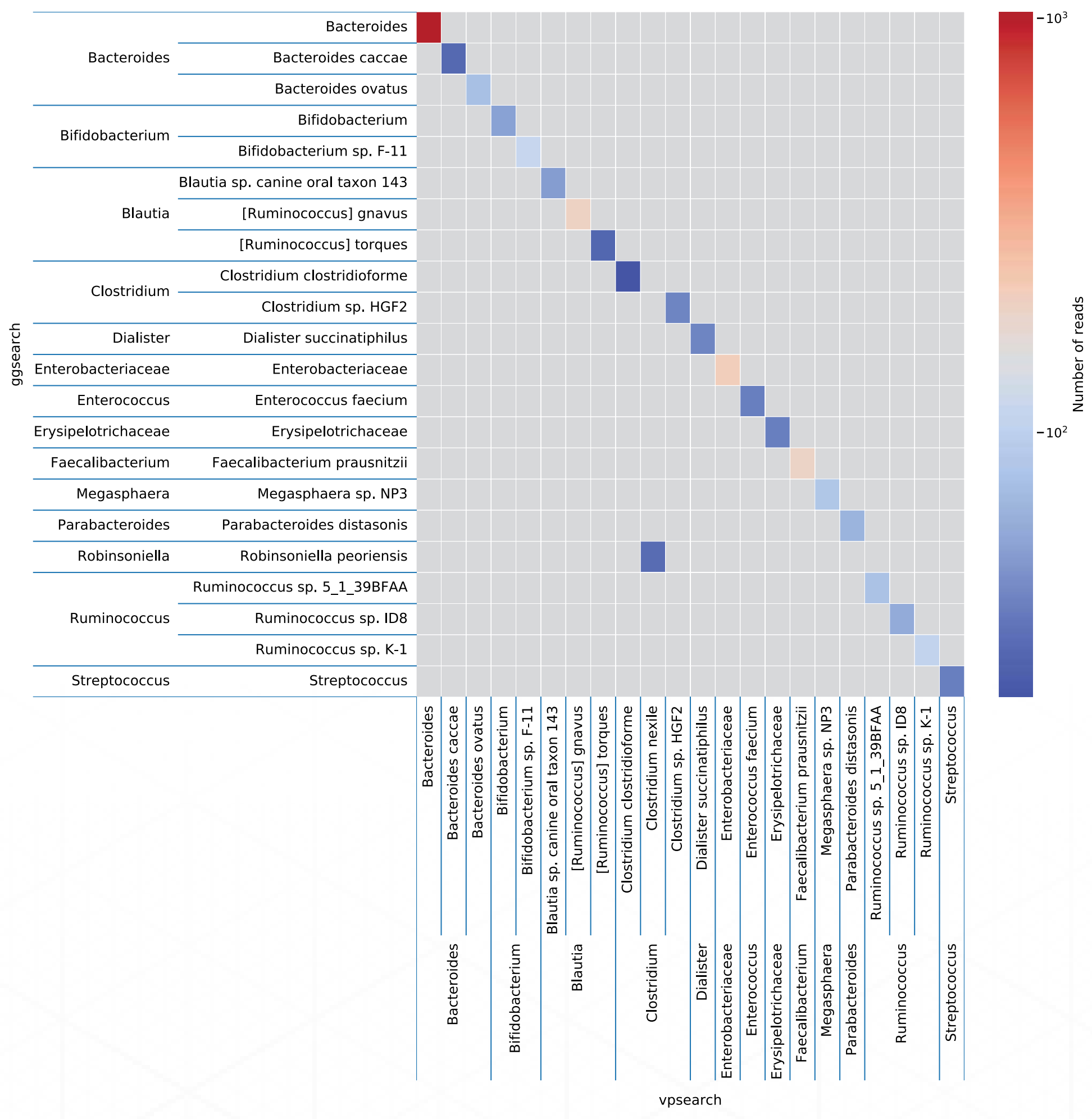


Figure 1 VPSEARCH shows similar concordance to direct alignment method
Comparison of taxonomic assignments in unknown bacterial sample using ggsearch (y-axis) or vpsearch (x-axis). Greater than 95% concordance was achieved. The 25 most populous OTUs are shown (including one misclassification).

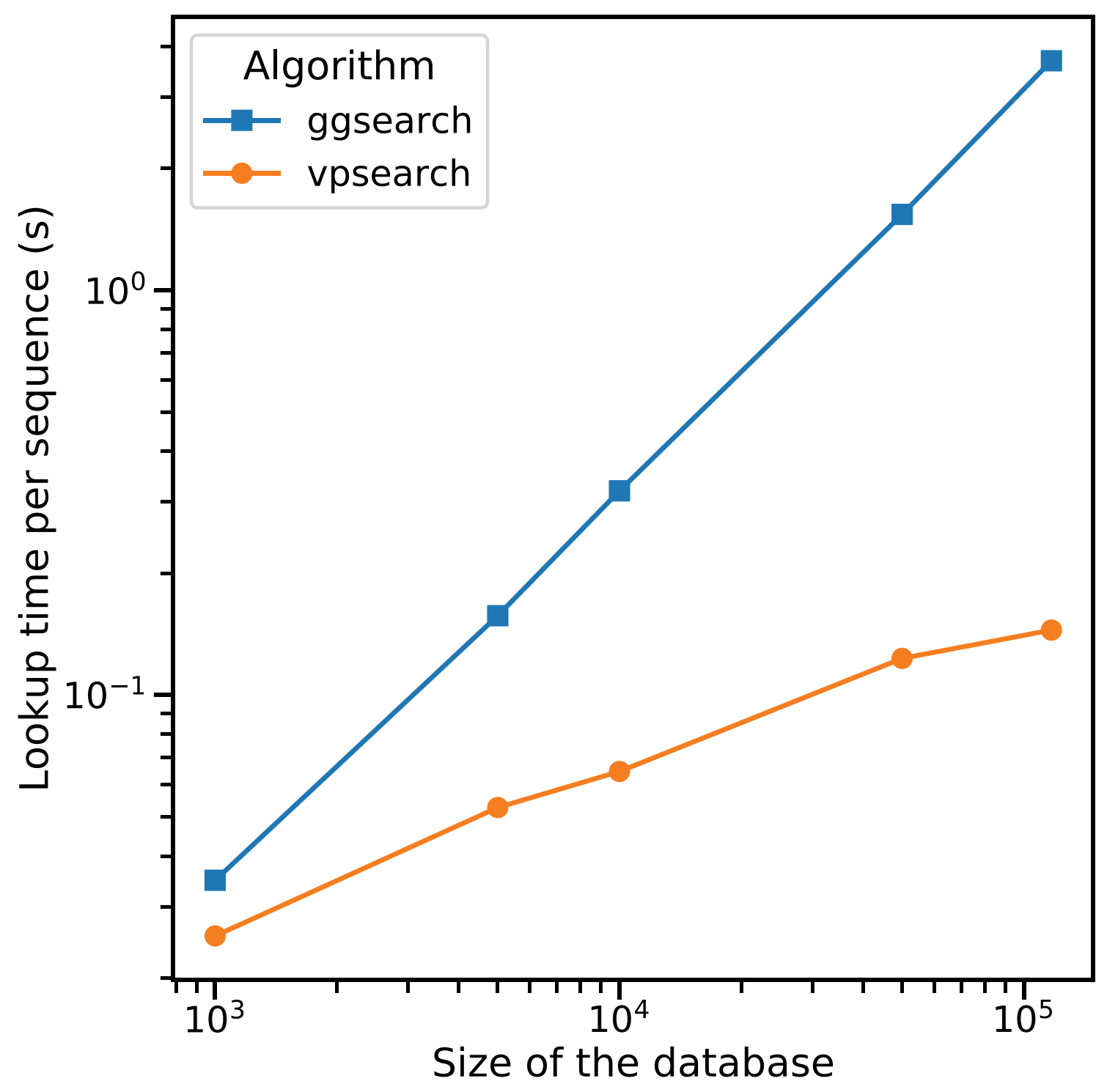


Figure 2 VPSEARCH improves database lookup speed
Using the vantage point tree methodology (vpsearch), sequence lookup time increases logarithmically as the size of the database increases. When using the same sequences, the alternative method, ggsearch, shows a linear increase in lookup time as the size of the database increases.

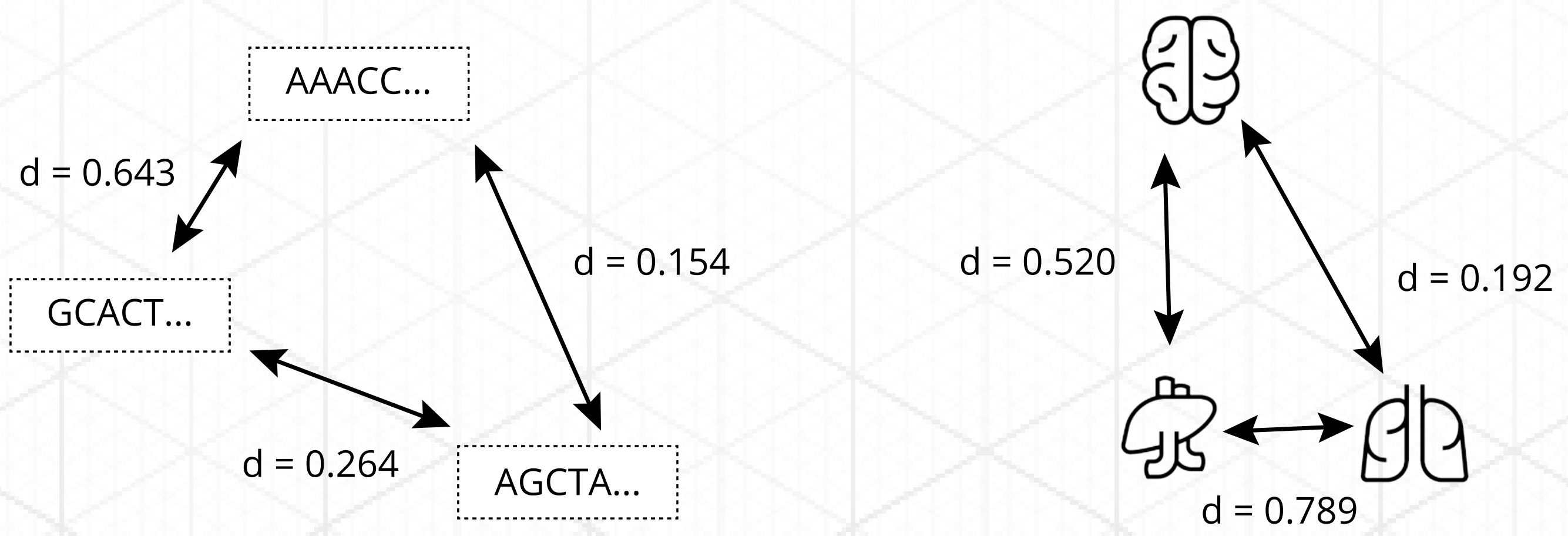
Conclusions

Vantage point trees classify genomic sequences effectively.

- Our Python/Cython implementation of vantage point trees may be used as part of standard analysis pipelines, returning data in BLAST 8-tabular output format.
- Using a bacterial dataset (16S rRNA amplicon sequences), a vantage point tree lookup improved speed 10X and classified OTUs with < 5% difference as compared to other traditional alignment-based methods.

Vantage point trees are not limited to short genomic sequences. By changing the metric function used, classification of more complex data is possible.

- Long reads (e.g. whole genome sequences) may use a k-mer based metric, instead of the exact global-to-global alignment metric used for short reads.
- Non-genomic data (e.g. cellular shapes) may be classified using an appropriate shape metric.



Vantage point trees may be used for a variety of applications, depending on metric function used, to approximate (dis)similarities by distance (d, where 0 < d < 1)

REFERENCES

P. N. Yanilos, Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces, Proc. Fourth ACM-SIAM Symp. on Discrete Algorithms, January 1993.

Uhlmann, J.K. (1991): Satisfying General Proximity/Similarity Queries with Metric Trees. In: Information Processing Letters. Vol. 40 (4):175--9.

IMPLEMENTATION

Our implementation code is available at: https://github.com/enthought/vpsearch/

Poster is available for download at enthought.com/lifesciences