Aligning Highly Variable DNA Sequences Using the W-curve and SQL

Steven Lembark[1], Shadi Beidas[2], Douglas Cork[2], Workhorse Computing[1], St. Louis, MO, 631101, Illinois Institute of Technology[2], Chicago, IL. 60616

Abstract

Unidimensional character strings are practical because of their simplicity. This simplicity made it possible to create the initial generation of tools for analyzing DNA. However there are some cases that these tools do not handle gracefully. DNA found in non-correcting viruses or oncology and "cloud" computing pose significant barriers to pushing these tools forward. The alternative we show here uses an abstract geometric representation of the DNA sequence called the W-curve. Geometric properties of these curves offer new avenues that bypass the roadblocks inherit in string-based approaches. Our approach described here uses a database with geometric fields and spatial indexes originally developed for geo-coding. The techniques improve handling of crossover-recombinant sequences and are suitable for distributed computing .

1. Background: Comparing DNA Character Strings.

The tools most commonly used today for analyzing DNA sequences are based on character-string representation of the DNA sequence. Representing the sequences in this manner makes intuitive sense and works for the most common cases. Analysis with these tools assumes largely similar sequences and that any differences between the sequences are significant. This approach works in most cases: for example, humans and chimpanzees have nearly 96% of genetic material in common [1],and nearly 60% of human genes are common to that of the fruit fly, *Drosophila melanogaster* [2].

The assumption of similarity starts to break down in studies of non-correcting RNA viruses or cancerous cells. The best-known example is HIV-1, the group also includes the filoviruses Marburgvirus and Ebolavirus and sequences found in cells damaged by radiation or botched mitoses. The process of studying these sequences today starts with a shotgun sequence from Next Generation Sequencing ("NGS") machines [3]. The following step is to align the small fragments output by NGS with template sequences. The high variance makes the fragments difficult to align, often requiring longer sequences for success. [4].

Crossover recombination is a common problem with HIV-1. The virus packages itself with two strands of RNA per viron. This leads leads to a fairly high rate of crossovers between the adjacent genomes in the viral progeny. Combined high rates of mutation and re-infection leave many patients with multiple distinct strains of HIV-1 infecting the same cells [5]. HIV-1's propensity for recombination makes hybrid strains relatively common compared to other viral infections. Similar problems arise in oncology studies where crossovers may be the cause of cancer: there is no good way to compare fragments to multiple template sequences at once utilizing the current generation of string-based techniques.

Compounding the problem is the scoring mechanism used by existing software, which produces a single value for the entire match. These tools offer no mechanism for comparing fragments piecewise and choosing the most relevant matches for each section.

BLAST, FASTA, and ClustalX2 all perform their comparisons recursively [6]. This works well enough for singly-threaded applications but is difficult to run in parallel. The growth of distributed computing environments makes it important to start looking for algorithms that are adaptable to parallel and highly-distributed execution. This requires an algorithm suitable for a divide-and-conquer approach, with the ability to compare regions separately and combine the results for final analysis.

2. Alternative: The W-curve

The W-curve was originally developed as a visualization tool for comparing large sequences of DNA. It uses a state machine to generate three-dimensional output based on the DNA sequence. The curve has a few useful properties for comparing sequences in the presence of SNPs and gaps, and its geometric result has more detail as the fine scale than a sequence of characters. Most important the location of points on the W-curve are influenced by the prior points on the

curve. This produces a more detailed structure which can be queried independently at each point on the curve [7].

Previous papers have described generating a W-curve in detail [8]. The W-curve is produced by a state machine using four corners of a square with corners labeled for the four bases in DNA (Fig. 2a). The X and Y axis are unit-less, the Z-axis is discrete, matching the sequence's base numbers. At each point on the curve, the next point is determined by gong half-way to the corner for the next base in X-Y (Fig. 2b). All curves begin at the origin, so the first point on all curves is at a distance of ½ along an axis with a Z value of 1.

Two important properties of the W-curve are shown in Figure 2. One is that altering one base in the sequence will change the locations of successive points in the curve. This is an important difference with character-based algorithms: Each point in the W-curve depends to a certain extent on the sequence of prior points. There is no analogous relationship between the bases in a character string. For example, knowing

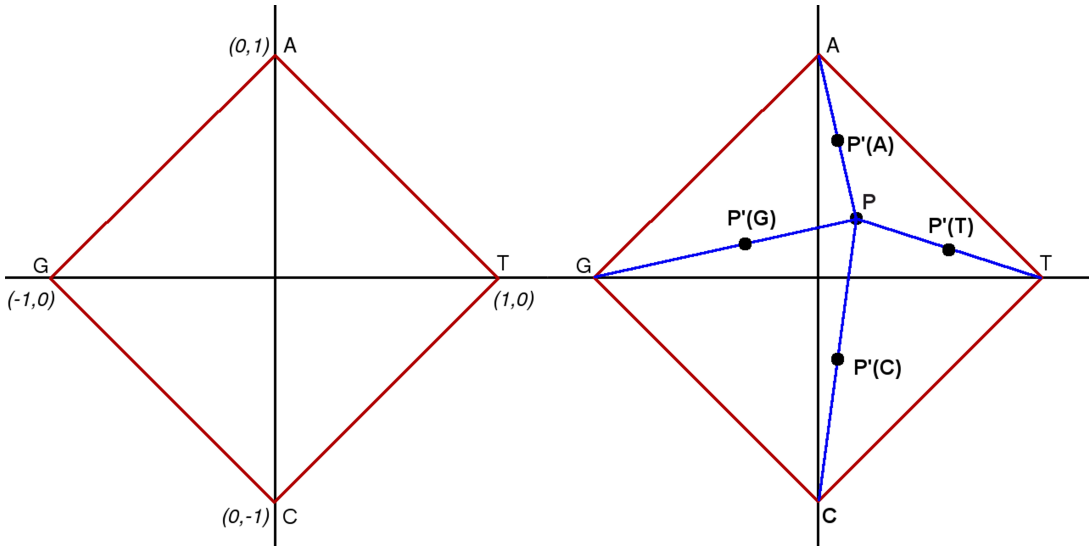

Figure 1: (a) Each corner of a square is labeled with one DNA base. (b) Successive points are halfway from the current point to the corner labeled with the next base. Shown are the next points from P for each possible next base (P'(A), etc).

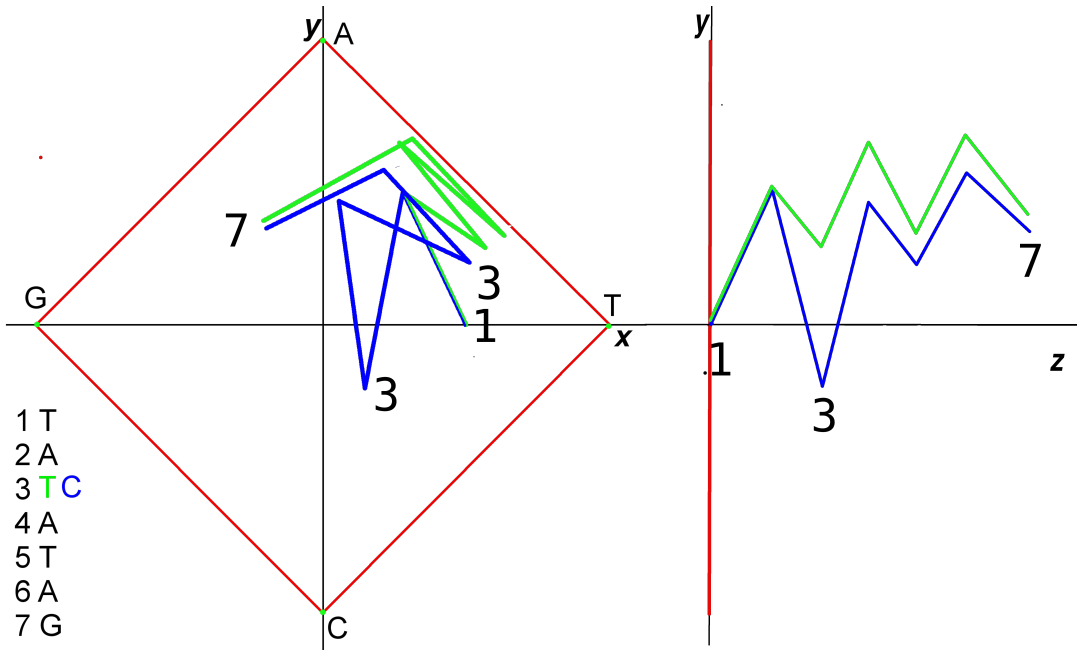

Figure 2: W-curve divergence and convergence after a SNP at base number 3 (T vs. C). The curves diverge noticeably at base 3 but have largely converged by base seven.This combination of divergence with auto-regression makes the curves useful for comparing sequences: local differences are detectable after which the curves converge due to auto-regression.

the x-y location of a point on a W-curve tells us something about the previous points. If nothing else, we know that if any of the last few points were different then the point would not be where we found it. This contrasts with a string-based sequence of characters: replacing any character in the string has no affect on the ones before or after it.

Another important property is that after a few common bases the curves rejoin one another. This property, called "auto-regression", is how the W-curve handles SNPs and gaps between the sequences. The effect can be seen in bases 4-7 of the curves in figure 2: the curves diverge noticeably at a SNP in base 3 but are nearly re-aligned by base 7. A similar effect is seen with gaps: within a few bases after a gap the curves converge with a phase-shift equal to the gap size.

The balance of local divergence and auto-regression makes it possible to align larger sequences while finding the differences between them. Auto-regression permits piecewise comparison of the curves since the alignment of any number of fragments will match their alignment taken as a whole. Regardless of local divergences from SNPs or gaps, common sequences will still align.

The following section will illustrate how the curves can be stored and compared using geometric extensions for relational databases.

3. Querying a Curve

Recent developments in relational database technology have added queryable geometric objects to the relational vocabulary [9]. We are using Postgres 9.1 with GiST objects (a.k.a. "postgis"). The geometric fields were originally designed for geographic or astronomical queries: find the roads in a city, or locations of postal codes. The constructs include points, lines, polygons, and circles which can be queried for overlap, intersection, inclusion or distance.

These database extensions also include "spatial indexes" which define bounding boxes for the geometric elements in the indexed fields. The indexes greatly improve performance in intersection or "contained within" queries.

There are any number of ways to apply these database extensions to model and query a W-curve. Our initial approach was similar to ones used with string-based approaches: began by selecting the template points close to the fragment's first base in X-Y. Then looking for points close in X-Y to the next point. However, this approach has problems with SNPs or gaps leaving no points to select in the next iteration or an initial SNP filtering out the correct templates.

One workaround for internal SNPs and gaps is to keep searching with an expanding window until one point is selected, then continue from that point forward. This approach handles internal SNPs or gaps does not account for crossovers or mismatches at the start of a curve. The problem with recombinant fragments is that they stop matching on one template at its midpoint, leaving us with nothing to select going forward. This approach is also not suitable for distributed computing since the process of acquiring each base depends on the previous one selected.

Some limitations of selecting incremental bases can be worked around by proceeding from both ends of the fragment

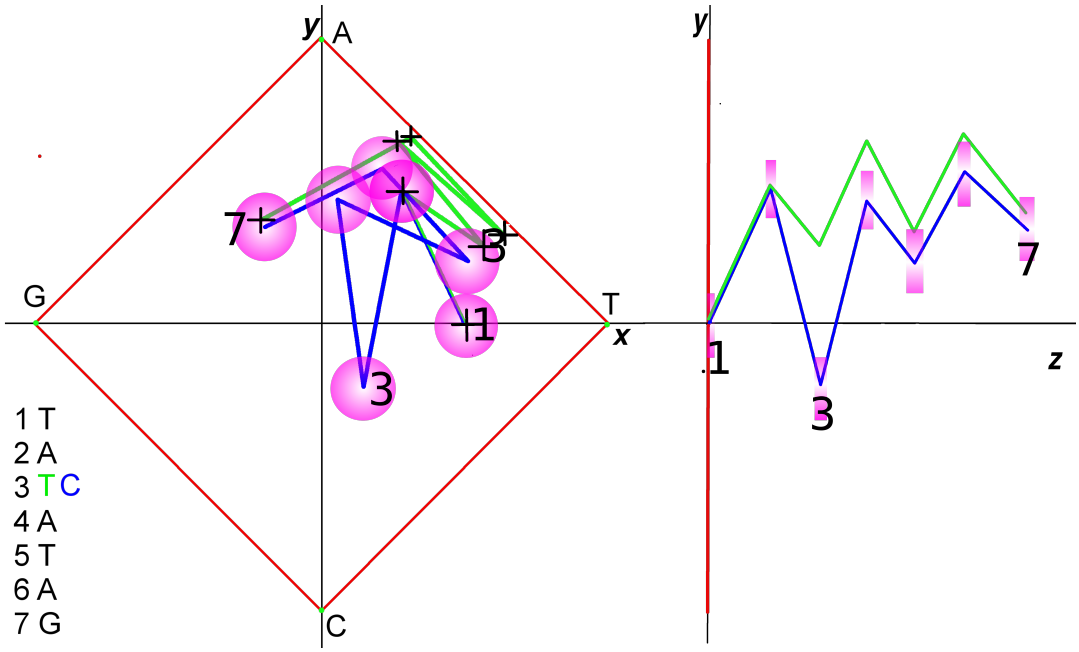

Figure 3: Overlap of fixed (green) point vertexes with fragment (blue) vertexes as circles using a radius of 0.10. After the curves diverge at base 3, the template's points intersect the fragment circles at base 7. Adjusting the circle size allows for fuzzy matching and helps compensate for progressive rounding error, variations caused by the fragment starting at (0,0), or multi-base/ low-quality FastQ entries.

at once: selecting any template points that match either end of the fragment and working back towards the fragment's center. This can handle crossovers but still leaves curves orphaned due to a SNP or gap at the ends of a fragment and is still not really suitable for distributed computation.

Avoiding issues with the endpoints requires dealing with points in the middle. Querying all of the points,will locate all the candidate templates in one pass. A more efficient two-pass approach is to first query a sample of the fragment and use those results to filter out trivial matches. The sampling approach we have developed is derived from the W-curve's original use as a visual tool: uses lining up the curves manually will start by making the extreme points match. Lining up these “peaks” in two curves is the fastest way to visually align the curves. In the database, this starts by selecting fragment points outside of a radius from the X-Y origin, usually 0.5. The first pass sample points are used to select matching template points,. The sample results are then filtered using adjacencies to remove trivial matches, providing a set of templates for complete matches. In the second pass, all points are compared to the templates with added restrictions based on the sample point matches. The result of this second selection are finally arranged for maximum coverage of the fragment and ranked by total coverage to produce candidate alignments.

This approach gracefully handles recombinant matches by simply locating the points on all available template curves. The queries are readily adapted to distributed computing since the point comparisons are independent, depending only on the positions of individual fragment and template point locations. The queries can restrict the locations of points using the relative base numbers of sample points, delivering a manageable amount of data to the central node for filtering. Even the filtering can be parallelized to the number of template sequences selected since those evaluations are independent.

The main remaining issue is defining a database which can be suitably queried.

4. Database Layout & Queries

The database schema that supports these queries has to deal gracefully with rounding errors, gradual effects of auto-regression after a SNP, and phase shifts in the curves after a gap. It also needs to be compact in both for query performance and distribution to nodes in the cloud.

One design that might seem attractive is storing the points as three-dimensional entities and simply querying the curves for intersecting points. This fails on two fronts, however, since it permit querying the X-Y locations of points independently of their base numbers or the direct selection of base numbers. Comparing the points without reference to their base numbers initially requires storing the curves with an X-Y value and separate base number. At this point storing X-Y values as points and querying the distance might seem reasonable. The storage is simple and compact, but the distance computation is too expensive and points alone are exquisitely sensitive to rounding errors. Storing all of the vertexes as polygons solves rounding errors but requires storing bulky objects with expensive intersect/overlap queries.

The final solution was a mixture of point and circle objects. The template W-curve vertexes are stored using X-Y points. The fragments, however, are stored as circles (Fig. 3). This provides a relatively compact database in both cases, with a simple query for the points contained within the circles. The circles also make effective use of spatial indexes, which store a bounding box containing the geometry. The bounding box for circles is efficient to compute, minimizes rounding error, and is an effective filter for the contained-within queries used with template points.

Storing fragment vertexes as circles also helps solve two issues with the W-curve that have been ignored thus far: initial bases in fragments and multi-base alternatives in the sequences. The former is a problem that W-curves generated from short reads all begin at the origin before their first base, but the template curves are in mid-sequence. This leaves the first few bases of the fragment's curve are guaranteed not to match the corresponding points on the template. One solution is to prefix the curve with the 16 possible two-base alternatives and draw a larger circle around the resulting locations for the first 2-3 bases. These larger circles are essentially a fuzzy-matching approach to the alignment. This approach permits matching in the first few bases of the fragment at the expense of filtering out more points due to extraneous matches.

Multi-base alternatives found in Fast Q output of NGS systems can be handled in a similar fashion: simply draw larger circles or store multiple *circularstring* objects in the database for the alternative bases. Resulting matches from each circle could be weighted according to the quality values for the final match. Again, the filtering process for adjacencies will remove any one-off matches.

A skeleton database supporting these queries has four tables: two of identifiers with any additional non-geometric data and two of W-curve values, one with points one with circles for the geometry (Fig. 4). The identifier tables have a candidate key of the template's external identifier and an integer surrogate key for use in the geometry tables. The template geometry table has a candidate key of the template's surrogate key (SK) and base number; fragment geometry requires a candidate key of the fragment's SK, a base number, and the base AA from which the geometry is defined. The

```
create table sequence
(
    id            serial          not null,
    parent        integer         not null references dna default 0,
    ident         varchar(32)     not null,
    sequence      text            not null default '',

    primary key ( id ),
    unique ( ident, parent )
);
create table template
(
    seq           integer not null references sequence,
    base          integer not null,
    nucleotide    char(1) not null,

    primary key ( dna, base_no )
);
create table fragment
(
    seq           integer not null references sequence,
    base          integer not null,
    nucleotide    char(1) not null,

    primary key( dna, base_no, base_aa )
);

select AddGeometryColumn( 'template', 'vertex', -1, 'POINT',          2 );
select AddGeometryColumn( 'fragment', 'vertex', -1, 'CIRCULARSTRING',  2 );

create index fragment_vertex_ix on fragment using gist( vertex );

insert into sequence ( id, parent, ident ) values ( 0, 0, 'root' );
insert into sequence ( ident ) values ( “B.K03455” );
insert into sequence ( ident, parent ) values ( “gp120”, 1 );

prepare     insert_template( integer, integer, char, varchar )
as          insert into template ( seq, base, nucleotide, vertex )
values      ( $1, $2, $3, St_GeometryFromText($4) );

prepare     insert_fragment( integer, integer, char, varchar )
as          insert into template ( seq, base, nucleotide, vertex )
values      ( $1, $2, $3, St_GeometryFromText($4) );

insert_template( 1, 1, 'T', “POINT(0.5 0)” );
insert_fragment( 2, 1, 'A', “CIRCULARSTRING(-0.05 0.5,0.05 0.5, -0.05 0.5)” );
```

Figure 4: Skeleton database for storing template and fragment W-curve vertices. Example inserts show the first vertex for HXB2 and its gp120 protein.

fragment's larger candidate key is required to accommodate storing multiple circles for handling FastQ results.

The circles are handled via *CIRCULARSTRING* objects. These can describe full circles using three points with the first and last points matching and the second point being opposite the first. Using an offset to the X-axis value for each vertex requires minimal computation for the input data and produces a bounding box without rounding error. For example, in Figure 3 the first vertex for 'T' at (0, 0.5) produces a circle with points (0, 0.55), (0, 0.45); in Figure 4 shows the input format with three points, with the initial vertex for 'A' at *CIRCULARSTRING(-0.05 0.50, 0.05 0.50, -0.05 0.50), -1.*

The general query for alignments selects the dna.id, base_no, and base_aa values from template, fragment tables "where template.vertex && fragment.vertex". The '&&' operator looks for intersecting geometry and makes efficient use of bounding boxes in the fragment's spatial index.

5. Further Research

Determining the most effective radius for the fragment radius and how to use either variable-radius or multiple-circle designs will be important for matching short sequences provided as FastQ inputs used with most NGS machines. In addition, an efficient, distributed filtering algorithm for the first pass selection from sample fragments will be key to making this approach efficient in cloud-computing environments.

6. Conclusion

The W-curves' abstract, three-dimensional geometry for representing DNA sequences provides more detail than a uni-dimensional character sequence. Its balance of local divergence and global convergence make the representation useful for aligning sequences that character-based algorithms cannot handle well. Geometric data objects now give us the tools to mine W-curve databases effectively, handling highly variable and crossover recombinant sequences. The algorithm described here uses generic tools such as SQL, and is suitable for highly-parallel environments such as cloud computing. Although the examples here use HIV-1, other non-correcting viruses or oncology are also good candidates for its application. We are not saying that the W-curve is a replacement for Fasta or Clustal, but used as an adjunct to them it opens up new opportunities for studying difficult sequences. And that has to be our goal going forward: expanding the range of tools available for studying the complexity of biology.

7. References

1. The Chimpanzee Sequencing and Analysis Consortium. Initial sequence of the chimpanzee genome and comparison with the human genome. Nature 2005 Sep 1;437:69–87.

2. Remsen J, O'Grady P. Phylogeny of Drosophilinae (Diptera: Drosophilidae), with comments on combined analysis and character support. Molecular Phylogenetics and Evolution 2002, 24:249–264.

3. Lee C, Grasso C, Sharlow MF. Multiple sequence alignment using partial order graphs. Bioinformatics 2002, 18:452-464.

4. Grasso C, Lee C. Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. Bioinformatics 2004, 20:1546-1556.

5. Christophe Fraser. HIV recombination: what is the impact on antiretroviral therapy? J R Soc Interface. 2005 December 22; 2(5): 489–503.

6. Biegert A, Söding J (March 2009). Sequence context-specific profiles for homology searching. Proceedings of the National Academy of Sciences of the United States of America 2009 March;106(10): 3770–5

7. Cork DJ, Toguemf A. Using fuzzy logic to confirm the integrity of a pattern recognition algorithm for long genomic sequences: the W-curve. Ann N Y Acad Sci. 2002 Dec;980:32-40.

8. Cork DJ, Lembark S, Tovanabutra S, Robb ML, Kim JH (2010) W-Curve Alignments for HIV-1 Genomic Comparisons. PLoS ONE 5(6): e10829. doi:10.1371/journal.pone.0010829

9. Ahmad N et al. Preserving Data Replication Consistency through ROWA-MSTS. Communications in Computer and Information Science 2011;180(1): 244-253.