Rotation Report

by Amir Alavi
Kingsford Group
September 8th – October 7th, 2016
aalavi@andrew.cmu.edu

Summary

During my time with the lab, my goal was to augment our Topologically Associating Domain (TAD) identifying software, Armatus, with resolution-partitioning capabilities, as well as to explore new methods for identifying TAD hierarchies in the chromosome. Instead of modifying the current algorithm (with new objective functions or quality functions), I decided to take a completely different approach to the problem. The methods described in this report outline a simplistic algorithm that shows some promise, and is believed to have less influence on domain sizes compared with Armatus' γ parameter.

After prototyping the method in Matlab, I was able to show that it results in domains that are closer to those generated by Dixon et al. than to those generated by Armatus. Meanwhile, this new algorithm is much faster than either method. While this shows some promise, I also discuss important areas for future work.

1. Introduction

Ever since its publication, Armatus has become a popular tool for identifying TADs. Its dynamic-programming strategy is both fast and elegant. However, the user must specify a resolution parameter γ which can be a heavy-handed way of influencing the domain sizes. This is evident in the typically very small domains that Armatus generates compared to other methods.

What's more, Armatus uses one γ value throughout its entire scan of the chromosome. Although a consensus domain is drawn from multiple resolutions in the end, we would like to find a way to free Armatus from this heavy resolution influence during its singular scans. Though not the main focus of this work, we would also like to identify TAD hierarchies.

2. Methods

All code is available at: https://github.com/AmirAlavi/tadFindingPrototypes

My proposed approach is very simple, and based on detecting edges in images. However, because of the approximate block diagonal nature of HiC data, we only consider rectangular edges, and thus our task is easier than edge detection in photography.

2.1 TAD Finding Algorithm

2.1.1 Algorithm at a High Level

The algorithm is a linear scan through the HiC "adjacency matrix." It's easiest to think of the linear scan happening along the diagonal of the matrix. For simplicity's sake, let's reduce the problem to identifying the edges of the blocks in a block diagonal matrix like the one below.

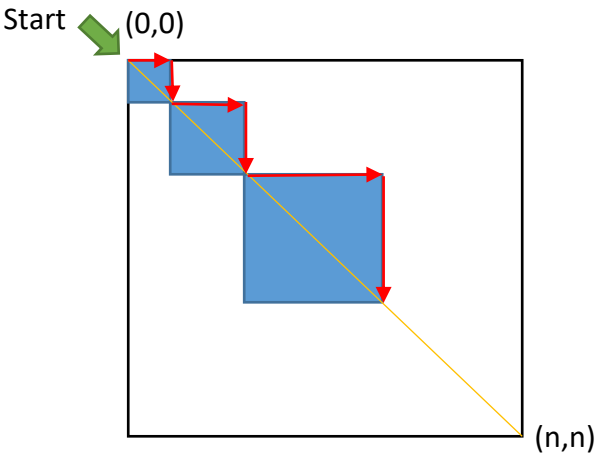

Figure 1 *Cartoon depiction of the algorithm operating on a block diagonal input matrix. Starting from the upper-left corner, the algorithm follows the red arrows down the matrix, and the points where the red arrows touch the diagonal define the domain boundaries.*

We begin at the upper-left corner. Throughout the scan we are updating an (x,y) coordinate pair to keep track of our current location. Our current location will always be in the upper-triangle of the matrix. In each iteration, we determine if our current location is the x-location of one of the vertical edges of a block. If not, we keep moving horizontally, incrementing x while maintaining the same y value. However, if we determine that we are at an edge, we "drop" back down to the diagonal by setting our y value equal to the x value, thereby marking the end of a domain.

2.1.2 "Windowing" Method for Detecting Edges

The way we detect an edge is by drawing a "window" from our current location, and comparing the density in its left half to the density in its right half. If the left half is more dense (beyond a certain **tolerance parameter**), then we say that the boundary between these two halves is the edge of a domain. In this report, unless otherwise noted, a tolerance parameter of 0.1 is used.

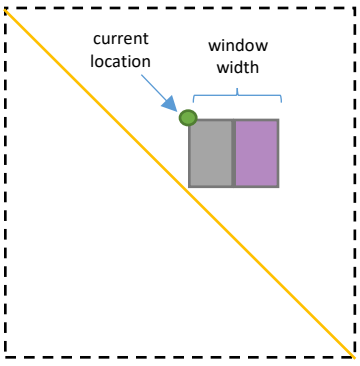

Figure 2 *Zoomed-in view of the "window" drawn from the current location in the algorithm. The grey box is the left half, and the purple box is the right half. Notice that the height is the vertical distance from the current location to the diagonal.*

The exact dimensions of this window are as follows: the upper-left corner of the window is the current (x,y) position, the lower-left corner is drawn from the upper-left all the way down to the diagonal, and the upper-right corner is at (x, y + **window width**).

Although tolerance and window width are yet another set of parameters that one has to set, I believe that they do not have as strong an effect on the size of the resulting domains when compared to γ. This should be obvious for the tolerance parameter: there is no setting of tolerance that should prefer a domain over another solely on the basis of size. For the window width parameter, I have seen through experimentation that if one chooses a small enough window size (for example, 5 pixels when each pixel is 40kbp), the effect of this setting on domain sizes is small. Obviously, the window width is effectively a minimum size for the domains, so one should pick a window width that will be smaller than most domains so as to reduce its effect on domain size. Throughout this report, all results were created using a window width of 5.

If our window heights will always be below a certain value ϵ, and $\epsilon \ll n$, then the density calculation is nearly constant and the whole algorithm is $O(n)$. Finding the value of ϵ is completely dependent on the particular input matrix used. However, the intuition is that the nature of HiC data is such that $\epsilon \ll n$ (the "current location" in the algorithm will not stray too far from the diagonal.

2.1.3 Finer Details of the Algorithm

An issue with HiC data is that there are regions for which no data is collected. Thus, a naïve algorithm that assumes it should immediately begin the next domain after it marks the end of the current domain would do poorly. To cope with this, I have added a **minimum density** parameter. If the density of the current window is below this minimum density, it will end or refrain from starting a domain. In my experiments, a minimum density value of zero was effective in avoiding these areas with no data while still including all data-containing regions. This is because the regions with no data are defined by sharp boundaries.

2.2 Testing

I tested the algorithm on simulated data in the form of random, binary, block diagonal matrices. That is, I generated randomly sized matrices of all ones, and placed them adjacent to each other along their diagonals, leaving the other elements in the enclosing matrix as zeros. Since these "perfect" matrices represent a trivial problem, they were used as a sanity-check to ensure the algorithm was working as intended.

To test the robustness of the algorithm and begin tuning the parameters, I added Gaussian or "Salt & Pepper" noise to the simulated data sets. This was particularly useful for exploring the effects of the tolerance parameter. However, the parameter values for tolerance did not translate over to real HiC data since this simulated data had only binary elements.

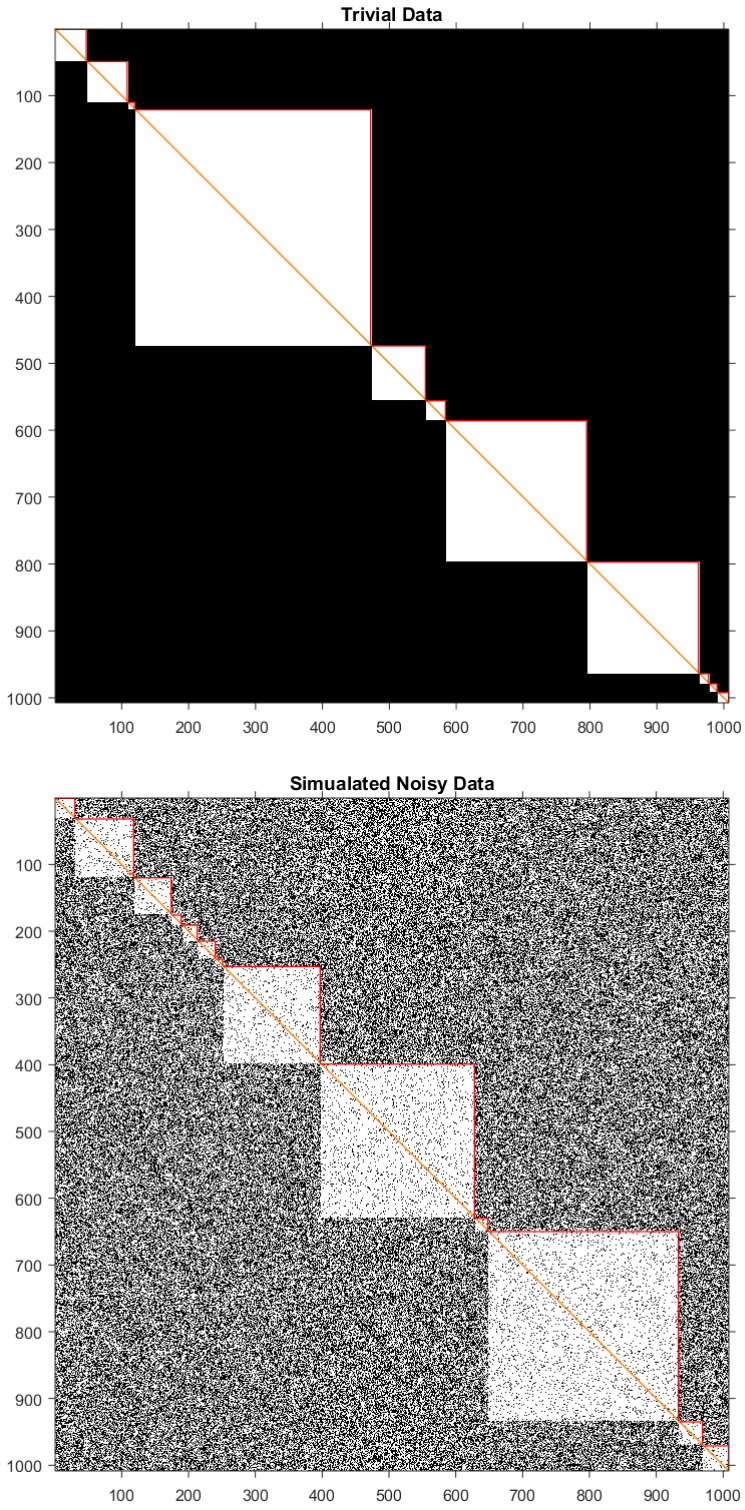

Figure 3 *Examples of testing the algorithm on simulated binary data. The domains are drawn in red. To produce the domains for the trivial data set, the parameters were set to tolerance=0.1, windowWidth=5. However, since the boundaries are so defined, many tolerance values will produce the correct results. To produce the domains for the simulated noisy data set, the parameters were set to tolerance=0.25, windowWidth=5.*

2.3 Comparisons to Other Methods

I compared the domains generated by this method to those generated by Armatus and those generated by Dixon et al. The comparisons were done by visually drawing them on the HiC matrix, comparing their mean and standard deviation of sizes, and calculating the variation of information (VI) between them.

In order to make use of the VI metric, we need a baseline VI number to compare against. I generated this number by randomly shuffling the domain set I wanted to compare against and calculating the VI of my domain set and that shuffled domain set. By "shuffled" I mean that we conserve the lengths of the domains, but randomize their start and end locations (while still maintaining that they do not overlap). I did this 1000 times. This resulted in a distribution that I could then compare my VI number against.

3. Results

I ran the algorithm on a long and short chromosome, using human IMR90 fibroblast HiC data at 40kb resolution from Dixon et al. The images below show that upon visual inspection, the domains identified by this algorithm seem reasonable.

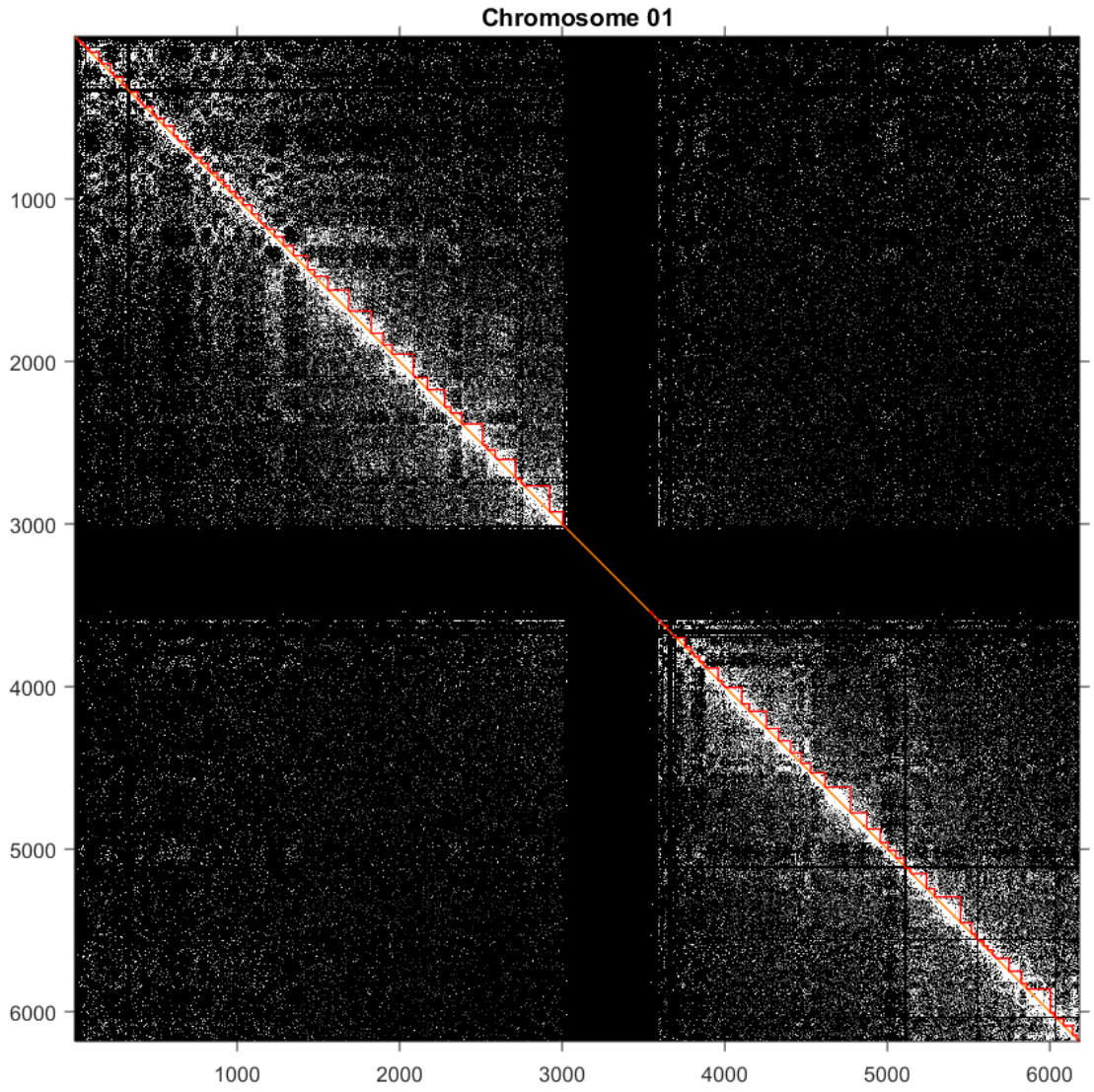

Figure 4 *The results of running the algorithm on 40kbp HiC data from human IMR90 fibroblast chromosome 1.*

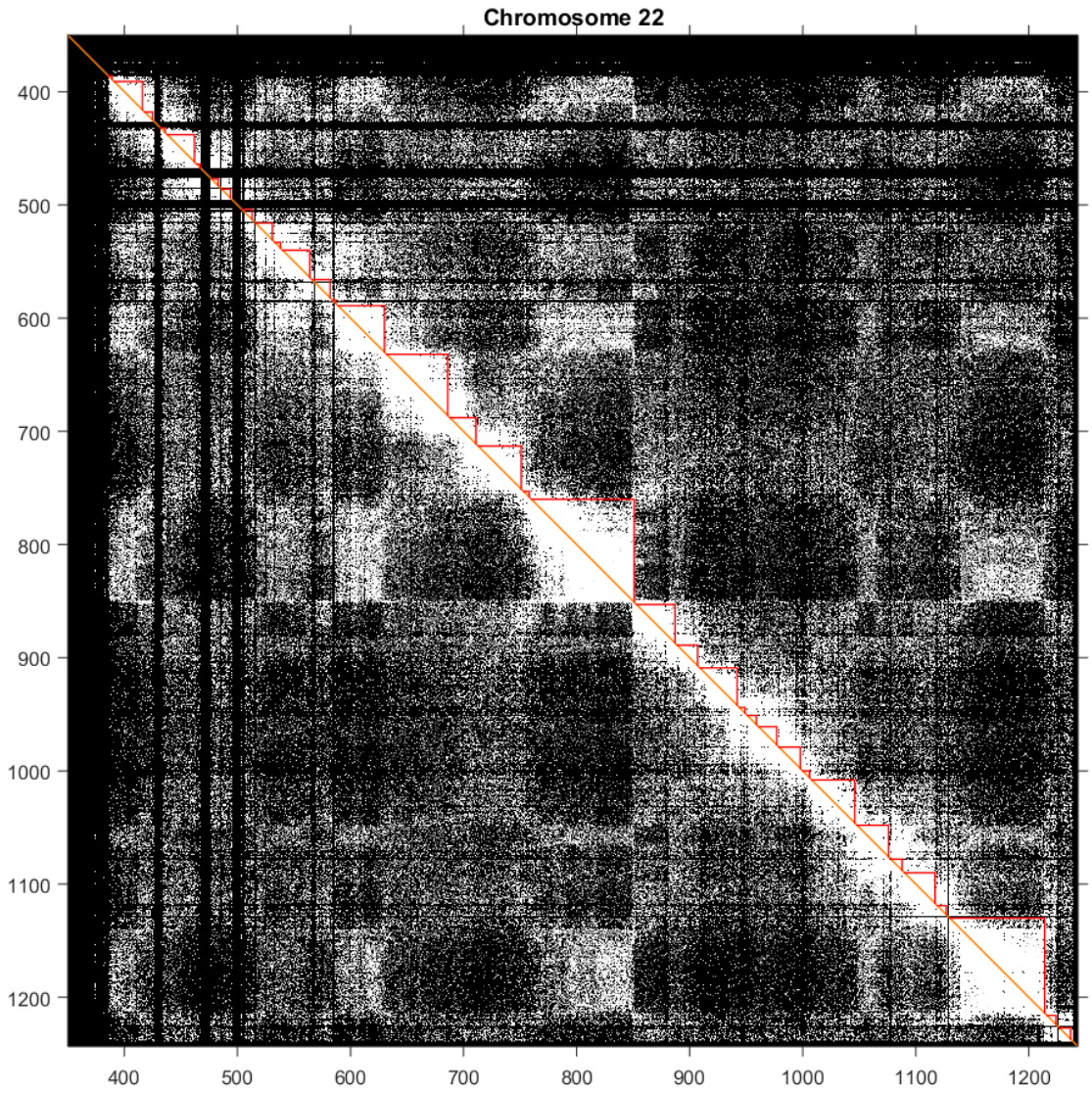

Figure 5 *The results of running the algorithm on 40kbp HiC data from human IMR90 fibroblast chromosome 22.*

When we compare to Armatus, we see that the Armatus domains are much smaller, and more uniform in size. Our domains are much larger (which is not necessarily better) and more diverse in size.

When we compare to Dixen et al. domains, we see that our domains are closer to these than to Armatus domains. The data and images below support this claim.

Table 1 *Distribution of Domain Sizes for Three Domain Finders*

Method	**Mean domain size**	**Standard deviation**
Our new method	38.028986	37.718501
Dixon et al.	22.746835	16.432297
Armatus	7.797170	9.307532

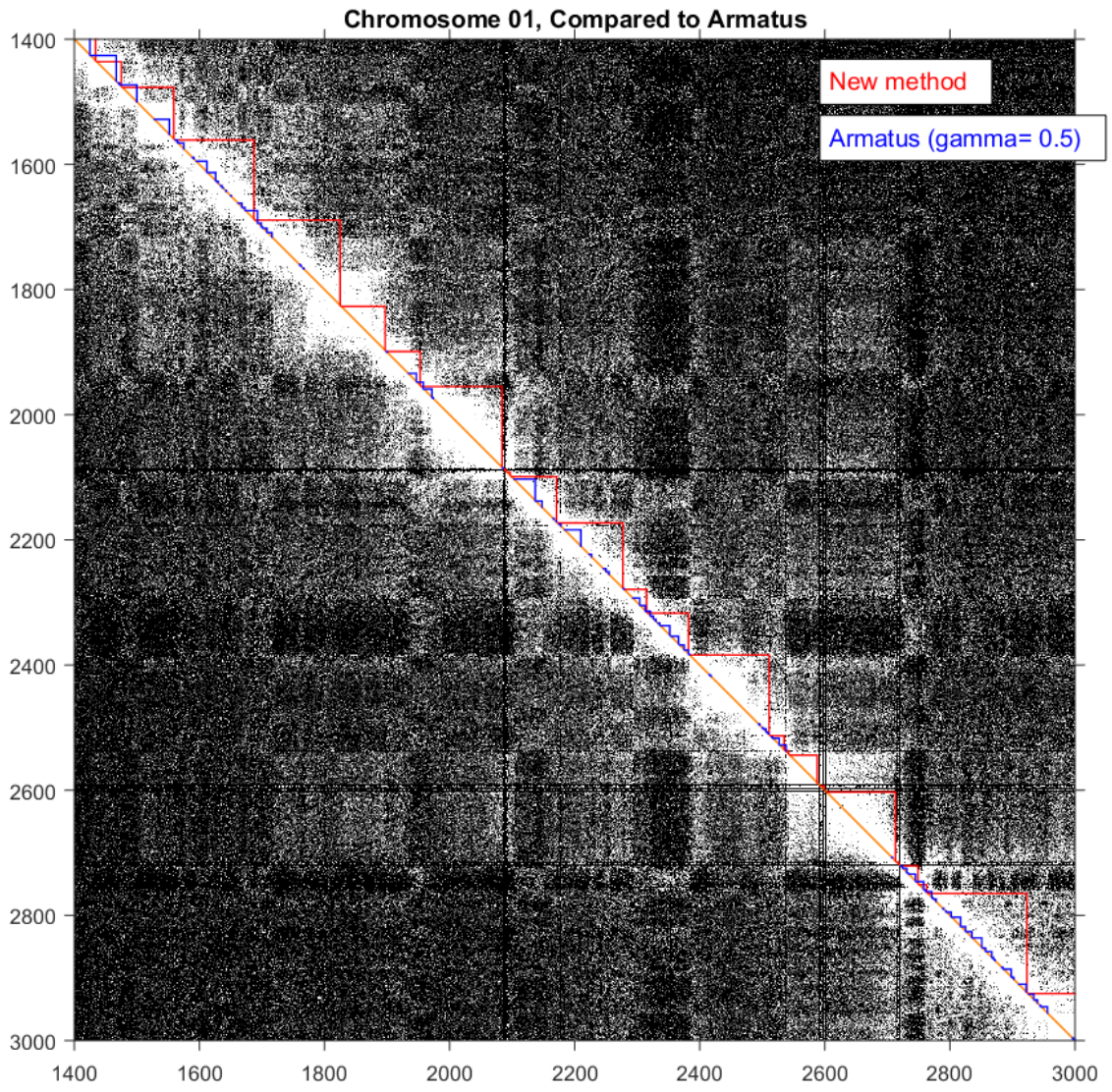

Figure 6 *Comparison of domains identified by our new method compared to Armatus on hIMR90 fibroblast chromosome 1. The apparent small size of the Armatus domains compared to our new domains stands out.*

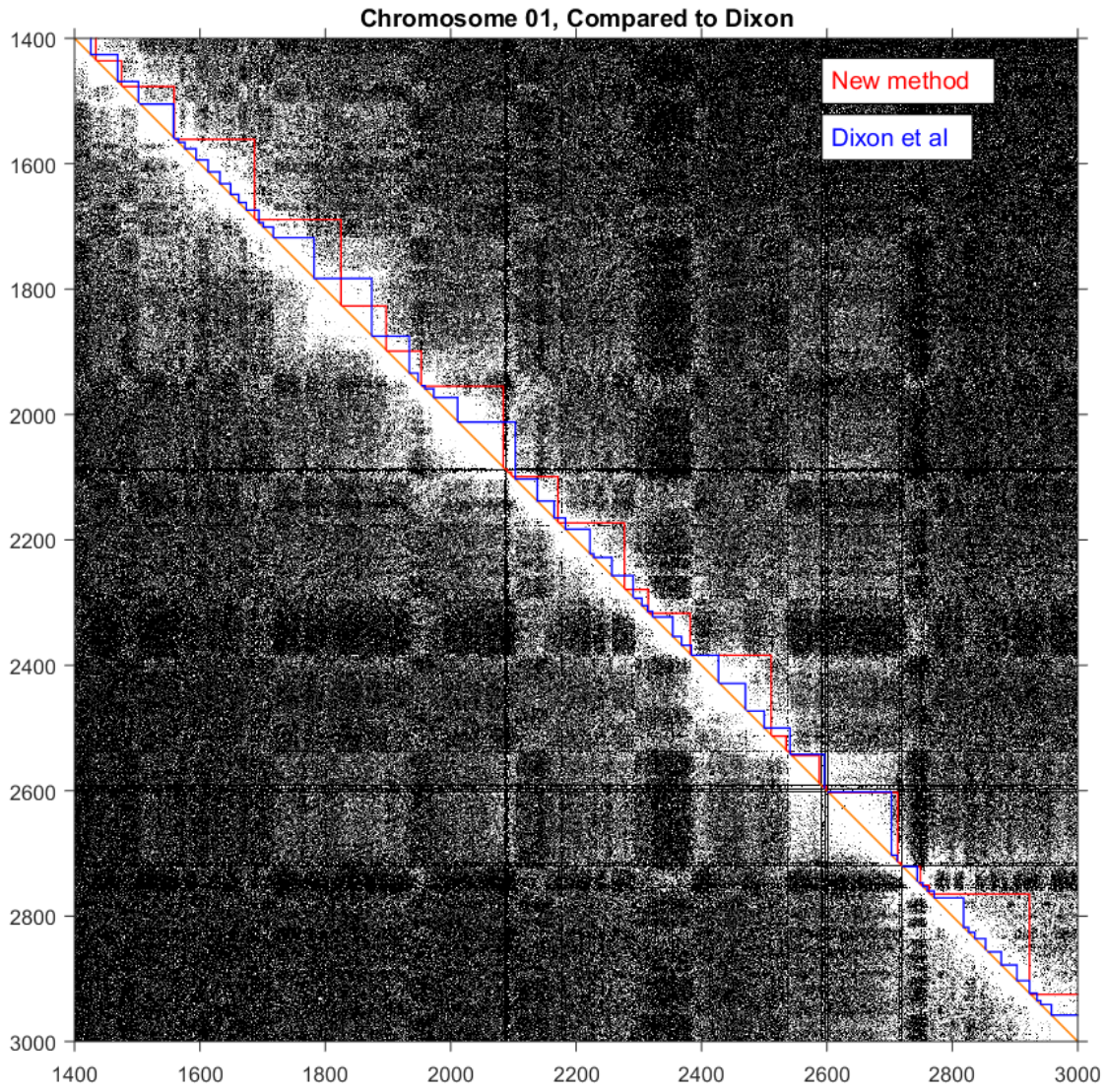

Figure 7 *Comparison of domains identified by our new method compared to Dixon et al. on hIMR90 fibroblast chromosome 1. We can see that our domains are closer to the size of Dixon's domains compared to Armatus domains. We can also see that our domains can be viewed as hierarchical parents of the Dixon domains.*

This leads to the question: how similar are the domains between our new method and Dixon et al? We would like to answer this question because it could illuminate the value of our new method. In order to do this, I used the variation of information metric.

I calculated the baseline VI distribution for my domains on chromosome 1 vs the domains found by Dixon et al. I then compared the actual VI to this distribution. The results are shown in the plot below and suggest that our VI is significantly lower than the baseline distribution. This means that the information represented by our domains is similar to the information represented by Dixon et al's domains.

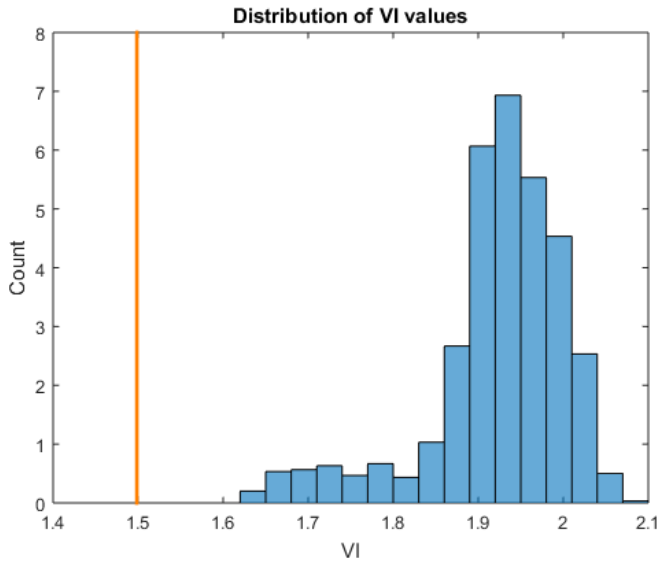

Figure 8 *The distribution of VI values resulting from the random shuffling method described above. The orange line marks the VI value from comparing our domains to Dixon et al. domains on hIMR90 fibroblast chromosome 1. We can see that it is much lower than the distribution.*

4. Conclusions

During my rotation, I prototyped and experimented with a simple TAD finding algorithm. Though primitive in its approach, it is an interesting solution because it does not have a parameter that influences domain size as heavily as Armatus' γ parameter. Another benefit of this approach is its intuitiveness and speed on typical HiC data.

Though this new method has shown some promise, there is still much to be done. One area that future work could focus on is exploring the parameter-space to determine the best set of parameters for TAD finding. Currently, the defaults in the code are what I have seen to be the best through preliminary testing.

Another interesting area to work on is exploring different tolerance settings as ways of building a **hierarchy of domains**. While running my algorithm and visualizing its domains, I noticed that it appeared as though the domains were enveloping the Dixon et al. domains. This leads to the question: can we use different settings of tolerance to wrap the Dixon et al. domains in interesting hierarchies? The Figures 9-11 below attempt to illustrate this. As we decrease the tolerance value, our domains wrap tighter and tighter around the Dixon et al. domains.

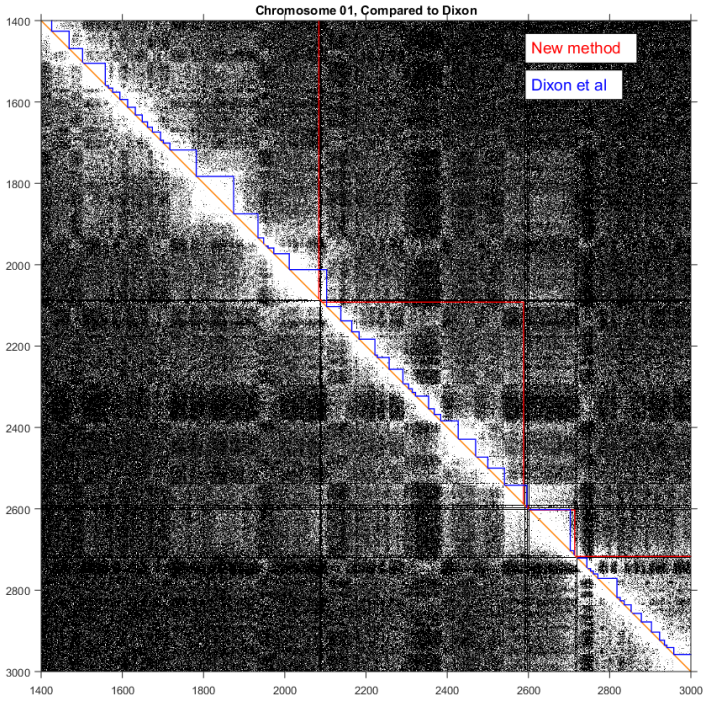

Figure 10 *Domains drawn with tolerance = 0.3 compared to Dixon et al. domains on a portion hIMR90 chromosome 1.*

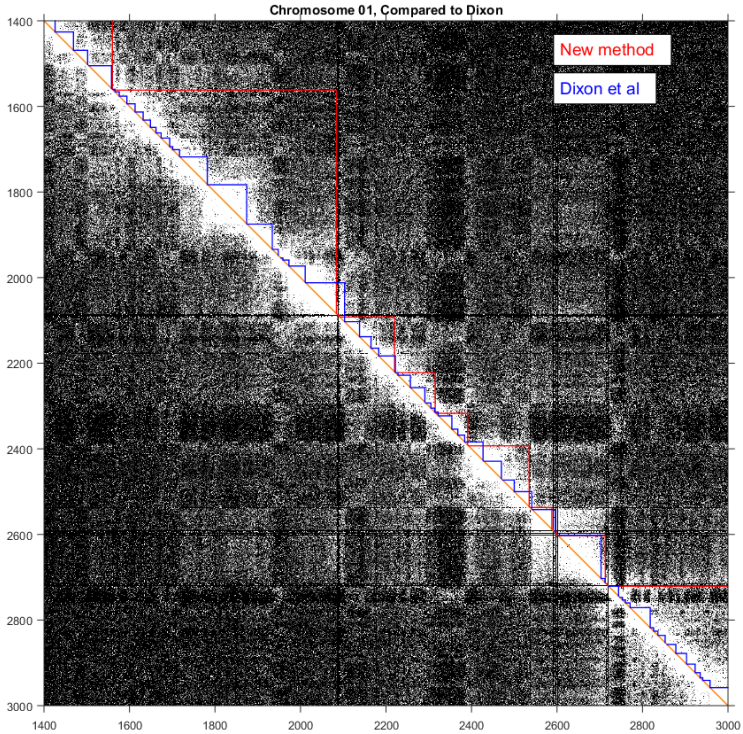

Figure 9 *Domains drawn with tolerance = 0.2 compared to Dixon et al. domains on a portion hIMR90 chromosome 1.*

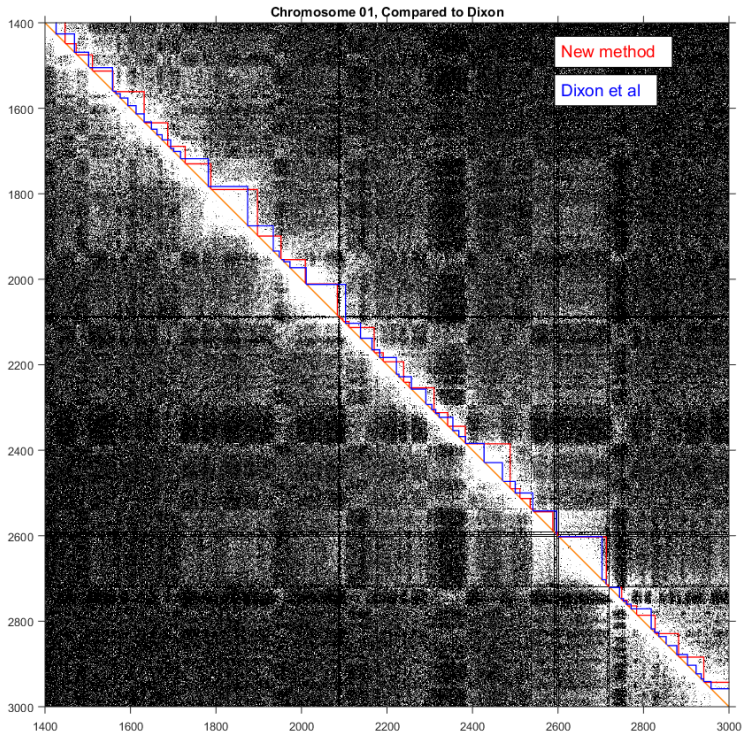

Figure 11 *Domains drawn with tolerance = 0.05 compared to Dixon et al. domains on a portion hIMR90 chromosome 1.*

Of course, further testing is required. In particular, using higher resolution HiC matrices like those in Rao et al. would be a good next step. In addition to this, we should look at the CTCF and histone marker enrichment of the domain boundaries that this algorithm produces, and compare them to Armatus and Dixon et al.

5. Acknowledgements

Thank you to Professor Carl Kingsford for giving me the opportunity to work in his lab over the past four weeks. He was generous with his time, and met with me several times to discuss and share his ideas.

I would also like to thank Natalie Sauerwald for providing mentorship and help while I was working on this project. She was also generous with her time and knowledge, and even helped move the project along by sharing some of her code from other work.