Read Length Magnitude Micro Inversion Remediation

Dan Smillie, Oneeb Malik, John Wilson, Joshan Bajaj

EN.600.439 Computational Genomics
December 9, 2016

Abstract

Micro inversions occur naturally in humans across different populations and are known to have serious biological implications. A micro inversion is an area in a person's DNA where instead of the expected nucleotides appearing, the reverse complement does. Generally, they are considered to be of size 15 to 100 nucleotides long. A correlation has been established between micro inversions and developing certain diseases, like cancer, so being able to detect micro inversions is an important problem that is still being researched by experts in the field. The goal of this project is to develop an algorithm to detect micro inversions to help correct misaligned reads that have micro inversions within them. The algorithm was evaluated by testing it against incremental simulated data, and then it was used against generated reads from Chromosome 21. The simulated data used was generated by a program that generates reads based on a given chain of nucleotides, and then it inserts micro inversions within them. The micro inversions detected were between 15 and 100 nucleotides, but the algorithm is adaptable to handle micro inversions of any size less than the length of a single read. While the time and space complexity of our algorithm are very large, it successfully detected micro inversions in both our simulated data and in the larger Chromosome 21 dataset so we are pleased with our results.

Introduction

The importance of detecting micro inversions is critical and is a problem that is still being worked on by researchers in the field of Genomics. As such, it seemed an interesting task to tackle in this project. The main challenge in finding a micro inversion was to identify where in the read the micro inversion occurred, since it could be between 15 and 100 nucleotides long and there's no defined start and stop area. Creating a program that finds micro inversions with complete accuracy operates rather inefficiently in terms of time. The method used spends a large amount of time preprocessing the original sequence into a dictionary with a five tuple of the nucleotide bases as a key that maps to the locations in the sequence where such the same base density occurs. Then, to find micro inversions in the reads, another program takes in each read individually and creates every possible manipulation of the read by converting parts of the read to its reverse compliment to see if it exactly matches an element within the dictionary with same five-tuple of base density. This approach was chosen despite its slow runtime on larger data sets because it guarantees correct results.

Prior Work

Several micro inversion detection programs have been developed for research ranging from identifying genetic disease causes in humans [1] to genetic differentiation of phylogenetically similar species. [2] The definition of micro inversions varies slightly from study to study depending on the research goals, but they are generally agreed to

be reverse complement genetic inversions that are between 15bp and 50kbp, which is smaller than most complete human genes.

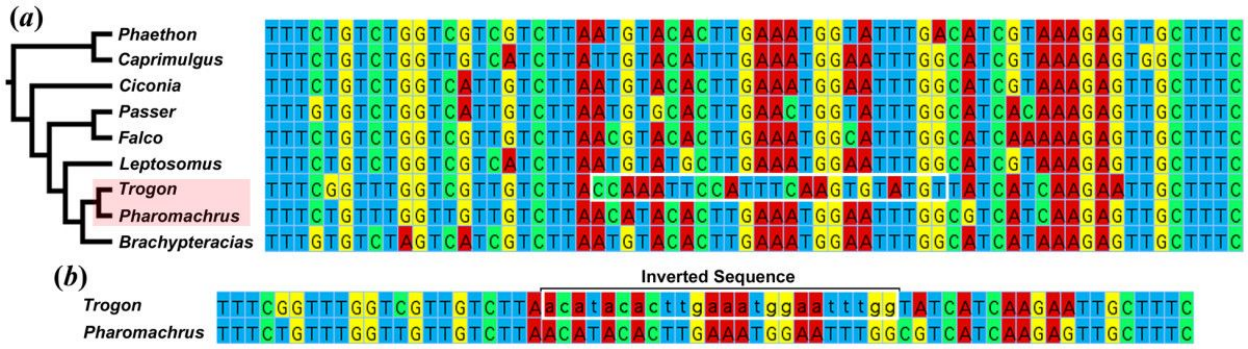


Figure 1: An illustration of taxonomic branching between the Trogan and Pharomachrus auriceps species, with a 24bp micro inversion in the TPM1 intron 6 region. Homoplastic micro inversions have been found to be useful phylogenetically markers. [2]

Especially with human sequence data, alignment programs are able to use relatively strict mismatch penalties for structural variants when aligning reads because we have strong confidence in human reference data that is constantly being updated and improved. When aligning sequenced human reads to the reference genome using alignment programs such as Bowtie, some reads are ultimately unmapped and deemed useless because there is no obvious position they overlap with in the reference genome. Micro inversions are longer and more complicated structural variants to detect relative to more common insertions, deletions, and substitutions, so naturally it is more difficult for current alignment programs to map reads with micro inversions.

Our program seeks to remediate reads that have been misaligned because they contain large enough micro inversions to disrupt their alignment penalties to the point that they are unusable reads for the assembly. With the thinking that reads with micro inversions that may be biologically relevant for diseases diagnosis but are very difficult

to align, our program searches for micro inversions in reads that would otherwise be thrown out by current alignment programs. Our interest in detecting micro inversions was inspired by their apparent influence on human diseases.

MID (Micro Inversion Detector), one of the most recent and effective micro inversion detection programs [3], uses a dynamic programming path-finding approach to identify micro inversions with multiple breakpoints within unmapped reads. While other programs have focused on longer micro inversions that are more relevant for taxonomic differentiation [2], MID is optimized to detect read length magnitude micro inversions under 100bp to help remediate unmapped reads. These sized micro inversions are not as well addressed by existing methods, which is why we also chose to focus on them.

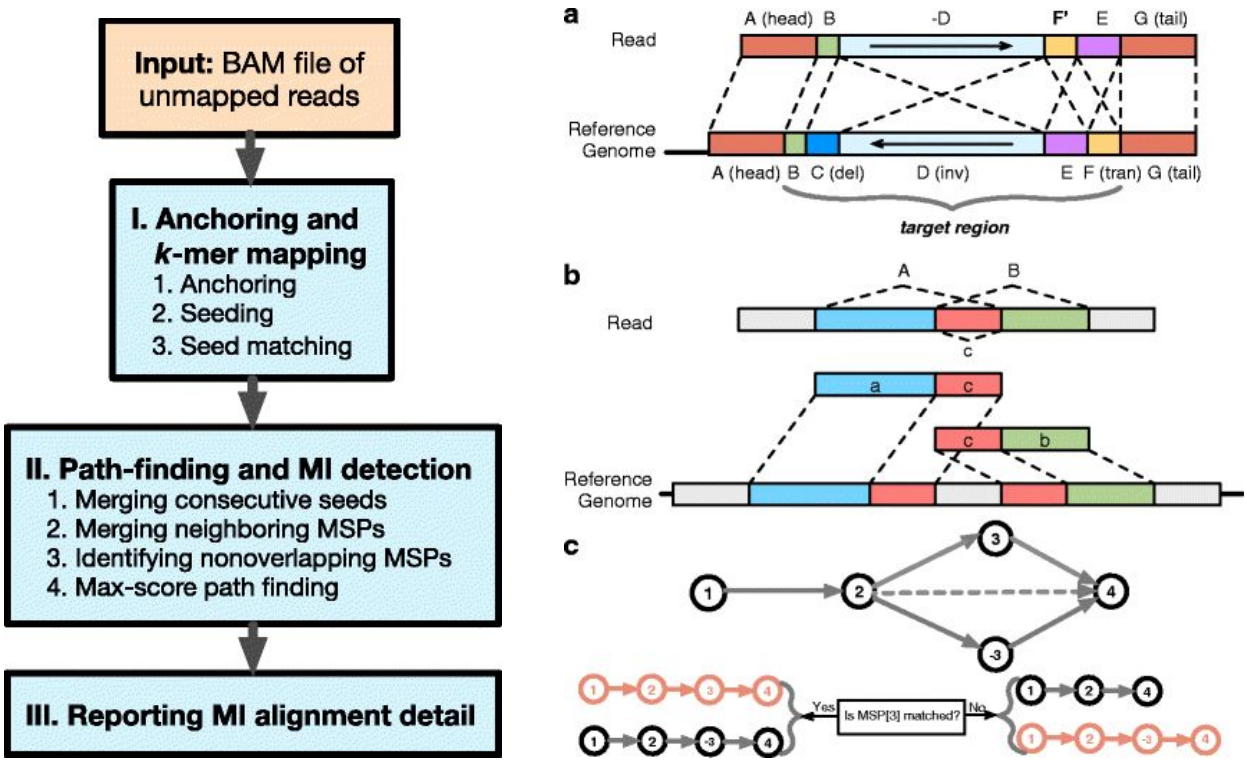


Figure 2 (left): MID pipeline, which first seeks to align seeded read segments, then merge seeds with neighboring aligned sequences, and then calculates alignment scores.

Figure 3 (right): a) Illustration of inverted seed with allowance for neighboring single nucleotide polymorphisms, b) multi-read seed matching c) merging consecutive seeds. [3]

This particular detection program “seeds” each possible k-mer of size 14 in every unmapped read, and then attempts to align reverse complement of the seed with the reference genome allowing for up to two total additional single nucleotide polymorphisms. When seeds align to the reference, they proceed to merge them with neighboring matching segments and then calculate a maximum alignment score for the reads.

Another effort, inspired by recent comparative analysis of mammalian genomes that revealed a greater prevalence of micro inversions than previously estimated, sought to create a method for identifying read length micro inversions sensitive to palindromic and inverted repetitive genetic sequences. [4] They adopted a semi-autonomous approach where they manually verified the detected inversions by analyzing their genomic microarchitectures to check for confounding structural variants and artifacts. There method proved effective for phylogenetic classification, as shown by the reconstructed tree shown below, as well as their claim about the time complexity of the process.

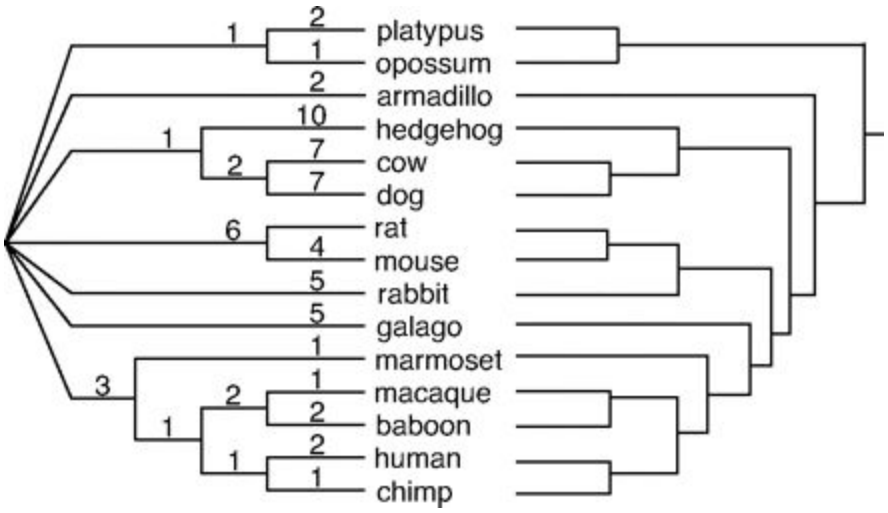


Figure 3: Reconstructed phylogenetic tree on the left, with the canonical evolutionary tree shown on the right.

"If n genomes of length m are produced by independent inversions, then both the correct evolutionary tree (up to the zero-length edges) for these genomes and the ancestral architectures of all its branching vertices may be reconstructed in polynomial time." [4]

Micro inversions clearly play a role in differentiating species and causing significant biological disruptions, so it also makes sense that they are a good research target for understanding and resolving genetic diseases that plague humanity today.

Methods and Software

As mentioned in the previous section, research into shorter into micro inversions (under 100bp) has not been very extensive. This is why we chose to explore this area and create our own micro inversion detection algorithm from the ground up, refining it as we went along.

Preprocessing

```
1. sequence_string = genome sequence from .fa file
2. setup sequence dictionary that maps from nucleotide density -> index
   positions in genome sequence
3. for (position + read_length) to length of sequence_string
4.      chunk = sequence_string[position to position + read_lenth]
5.      count number of each nucleotide in chunk
6.      if(chunk with same nucleotide density exists)
7.           append its index position to that location in the dictionary
8.      else
9.           Create new entry in dictionary
```

Detection

```
1. //Attempt to find a micro inversion in the sequence using a read
2. inversion_length = 100
3. while inversion_length >= minimum_inversion_length
4.      reverse_complement_read = the reverse complement of the read
5.      iterate through inversion_length chunk of read and reverse
   complement chunks until a matching read in the dictionary is found
6.      inversion_length -= 1
```

To summarize the above, we begin by preprocessing the input genome and create a dictionary that maps nucleotide density of chunks of the genome sequence to their index positions in the genome. Reads can then be input and processed. For each read, different length sections are replaced with their reverse complements and then

searched through the dictionary until a valid match is found. Information about the micro inversion is then output by our program.

Our end product is very similar to the algorithm we originally envisioned. We felt that preprocessing the genome sequence using nucleotide density was the best way to approach the issue of micro inversion detection.

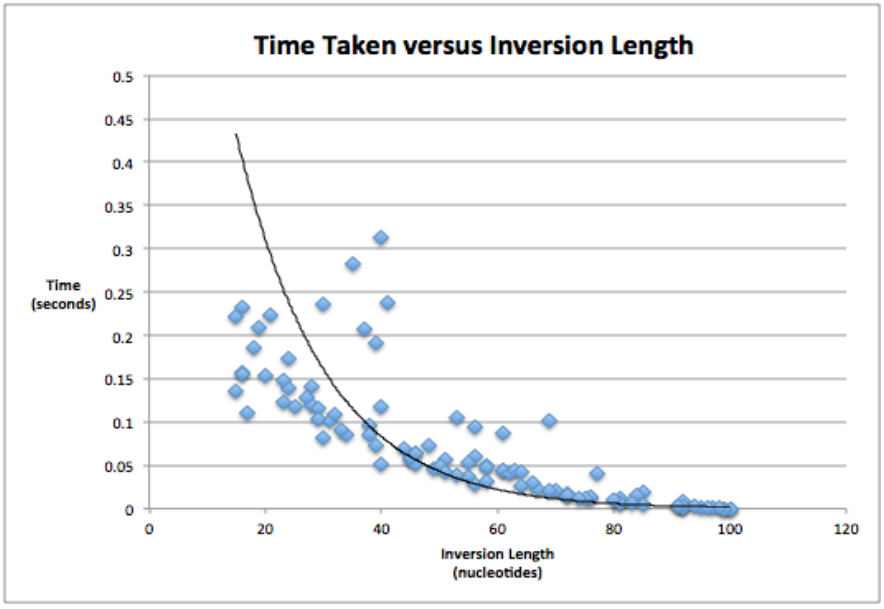
Results

The exact results of the findings can be find inside the git repository in submission/data. The findMicroInversion version of the algorithm correctly finds every micro inversion given to it, regardless of the sequence size. It took 20886 seconds to complete 2000 reads of a 50 million sequence dataset on a Macbook Pro, and 5286 seconds to complete 1000 reads of another 50 million sequence dataset on the Ugrad Servers. Additional timings can be found within the git repository in output files. The most time elapsed when there were smaller inversions because the loops were executed more times. Please see Graph 1.

The findMicroInversion method performs outperformed the AT-GC coupling method, as seen in graph 2. The AT-GC coupling method (inEffBrute.py) made a dictionary with a key of numAT, numGC, and numN instead of the five-tuple since it allowed the reverse complement of each read to only be computed once instead of over 100 like it does in the regular method. As seen in the graph, the coupling method worked well on very small sequence sizes, but it failed to keep its performance up in larger sequence sizes.

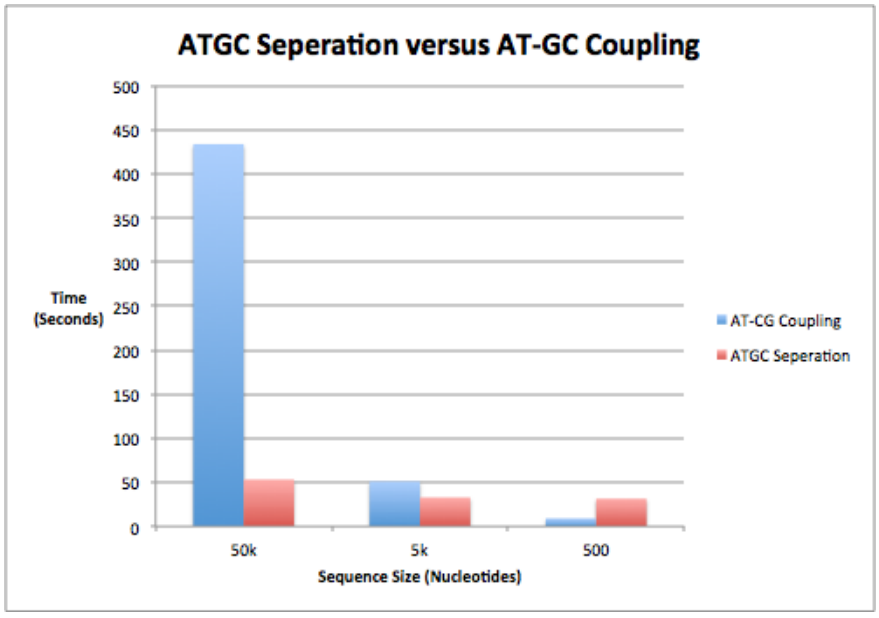
The findMicroInversion method did not outperform the parallelized version of itself (parallelInversions.py) in terms of time, as expected, but it was more accurate than the parallelized version so it was chosen as the final product. See graph 3 for a comparison of all three methods.

Graph 1:



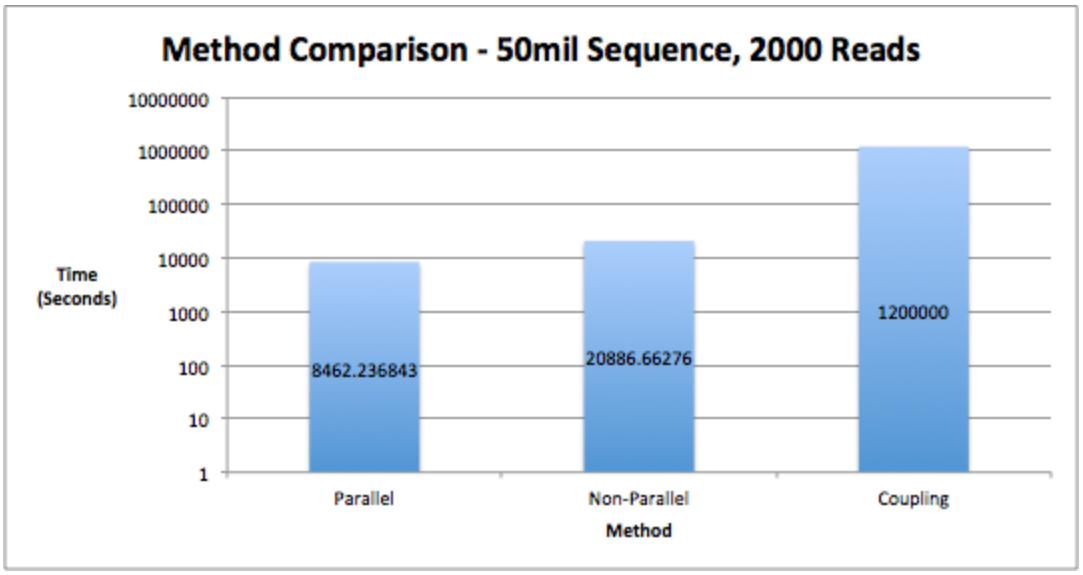
Results from a 100k nucleotides Sequence and 100 reads

Graph 2:



Results with 1000 reads in every run

Graph Three:



Note, Coupling time is a projection, with an estimation of 600 seconds needed per read

Conclusions

While working to try and identify micro inversions within sequencing reads, we found that the challenge of developing an algorithm that would actually find micro inversions was much more difficult than we had anticipated. By trying to “pre-optimize” we realized we backed ourselves into a corner by trying to code a solution that was unnecessarily difficult.

After switching to a more simple brute force algorithm and trying to optimize this with preprocessing, we found that our rate of progress increased drastically. After making many smaller optimizations and tweaks to our brute force algorithm we were able to properly handle reads that had nucleotides that could not be accurately called by the sequencer.

One of the largest challenges we faced was dealing with the substantial runtime of our algorithm on large sets of reads that align to a longer sequence. To overcome

this obstacle, we looked closely at our algorithm and saw that it was “embarrassingly parallelizable.” By letting different cores process sequencing reads in parallel, we got tremendous speed-up in our algorithm. However, the parallelizable method failed to register every every micro inversion and printed strange results for a small fraction that it found, and as such that algorithm was scrapped to return to the non-parallelized version. To see the most improvement on the program in terms of speed, more knowledge on how to correctly parallelize programs is needed. It was determined accuracy was more important than speed since micro inversions are very rare so one wouldn’t need to run the program on large datasets of misaligned reads that contain micro inversions.

All in all, we were pleased with the success and speed of our final running code. If we were to continue to develop this program, we would focus on adding tolerance for aligning reads with micro inversions and additional structural variants within, as well as improving the specificity of our algorithm given our preprocessing strategy, in addition of course to successfully parallelizing it.

Literature Cited

[1] Cancer genome sequencing: a review
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2758710/

[2] Homoplastic microinversions and the avian tree of life:
http://bmcevolbiol.biomedcentral.com/articles/10.1186/1471-2148-11-141

[3] Identifying micro-inversions using high-throughput sequencing reads:
https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-015-2305-7

[4] Microinversions in mammalian evolution:
http://www.pnas.org/content/103/52/19824.full

[5] Gustaf: Detecting and correctly classifying SVs in the NGS twilight zone
http://bioinformatics.oxfordjournals.org/content/early/2014/07/29/bioinformatics.btu431.long

Group Member Contributions

Dan Smillie

- Implemented the parallelization in parallelInversions.py (see submission/alternate methods) from findMicroInversion.py
- Helped in coming up with the original algorithm that involved back-tracing and the partially completed original implementation of the brute force algorithm in brute.py although neither made its way into the final algorithm.
- He also wrote the conclusions section of the final paper.

Oneeb Malik

- Wrote the methods and software section of the final paper.
- Helped with the design of the algorithm, particularly with regards to using nucleotide density for micro inversion detection.
- Wrote a script to run all the required steps for our program.
- Helped test different iterations of the implemented algorithm, both in preprocessing and micro inversion detection.

John Wilson

- Wrote the prior work section of the final paper and conducted related research
- Programmed the inversion generator python script we used to simulate reads from sequence data and insert randomly sized, non-overlapping micro inversions according to adjustable parameters

Joshan Bajaj

- Set up git repository and managed its organization
- Managed delegation of work and meetings
- Wrote generateSequence.py, findMicroInversion.py, preprocessSeq.py, inEffPreprocess.py, ineffBrute.py, as well as many other files to help with data collection and testing python (see ShelvedWork/)
- Developed algorithm with Oneeb and Dan, and made tweaks to it when implemented
- Wrote Abstract, Introduction, Results, and part of the Conclusion