Approximation Algorithms for Genomic Structural Variant Detection

Matt Edwards

December 18, 2009

Abstract

This paper describes algorithmic approaches to detecting structural variations between individual genomes. It reviews a recent approach and examines its approximation accuracy via simulation. It explains the good results of the algorithm based on the structure of the problem and the nature of the experimental data. It shows a simple way to tighten these (empirical) approximation bounds without losing accuracy, and verifies these results experimentally. It shows that a small reformulation of the set cover problem leads to an $O(1)$ approximation algorithm and a higher recall on simulated data. Finally, its runtime results suggest that the standard greedy approach is the most computationally feasible solution.

Introduction

There are many small differences in the genomes between individuals of the same species. Perhaps the most well-known are single base differences (single nucleotide polymorphisms, SNPs), but there are also "structural variants." These are larger-scale events where a particular region of an individual's genome is missing (a deletion), novel (an insertion), or repeated a different number of times (copy-number variation). We will briefly go over the required biology and experimental information, but only to motivate the algorithmic problem that follows - many details will be simplified or eliminated. Reference [3] offers a biological overview of structural variation, containing both experimental approaches and their implications for disease. Reference [8] is a more recent review describing current computational approaches to structural variant detection using the latest sequencing techniques.

We think about identifying structural variation as identifying differences in the genome of a particular individual (here called the sample) from the canonical human reference sequence. The reference sequence is completely (previously) known, and our experiments give us limited information about the sample. Our experiments (high-throughput sequencing) give us pairs of short substrings (reads) that we know to be a given distance apart in the sample genome. Specifically, we might produce millions of pairs of reads, each 36 bases long, where each pair is 200 bases apart in the sample genome. We can locate each read in the reference genome and, for each pair, ask whether its distance in the reference is consistent with its known distance (200 bases) in the sample. If two paired reads are too close in the reference, then the sample has experienced has a small insertion (relative to the reference genome). If two paired reads are far apart in the reference, the sample has experienced a deletion. Figure 1 indicates this graphically.

Difficulty arises because of experimental noise (sequencing errors or variations in paired read distances) and because of the repetitive nature of the human genome. The human genome is not a

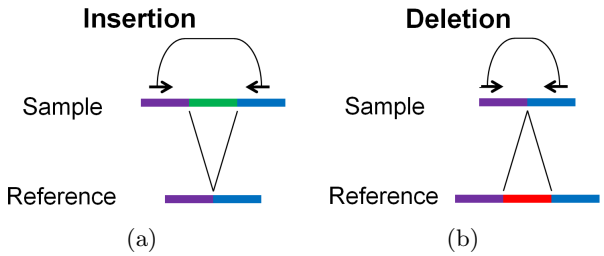


Figure 1: Detecting insertions and deletions with paired end reads: the arrows represent the pair of sequencing reads, with the arc representing their experimental pairing. In (a) the sample has an insertion between the two reads and in (b) the sample has a deletion. The colors are only used to identify contiguous regions of the genome and are not meant to have biological significance. Note that at this point we do not consider reads that span the insertion or deletion.

random sampling of letters (nucleotides), thus there is difficulty in mapping 36 base reads uniquely to the reference genome. For instance, only 75% of simulated 36 base reads mapped to a unique location in the human genome, and only 90% of simulated 72 base reads map uniquely. Note that these numbers can change depending on the assumed or measured accuracy of the sequencing data (lower accuracy means we should tolerate more errors in the mapping process, and therefore will have more possible matches).

We therefore examine computational methods to uncover the set of structural differences between a particular sample genome and the reference, given these experimental data. We will work within the framework of [6], who formulate this reconstruction task as the set cover problem. We implement the greedy approximation algorithm and examine its accuracy on simulated data. We obtain tighter empirical approximation bounds by showing how to preprocess the data without losing many structural variants. We relax some of the constraints of the set cover problem and improve performance. Our simulation results demonstrate the accurate approximation results and computational efficiency of the greedy approach.

Methods

We largely follow the exposition of [6] in describing the structural variant detection problem as set cover. Given many paired reads, we calculate all possible locations (alignments) of each read in the reference genome. Taking all possible locations separates this method from other approaches (see [8]) that employ other heuristics to uncover variation. We take the read pairs whose distances in the reference are incorrect (for all possible alignments) as our interesting dataset that will describe the structural variation. Given a set of paired reads, we wish to find the smallest set of events (insertions and deletions) that can explain the experimental data. As described in [6], this is a maximum parsimony solution and has the equivalent objective of maximizing the average number

of reads supporting each predicted event. The reads compose the elements of the sets and each set is a particular event (insertion or deletion), containing all reads consistent with that event. Note that a read may have different aligned positions, and each possible position could belong to different (or multiple) event sets. We can construct the event sets from the reads greedily in polynomial time by considering all pairs of intervals in the dataset. For each interval, we add the reads that have an alignment consistent with that event (see [6] for details).

Once we have translated the problem into set cover, we can run the standard greedy approximation algorithm that selects the set containing the most uncovered vertices at each iteration ([7]). This yields a $\min(f, 1 + \log d)$ approximation algorithm, where d is the size of the largest set and f is the maximum number of sets to which any vertex belongs ([10]). Furthermore, this approximation is tight under reasonable complexity assumptions ([2]). We can also run an exact integer programming or linear programming relaxation algorithm ([5]).

An extension that will prove useful is what we call "noisy set cover": the set cover problem where only a certain fraction of vertices need to be covered (an existing name for this formulation was not identified in the literature). The natural interpretation in this application is that some of the reads (vertices) are corrupted or otherwise incorrect due to experimental reasons, and therefore should be ignored without penalty. We will show with simulated data that this eliminates spurious deletion or insertion predictions that are supported by only a small number of reads.

More formally, define x-noisy set cover to be the set cover problem where at most x out of n vertices do not have to be covered. Let OPT be the size of the optimal x-noisy set cover and k be the size of the cover constructed by the greedy algorithm that leaves no more than x vertices uncovered.

Theorem: The greedy set cover algorithm approximates x-noisy set cover to within $\log \frac{n}{1+x}$:
Proof: We proceed as in [4]. Using the pigeonhole principle and induction, the number of uncovered items after k greedily-added sets is at most $n(1 - \frac{1}{OPT})^k$. We want this quantity to be less than $1 + x$:

$$n(1 - \frac{1}{OPT})^k < 1 + x$$

$$\exp\left(\frac{-k}{OPT}\right) < \frac{1+x}{n}$$

$$k < OPT \cdot \log \frac{n}{1+x}$$

For $x = 0$ this reduces to set cover and yields the same $\log n$ bound. Note that this relation does not hold for $x = n$ because OPT is then 0. When x is a constant fraction of n (say cn), greedy x-noisy set cover produces an $O(1)$ approximation ($\log \frac{1}{c}$ approximation ratio).

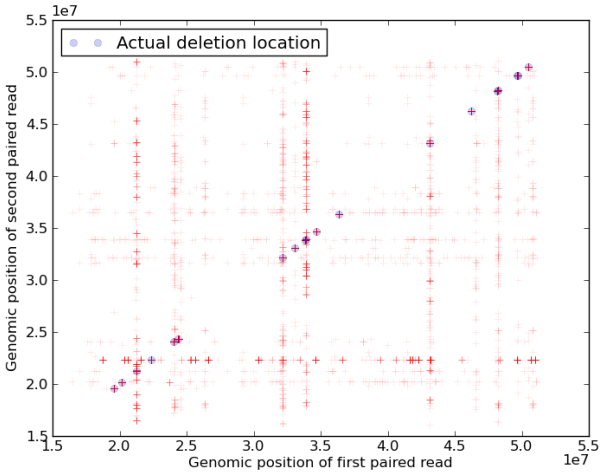


Figure 2: All possible alignments of discordant read pairs. Horizontal and vertical line effects are presumably repeated elements that allow multiple matchings. The blue circles show the locations of the simulated deletions.

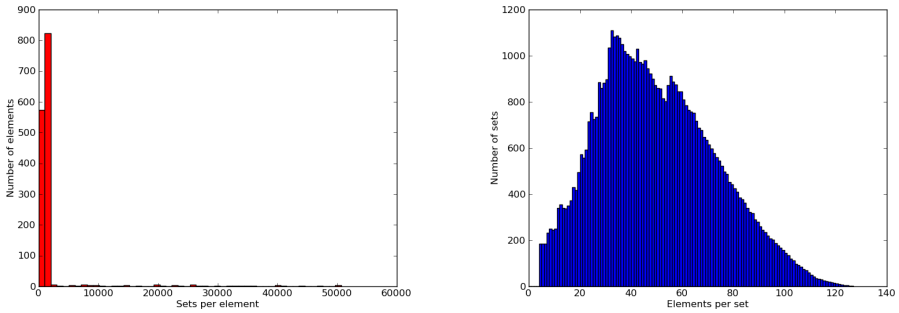
Results

Simulation

Following the evaluation procedures of similar approaches ([9]), we simulated 20 deletions in human chromosome 22 and tested our ability to recover them. The simulated deletions followed the approximate size distribution of [6]. We generated 30-fold coverage paired-end reads of this dataset. We found all possible locations for each read using the tool of [1] and found pairs where both reads map but at inappropriate distances (discordant pairs). Figure 2 shows all possible alignments of paired read. Approximately 1500 of the 40 million reads were discordant. We computed the possible events (deletion ranges) and assigned supporting reads to each possible event. We collapsed into single ranges all deletion ranges that had identical supporting reads. Based on all possible alignments of the discordant read pairs, there are about 60000 possible deletion events. We examined the characteristics of the sets, in order to predict the quality of the greedy approximation algorithm. Figure 3 shows the set membership distribution for all elements as well as the set size distribution.

Algorithms

Given this set cover instance, we can solve it in several ways. Initially, this project hoped to explore several ILP and LP techniques. However, the problem instances are very large: here, 1500 variables and 60000 constraints for only 20 deletions. The problem size will increase further for larger chromosomes, more events, and noisier data. A memory exception was produced when loading



(a) (b)

Figure 3: Set cover instance characteristics from 20 simulated deletions on chromosome 22.

the constraint matrix in MATLAB and the Python interface to the GNU linear programming kit (GLPK), even on 64-bit architectures. It is possible that commercial solvers (CPLEX, MOSEK) would have more success, but these were not accessible during the course of this project.

Tightening approximation bounds

In this instance, we see that most elements belong to a small number of sets, but a few elements belong to very many sets (so f as defined previously is large). These elements are presumably common (perhaps functional) patterns that repeat many times in the genome. We could artificially impose a limit on this quantity, but this would introduce bias around certain repetitive regions. An alternate approach is to consider the number of elements per set. Assuming read pairs are unique (or that duplicate reads are eliminated), we can use the structure of the problem to bound d, the size of the largest set of reads supporting a structural event. If paired reads are exactly L bases apart (previously we set $L = 200$ for concreteness), then there are at most L unique read pairs for each event (sliding the pair over the breakpoint). Our results on unique read pairs, shown in figure 3, match this observation. In experimental reality, L is not a constant but an upper bound: so L^2 bounds the number of unique reads supporting an event. Thus we can have an $O(\log L)$ approximation algorithm with simple preprocessing. As compared to limiting f, this introduces no bias. In any case, $O(\log L)$ is typically much smaller than any available f bound since L is in the hundreds for current experimental techniques.

Accuracy

Using the greedy approximation algorithm, all 20 deletions were successfully recovered (100% recall). We defined a successful recovery as one where the predicted and actual deletions overlapped for more than half of their average length. There were 24 total deletions predicted (83% precision) by the standard greedy set cover algorithm. When we ran the greedy noisy set cover algorithm,

allowing up to 1% of the reads to be uncovered, the spurious deletion predictions were eliminated. Thus the precision was increased to 100%. These results compare very favorably with the simulation results reported in [6] and [9], though the simulation conditions may not have been perfectly matched. For instance, smaller deletions are more difficult to detect, and deletions might have been simulated in more repetitive or unsequenced regions of the genome.

Runtime

The simulation pipeline and approximation algorithms were all implemented in Python and run on 64-bit Linux machines with approximately 3 Ghz CPUs. The greedy approximation algorithm takes less than a second on this instance, even with a fairly naive implementation. If this phase grows to be a bottleneck, more efficient implementations could be developed. In terms of practical impact, the upstream read mapping phase (and associated I/O) takes dramatically longer. As one example, the authors in [6] note that mapping the reads from a high-coverage whole-genome dataset took several days on a medium-sized cluster.

Discussion

This work evaluated a recently proposed technique for detecting genomic structural variation using noisy experimental data. The simulation results validate the set cover modeling framework as well as the use of the standard greedy algorithm. Theoretical and empirical arguments showed why the set cover instances generated by this problem are very closely approximated by the greedy algorithm. Introducing the noisy set cover problem allowed for a natural way to reduce spurious predictions without heuristics.

Future work in this area could take several directions. From a modeling standpoint, there are several simplifications in this work that could be more appropriately handled. First, this approach assumes various structural events occur independently. A large point against this is the fact that some structural events can be mutually exclusive: two predicted deletions cannot be overlapping. Integrating these exclusivity constraints into the set cover framework is one challenge. Second, single reads that span a breakpoint are ignored in this analysis. In the read processing step, we only used pairs where both component reads mapped successfully. Integrating these unmapped reads could localize structural variants to single-base precision. However, handling them in the appropriate way is a challenge because of the very short length of these "half reads." Third, complementary experimental techniques exist to discover structural variation. The described approach could be extended to handle unpaired long reads or other sequence data along with the paired short reads.

Bibliography

[1] Alkan et al. (2009). "Personalized copy number and segmental duplication maps using next-generation sequencing." *Nature Genetics*.

[2] Feige (1998). "A threshold of ln n for approximating set cover." *Journal of the ACM*.

[3] Feuk, Carson, Scherer (2006). "Structural variation in the human genome." *Nature Reviews Genetics*.

[4] Hinkemeyer and Zeleny (2007). "Greedy algorithms lecture notes." `http://pages.cs.wisc.edu/~shuchi/courses/787-F07/scribe-notes/lecture02.pdf`

[5] Hochbaum (1997). "Approximation algorithms for NP-hard problems."

[6] Hormozdiari, Alkan, Eichler, and Sahinalp (2009). "Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes." *Genome Research*.

[7] Lovasz (1975). "On the ratio of optimal integral and fractional covers." *Discrete Mathematics*.

[8] Medvedev, Stanciu, and Brudno (2009). "Computational methods for discovering structural variation with next-generation sequencing." *Nature Methods*.

[9] Ye, Schulz, Long, Apweiler, and Ning (2009). "Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads." *Bioinformatics*.

[10] Vazirani (2001). "Approximation algorithms."