Building compacted de Bruijn graph from 100 human genomes

Ilia Minkin[1] and Paul Medvedev[1]

Department of Computer Science and Engineering, The Pennsylvania State University, USA

1 Introduction

A paragraph stub. Discuss the importance of de Bruijn graphs [1] in assembly [cite assembly applications] and comparative genomics [cite comparative genomics applications].

A paragraph stub. Tell about compressed graph and its advantages [cite paper where it first appeared]. State that it is desirable to avoid construction of an ordinary graph first.

A paragraph stub. Notice that all methods applicable to pan-genome are slow and/or require a lot of memory.

A paragraph stub. "We invented a new algorithm that is parallelizable and requires much smaller memory..." Say a few a words about the idea: based on Bloom Filters, constructs a partially compacted graph first, then filters out false positives.

2 The Basic Algorithm

A paragraph stub. Define ordinary de Bruijn graph [figure needed] for pan-genome. Define the compacted graph [figure needed]; define what a Bifurcation is.

A paragraph stub. Say a few words high-level about Bloom filters: structure, supported operations, etc.

A paragraph stub. Basic observation #1: if a genomic substring S is flanked by a pair of bifurcations; S is an edge in the compacted graph. Note that it is true only for pan-genome case [figures needed].

A paragraph stub. Basic observation #2: suppose that we have a data structure that can list output/input edges of vertex. Given such a structure, it is easy to decide whether a vertex is a bifurcation.

A paragraph stub. If we use Bloom filter as such a structure, we can discover vertices of a partially compacted graph [figure?]

A paragraph stub. We can quickly remove false bifurcations by explicitly exploring edges of candidate bifurcations [figure?].

A paragraph stub. Present a figure with the whole algorithm.

A paragraph stub. Discuss double-strandness: for each copy of a $k+1$-mer store its "canonical" version in a Bloom Filter.

3 Parallelization Scheme

4 Effects of Bloom Filter Size and Parameter Selection

5 Results

A paragraph stub. Overview the experiment design:

1. Comparison with other tools
2. Parallel scalability
3. Round-splitting efficiency

A paragraph stub. Highlight the results of comparison with other tools Notice that Schatz's paper mentioned Sibelia in a totally, absolutely, completely, fully, entirely, perfectly, thoroughly incorrect way.

A paragraph stub. Discuss the results of scalability experiments.

A paragraph stub. Speculate about round-splitting results.

6 Discussion

A paragraph stub. State that the algorithm works well and have the following advantages:

1. Faster than competitors
2. Smaller memory than competitors
3. Parallelization scalability
4. Smooth memory/time tradeoff
5. Simple

Note that experimental results directly support claims 1-4.

A paragraph stub. Discuss possible applicability of partially compacted graphs.

A paragraph stub. Show limitations & drawbacks:

1. Can't be applied to assembly setting
2. Bloom filters are cache inefficient
3. ?

A paragraph stub. Main take-home message: de Bruijn graph for pan-genome are easy to construct, and can form the backbone of sequence genome comparison: reference/variant representation, alignment and synteny blocks construction.

Acknowledgements. Say thanks to Daniel Lemire, author of [2] for his enormous support.

References

1. Bruijn, d.N.: A combinatorial problem. Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen. Series A 49(7), 758 (1946)
2. Lemire, D., Kaser, O.: Recursive n-gram hashing is pairwise independent, at best. Computer Speech & Language 24(4), 698–710 (2010)