Algorithms for Bioinformatics

Compressive Genomics

Ulf Leser

Content of this Lecture

- Next Generation Sequencing
- Sequence compression
- Approximate search in compressed genomes
- Using multiple references

- This lecture is not part of the examination

Large Scale Sequencing Projects

50.000 samples: To obtain a comprehensive description of genomic, transcriptomic and epigenomic changes in 50 different tumor types and/or subtypes which are of clinical and societal importance across the globe.

Genomics England ... is creating a lasting legacy for patients, the NHS and the UK economy through the sequencing of 100,000 genomes: the 100,000 Genomes Project.

The Veterans Affairs (VA) Office of Research and Development is launching the Million Veteran Program (MVP) .... The goal of MVP is to better understand how genes affect health and illness in order to improve health care.

Next Generation Sequencing

- New generation of sequencers since ~2005
  - Illumina, Solexa, 454, Solid, ...
- Much higher throughput
  - ~15 TB raw data in 3-5 days
  - ~600 GB processed data/week
  - Cost for sequencing a genome down to ~2.000 USD
- 3rd generation sequencers
  - Single molecule sequencing
  - A (human) genome in a day
  - Sequence every human
 - Sequence different cells in every human


Illumina HiSeq 2000. DNAVision

Latest

- 600GB / day, 18.000 genomes per year
- $1,000 genome at 30x coverage
  - Amortized over 18,000 genomes per year over four-year period
- (Not cheap)

Data Tsunami
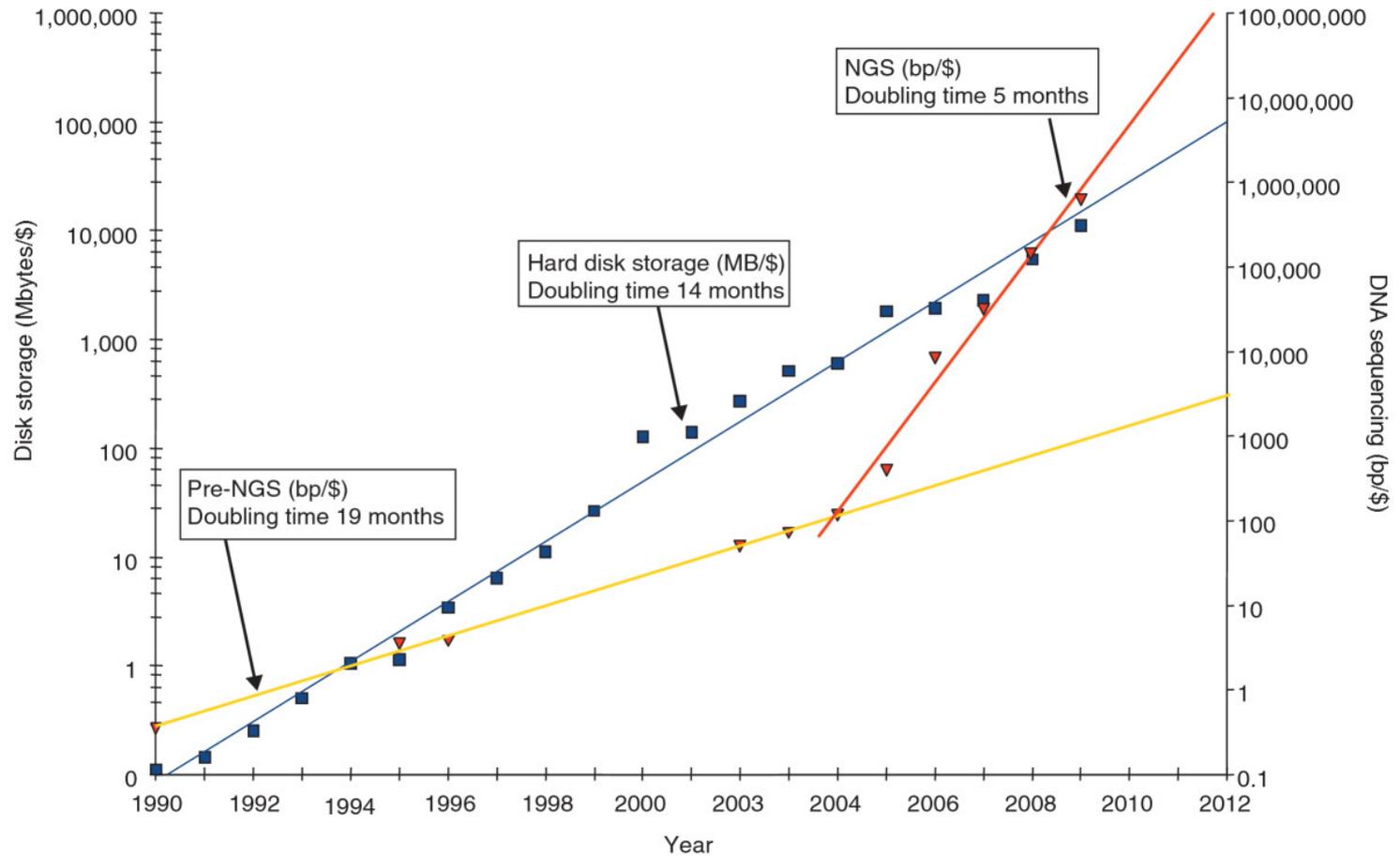
Stein, L. D. (2010). *Genome Biol*

The „real" Cost of Genomic Sequencing
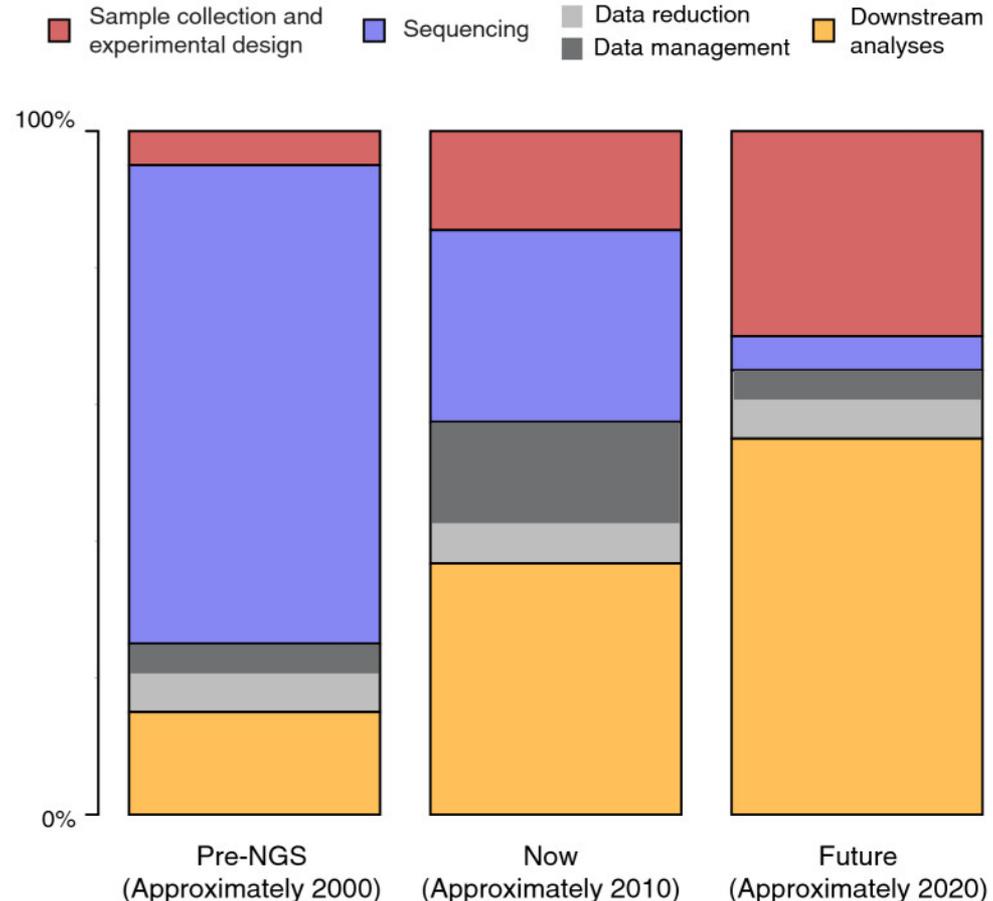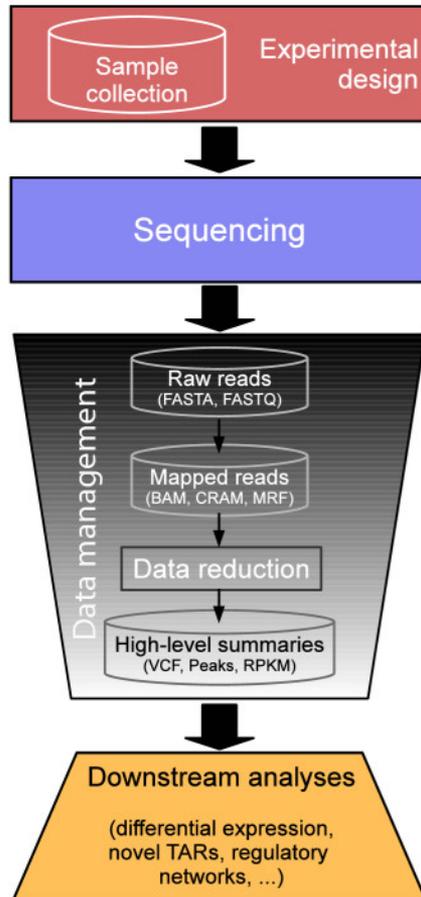
Sboner, A. (2011). The real cost of sequencing: higher than you think! Genome Biology 2011

New Problems

- Need to process huge amounts of data with complex pipelines
  - Terabytes per week
  - Pipelines with dozens of steps
- Need to store huge amounts of sequence data
  - (Hundreds of) thousands of genomes

Content of this Lecture

- Next Generation Sequencing
- Sequence compression
  - Referential compression
  - Four issues
- Approximate search in compressed genomes
- Using multiple references

Compressing Genomes

- Four basic techniques (lossless)
  - Bit packing
  - Statistical compression
  - Dictionary-based
  - Referential compression
- Criteria for compression methods
  - Compression ratio
  - Compression speed / decompression speed
  - Analyzing (searching) compressed data
- Compressing reads is another topic
  - Quality information, non-standard bases, short strings, …
- Another big topic: Lossy compression

1. Bit Packing

- A genome consists of 4 different bases
- Representing one bases thus requires 2 bits only
- One byte – four bases
- Compression ratio (compared to ASCII / FASTA): 1:4
- Advantages: Fast, universal, simple
- Disadvantage: Low compression ratio

2. Statistical Compression

- Idea: Don't use the same number of bits for every char
- Frequent characters are represented with less bits
- Useful for larger alphabets with large differences in character frequencies
- Can be extended to q-grams
- But: DNA q-gram are roughly equally frequent
- Disadvantage:
  Low compression ratio (~1:5)

Statistical Compression

0: 66%   1: 34%

A

10: 16%   11: 16%

G

110: 8%   111: 8%

C   T

Uncompressed sequence

| A | A | A | G | G | C | A | A | A | A | T | A | ... |

Compressed sequence

| 0 | 0 | 0 | 10 | 10 | 110 | 0 | 0 | 0 | 0 | 111 | 0 | ... |
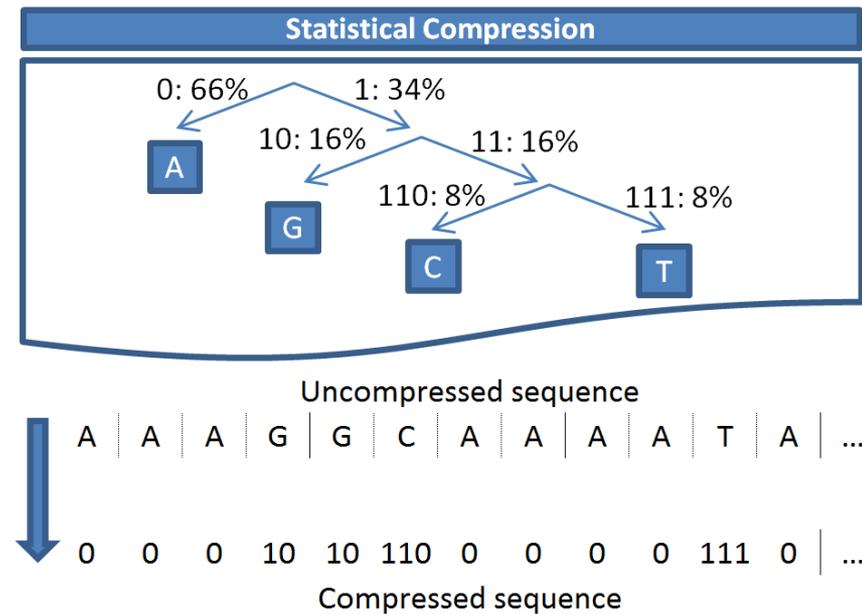
3. Dictionary-based Compression

- Idea: Represent frequent substrings with short codes
- Ziv-Lempel-Welch: Find most frequent substrings online
  - Stored in a dictionary
  - Index in dictionary is used as code
    - Can be compressed
  - Trade-Off: Dictionary-size, compression speed, compression ratio

- Useful when large diffs in frequency of substrings exist
  - Recurring patterns: Images, language, tables, ...
- Disadvantage: Low compression ratio (for DNA, ~1:4-6)
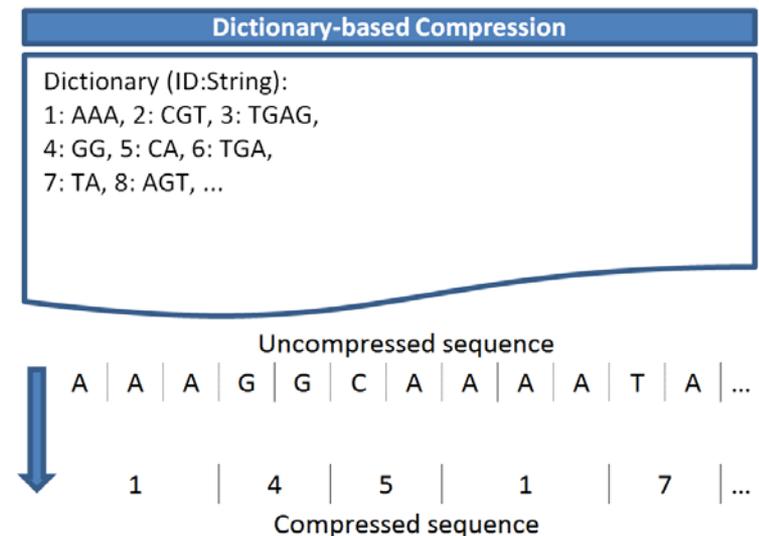
4. Referential Compression

- For NGS data, we usually know the reference genome
- Idea: Use reference as (external, predefined) dictionary
- Sequences are represented as lists of referential match entries: (start, length, mismatch)
- Issues
  - Find long matches fast
    - Trade-off: Long matches: ratio++; faster compression: ratio—
  - Efficient coding of RMEs
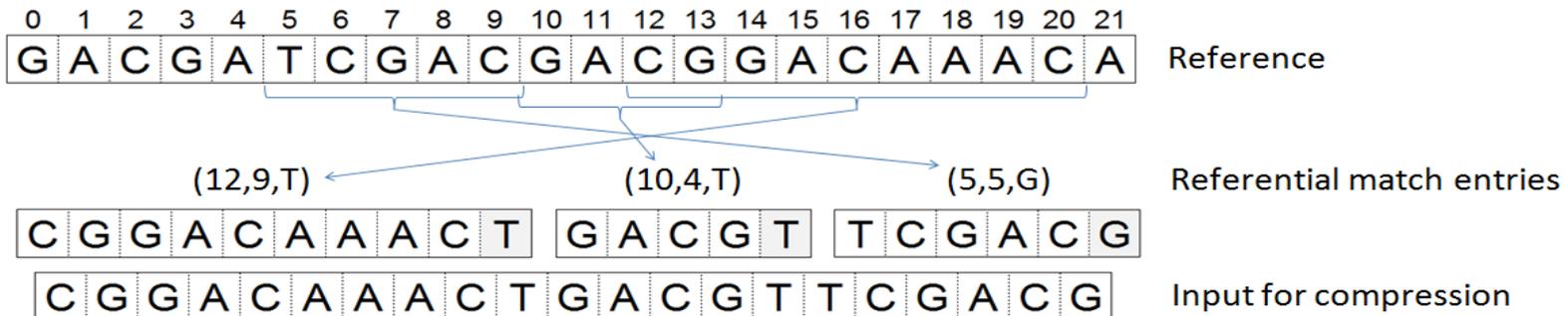
Greedy Algorithm

Algorithm 1 Referential Compression Algorithm

Input: to-be-compressed string s and reference string ref
Output: referential compression $result$ of s with respect to ref

1: Let $result$ be an empty collection
2: **while** $|s| \neq 0$ **do**
3:     Let pre be the longest prefix of s occurring in ref, and let i be a position of an occurrence of pre in ref
4:     Add $\langle i, |pre|, s(|pre|) \rangle$ to the end of $result$
5:     Remove the first $|pre| + 1$ symbols from s
6: **end while**

- Compression rate for human chromosomes: ~1:60
- Compression speed for human chromosomes : 80 MB/s
- Main memory usage: ~4*size(ref)+size(s)
  - Using DNA-optimized compressed suffix trees for reference

Content of this Lecture

- Next Generation Sequencing
- Sequence compression
  - Referential compression
  - Four issues in referential compression
- Approximate search in compressed genomes
- Using multiple references

Issues

- Compact encoding of RMEs
- Main memory usage
- Faster compression / decompression
- Which reference?

- General: Balancing the trade-off between compression ratio and compression speed

1. Encoding RME′s

- Very frequent: Series of consecutive matches with SNVs in between

Rare!

(1000,5,A), (1006,12,C), (1019,4,A), (1024,20,C), (1045,8,B), (9453,25,C), …

- Improvement: Delta encoding (with/out default stepsize)

(1000,5,A), (1006,12,C), (1019,4,A), (1024,20,C), (1045,8,B), (9453,25,C), …
(1000,5,A), (+6,12,C),   (+13,4,A), (+5,20,C),   (+21,8,B),   (9453,25,C), …
(1000,5,A), (+0,12,C),   (+0,4,A),   (+0,20,C),   (+0,8,B),   (9453,25,C), …

- Large impact on compression ratio

2. Improving Main Memory Usage

- Best (compressed) suffix tree libraries need ~3-4*n space
- Observation: We often find matches in sequential regions


- Can be exploited to save main memory
  - Partition reference and input into blocks (e.g. 5MB)
  - Keep one (indexed) block each at a time in main memory
  - Search other reference blocks only when no good match is found
- Switching blocks is costly: Avoid
  - Even if this means less optimal compression
  - Typical: Threshold on minimal length of RMEs; otherwise switch

# Memory / compression speed / compression ratio

- Evaluation for human chromosome 1
  - Small blocks: Frequent block changes, bad ratio
  - Blocks larger than ~100MB: No further improvements
  - Compression/decompression requires only ~500MB for dictionary
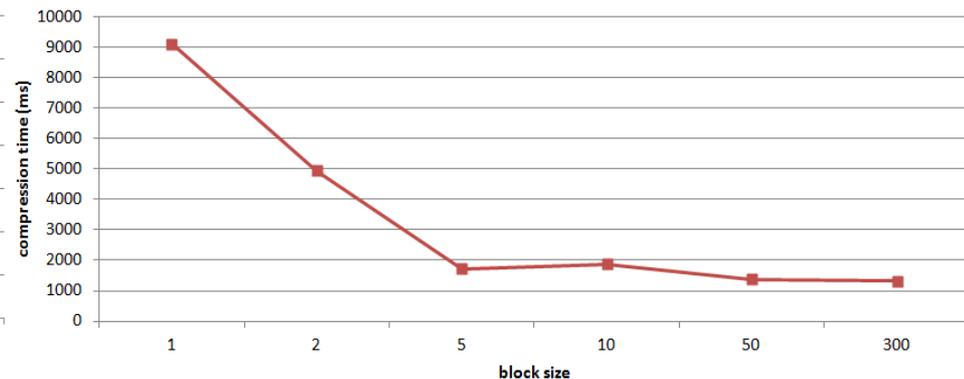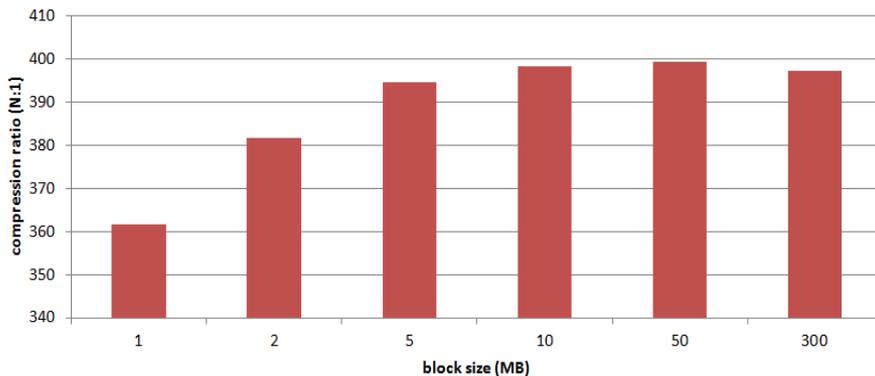
3. Improving compression speed

- Runtime dominated by looking up prefixes in the compressed suffix tree (CST++)
  - Decoding the compressed suffix tree structure costs time
  - Maximal throughput: ~50.000 lookups / sec
- Improvement: Local matching
  - Search next RME near previous RME directly in the reference
    - Ignoring the index
  - Accept best next match iff RME sufficiently long
  - Speed-up by a factor of ~5-10
- Also improves compression ratio
  - Next matches close to previous ones – effective delta encoding
  - But may not find longest RME
  - Evaluation: Overall space reduction

Results: Ratio (Data: 1000 Genomes project)
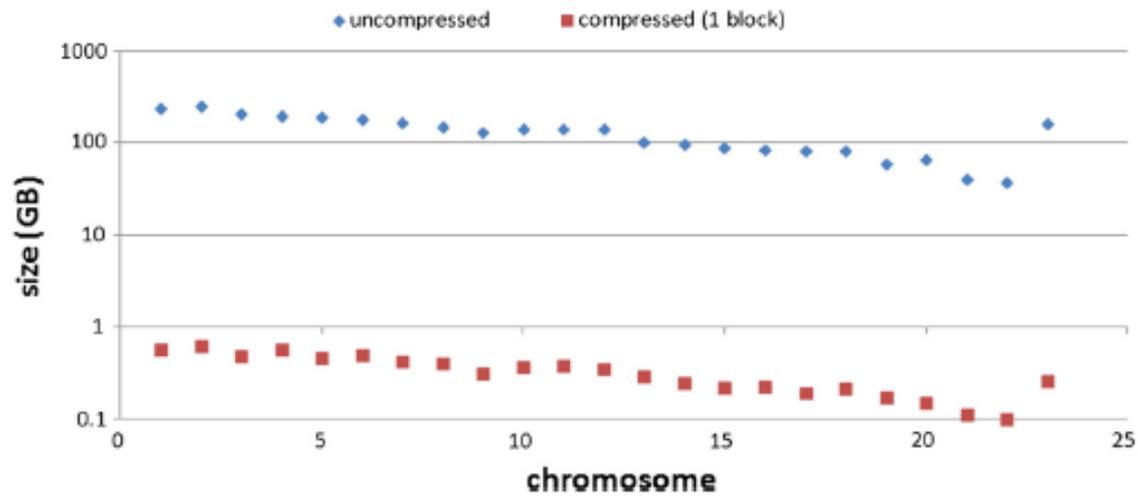
Overall compression ratio: ~1:400

Results: Speed

Dataset	Compressed size (in MB)			Runtime (in s)			Compression factor			Compression speed (MB/s)		
	GDC	RLZ	FRESCO	GDC	RLZ	FRESCO	GDC	RLZ	FRESCO	GDC	RLZ	FRESCO
H-1	3.7	15.5	4.2	495.2	224.0	20.0	680.0	160.8	590.6	5.0	11.1	124.3
H-2	3.9	15.9	4.5	454.9	199.4	19.4	625.5	152.9	542.8	5.3	12.2	125.5
H-3	3.3	13.4	3.8	314.6	165.5	14.9	593.6	147.5	513.9	6.3	11.9	132.4
H-4	3.5	13.8	4.1	247.0	159.4	15.0	543.8	138.4	466.1	7.7	12.0	127.1
H-5	3.0	12.0	3.4	243.4	144.0	13.9	608.2	150.6	526.3	7.4	12.6	130.2
H-6	3.0	11.9	3.6	248.0	143.8	15.3	566.1	143.7	475.1	6.9	11.9	112.0
H-7	2.7	10.7	3.1	403.1	121.1	12.8	591.2	148.7	508.8	3.9	13.1	124.7
H-8	2.5	10.1	2.9	171.8	122.9	11.6	577.5	144.8	500.5	8.5	11.9	126.3
H-9	2.0	8.4	2.3	130.0	102.2	11.0	714.3	168.0	618.2	10.9	13.8	128.8
H-10	2.4	9.4	2.7	183.6	109.8	10.9	572.2	144.1	493.4	7.4	12.3	124.7
H-11	2.5	9.6	2.8	153.6	118.3	11.0	548.3	140.5	474.3	8.8	11.4	122.2
H-12	2.3	8.9	2.6	199.2	113.5	10.0	593.0	150.4	514.1	6.7	11.8	133.5
H-13	1.9	7.5	2.2	65.5	90.9	9.2	602.5	153.4	532.2	17.6	12.7	124.5
H-14	1.6	6.4	1.8	68.5	77.0	8.6	664.7	167.6	591.1	15.7	13.9	124.2
H-15	1.4	5.9	1.6	72.2	70.7	8.1	710.1	173.7	636.9	14.2	14.5	126.9
H-16	1.4	5.4	1.6	103.1	68.9	6.9	638.5	167.1	552.5	8.8	13.1	131.4
H-17	1.3	5.1	1.5	140.3	68.9	6.5	635.3	159.1	552.8	5.8	11.8	125.4
H-18	1.4	4.8	1.6	44.6	66.7	6.6	565.2	162.5	487.0	17.5	11.7	118.3
H-19	1.1	4.0	1.3	116.8	50.8	5.3	546.7	147.8	468.0	5.1	11.6	111.1
H-20	1.0	4.0	1.2	43.8	49.5	4.5	623.7	157.4	542.5	14.4	12.7	139.3
H-21	0.7	2.8	0.9	12.3	33.3	3.5	684.3	171.8	553.0	39.1	14.5	138.2
H-22	0.6	2.7	0.7	19.3	32.0	3.7	816.9	189.7	735.9	26.5	16.0	137.1
H-X	1.7	7.7	2.0	168.2	96.3	12.1	903.6	201.6	789.0	9.2	16.1	128.0
AT-1	2.0	6.5	2.3	8.3	41.3	2.5	154.2	105.3	133.2	36.7	7.4	123.1
AT-2	1.4	4.5	1.7	4.2	25.4	1.4	145.0	98.5	119.0	46.9	7.8	136.8
AT-3	1.7	5.5	2.0	5.5	32.1	1.6	139.8	96.0	117.2	42.7	7.3	145.1
AT-4	1.3	4.3	1.6	3.7	24.4	1.5	139.5	97.2	116.7	50.2	7.6	126.5
AT-5	1.9	6.1	2.2	6.3	37.5	1.9	144.6	99.5	121.3	42.8	7.2	141.7
Y-WG	1.0	86.8	1.4	2.8	47.6	1.0	127.3	1.4	89.0	44.5	2.6	124.7
AVG	2.0	10.7	2.3	142.4	90.9	8.6	532.9	142.8	460.7	18.0	11.5	128.0

Fig. 3: Compression statistics for 10 random sequences against a fixed reference (best values bold).

4. Which reference to use?

- Given a set of genomes: Which should be the reference?
- Similarity to reference is key to high compression rates
  - Compressing Human against Mouse: Disaster
  - Similarity in non-coding region is low
- Exhaustive reference selection is very time consuming (took 6 days for 1092 * 1091 H-22)
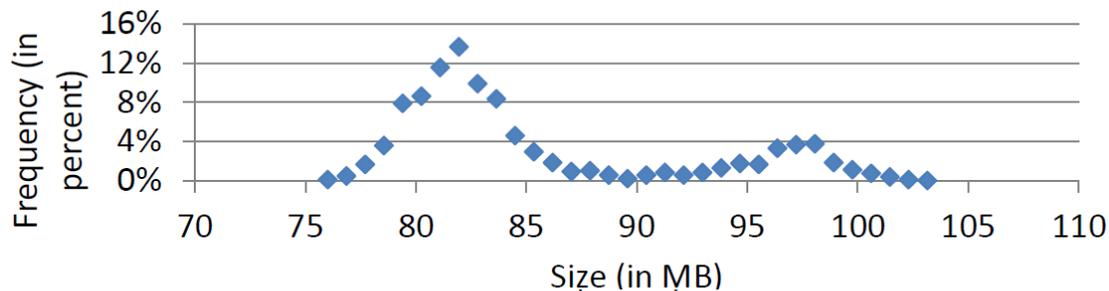
Fig. : Distribution of total storage requirements for all against all for H-22.

Two Alternatives

- Heuristic-based reference selection
  - Define a heuristic for the similarity of two sequences
    - For instance: Compute best reference based on small sample
    - Use any other fast similarity estimation method
  - Pick the sequence most similar to all other sequences according to this heuristic

- Better: Build your own reference
  - Reference rewriting
  - Given a reference, rewrite it in order to obtain higher compression ratios

Selection versus Rewriting

Dataset	C. factor	Total time (s)	C. speed (MB/s)	C. factor increase
H-1	637.5	3,581.1	76.0	+7.3%
H-2	578.6	3,207.5	82.8	+5.5%
H-3	557.0	2,663.8	81.0	+6.2%
H-4	519.2	2,616.5	79.7	+8.7%
H-5	547.5	2,392.4	82.5	+0.3%
H-6	512.9	2,585.0	72.2	+3.3%
H-7	536.1	2,251.9	77.2	+3.8%
H-8	527.0	1,944.2	82.2	+2.7%
H-9	636.7	1,822.0	84.6	+1.4%
H-10	528.6	1,862.3	79.4	+4.4%
H-11	547.0	1,823.0	80.8	+11.2%
H-12	550.4	1,738.6	84.0	+5.7%
H-13	630.1	1,454.6	86.4	+14.8%
H-14	651.3	1,394.4	84.0	+7.6%
H-15	681.4	1,317.4	85.0	+5.5%
H-16	558.9	1,262.7	78.1	-1.3%
H-17	607.4	1,153.5	76.8	+7.6%
H-18	542.9	1,055.1	80.7	+9.9%
H-19	498.1	991.1	65.1	+0.8%
H-20	571.7	766.5	89.7	+3.6%
H-21	663.3	594.4	88.4	+12.8%
H-22	736.0	645.8	86.6	+3.1%
H-X	859.5	2,028.5	83.6	+8.7%
AT-1	138.4	112.2	48.8	+4.3%
AT-2	129.3	61.4	57.8	+7.8%
AT-3	120.8	70.9	59.6	0.0%
AT-4	120.8	60.5	55.3	+1.5%
AT-5	125.1	81.1	59.8	-0.3%
Y-WG	91.9	22.4	21.1	0.0%
AVG	496.7	1,433.1	74.8	+5.1%

Fig.: Compression statistics for selecting references

Dataset	C. factor	Total time (s)	C. speed (MB/s)	C. factor increase
H-1	804.3	3,334.8	81.6	+35.4%
H-2	736.4	3,033.3	87.5	+34.2%
H-3	697.6	2,520.7	85.6	+33.0%
H-4	651.0	2,340.8	89.1	+36.3%
H-5	704.9	2,138.6	92.3	+29.1%
H-6	643.7	2,311.6	80.8	+29.6%
H-7	675.1	1,994.4	87.1	+30.7%
H-8	674.3	1,737.2	92.0	+31.4%
H-9	834.1	1,612.7	95.5	+32.8%
H-10	676.1	1,655.1	89.4	+33.5%
H-11	673.7	1,659.9	88.8	+36.9%
H-12	698.2	1,586.9	92.1	+34.0%
H-13	765.9	1,350.8	93.0	+39.5%
H-14	806.1	1,266.1	92.5	+33.2%
H-15	864.1	1,190.6	94.0	+33.8%
H-16	753.6	1,024.3	96.2	+33.1%
H-17	729.8	1,030.2	86.0	+29.3%
H-18	671.2	946.6	90.0	+35.9%
H-19	619.8	846.5	76.2	+25.5%
H-20	703.1	670.3	102.6	+27.5%
H-21	769.0	508.2	103.4	+30.8%
H-22	904.5	548.3	102.0	+26.8%
H-X	1,018.0	1,993.8	85.0	+28.8%
AT-1	132.7	104.7	52.3	0.0%
AT-2	119.9	56.6	62.6	0.0%
AT-3	120.9	65.8	64.2	+0.1%
AT-4	119.0	56.1	59.6	0.0%
AT-5	125.5	75.8	64.1	0.0%
Y-WG	91.9	22.0	21.5	0.0%
AVG	613.3	1,299.4	83.0	+25.6%

Fig.: Compression statistics for rewriting references

Selection versus Rewriting

Dataset	C. factor	Total time (s)	C. speed (MB/s	C. factor increase
H-1	637.5	3,581.1	76.0	+7.3%
H-2	578.6	3,207.5	82.8	+5.5%
H-3	557.0	2,663.8	81.0	+6.2%
H-4	519.2	2,616.5	79.7	+8.7%
H-5	547.5	2,392.4	82.5	+0.3%
H-6	512.9	2,585.0	72.2	+3.3%
H-7	536.1	2,251.9	77.2	+3.8%
H-8	527.0	1,944.2	82.2	+2.7%
H-9	636.7	1,822.0	84.6	+1.4%
H-10	528.6	1,862.3	79.4	+4.4%
H-11	547.0	1,823.0	80.8	+11.2%
H-12	550.4	1,738.6	84.0	+5.7%
H-13	630.1	1,454.6	86.4	+14.8%
H-14	651.3	1,394.4	84.0	+7.6%
H-15	681.4	1,317.4	85.0	+5.5%
H-16	558.9	1,262.7	78.1	-1.3%
H-17	607.4	1,153.5	76.8	+7.6%
H-18	542.9	1,055.1	80.7	+9.9%
H-19	498.1	991.1	65.1	+0.8%
H-20	571.7	766.5	89.7	+3.6%
H-21	663.3	594.4	88.4	+12.8%
H-22	736.0	645.8	86.6	+3.1%
H-X	859.5	2,028.5	83.6	+8.7%
AT-1	138.4	112.2	48.8	+4.3%
AT-2	129.3	61.4	57.8	+7.8%
AT-3	120.8	70.9	59.6	0.0%
AT-4	120.8	60.5	55.3	+1.5%
AT-5	125.1	81.1	59.8	-0.3%
Y-WG	91.9	22.4	21.1	0.0%
AVG	496.7	1,433.1	74.8	+5.1%

Fig. : Compression statistics for selecting references

Dataset	C. factor	Total time (s)	C. speed (MB/s	C. factor increase
H-1	804.3	3,334.8	81.6	+35.4%
H-2	736.4	3,033.3	87.5	+34.2%
H-3	697.6	2,520.7	85.6	+33.0%
H-4	651.0	2,340.8	89.1	+36.3%
H-5	704.9	2,138.6	92.3	+29.1%
H-6	643.7	2,311.6	80.8	+29.6%
H-7	675.1	1,994.4	87.1	+30.7%
H-8	674.3	1,737.2	92.0	+31.4%
H-9	834.1	1,612.7	95.5	+32.8%
H-10	676.1	1,655.1	89.4	+33.5%
H-11	673.7	1,659.9	88.8	+36.9%
H-12	698.2	1,586.9	92.1	+34.0%
H-13	765.9	1,350.8	93.0	+39.5%
H-14	806.1	1,266.1	92.5	+33.2%
H-15	864.1	1,190.6	94.0	+33.8%
H-16	753.6	1,024.3	96.2	+33.1%
H-17	729.8	1,030.2	86.0	+29.3%
H-18	671.2	946.6	90.0	+35.9%
H-19	619.8	846.5	76.2	+25.5%
H-20	703.1	670.3	102.6	+27.5%
H-21	769.0	508.2	103.4	+30.8%
H-22	904.5	548.3	102.0	+26.8%
H-X	1,018.0	1,993.8	85.0	+28.8%
AT-1	132.7	104.7	52.3	0.0%
AT-2	119.9	56.6	62.6	0.0%
AT-3	120.9	65.8	64.2	+0.1%
AT-4	119.0	56.1	59.6	0.0%
AT-5	125.5	75.8	64.1	0.0%
Y-WG	91.9	22.0	21.5	0.0%
AVG	613.3	1,299.4	83.0	+25.6%

Fig. : Compression statistics for rewriting references

Fresco: Comparative Evaluation

	GDC		RLZ		FRESCO		FRESCO (reference selection)		FRESCO (reference rewriting)		FRESCO (second-order compression)	
	CF	C.Speed	CF	C.Speed	CF	C.Speed	CF	C.Speed	CF	C.Speed	CF	C.Speed
H-*	635.0	11.2	158.4	12.8	550.7	126.8	594.7	81.2	742.4	90.6	3,057.2	58.4
AT-*	144.6	43.9	99.3	7.5	121.5	134.5	126.9	56.3	123.6	60.6	407.7	53.7
Y-WG	127.3	44.5	1.4	2.6	89.0	124.7	89.0	21.1	91.9	21.5	712.8	41.4
AVERAGE	302.3	33.2	86.4	7.6	253.7	128.7	270.2	52.8	319.3	57.5	1,392.6	51.1

Fig. 12: Summary of all techniques (CF=compression factor, C.speed=compression speed in MB/s)

- Second Order Compression: Compress RME sets
  - All sequences are similar to each other
  - Thus, different sequences produce very similar RME lists
  - Idea: Compress (using "meta" referential compression)
- Best algorithms today [Deorowisc 2015, GDC-2]
  - Compression ratio 1:9500
  - 7TB FASTA compressed to 700MB
  - Speed: 200MB/sec (beware: measured on different hardware)

Content of this Lecture

- Next Generation Sequencing
- Sequence compression
  - Referential compression
  - Four issues
- Approximate search in compressed genomes
- Using multiple references

K-Approximate Matching (k-difference Mathcing)

- Given a collection of (referentially compressed) sequences S, find all k-approximate matches of a query q
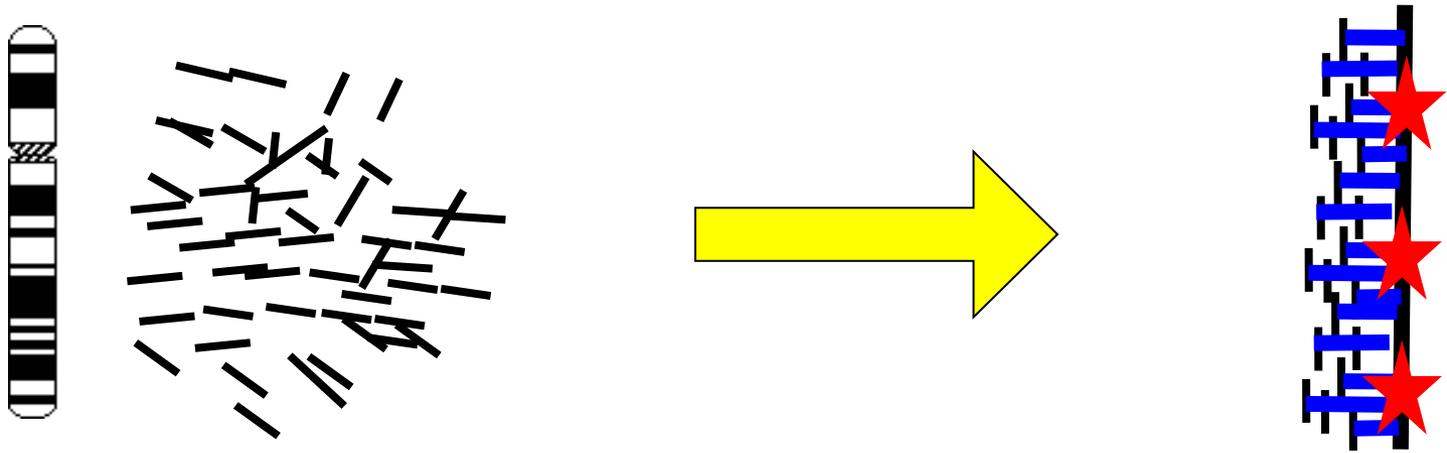
Example Application: Translational Medicine

- Modern cancer drugs depend on genotype of patients

- Genotype: Mutations in certain cancer genes

- Clinics sequence thousands of human genomes

- Given a set of patient genomes C with known outcome and the sequence of a cancer gene g in a new patient q – what is the most similar occurrence of g in C?
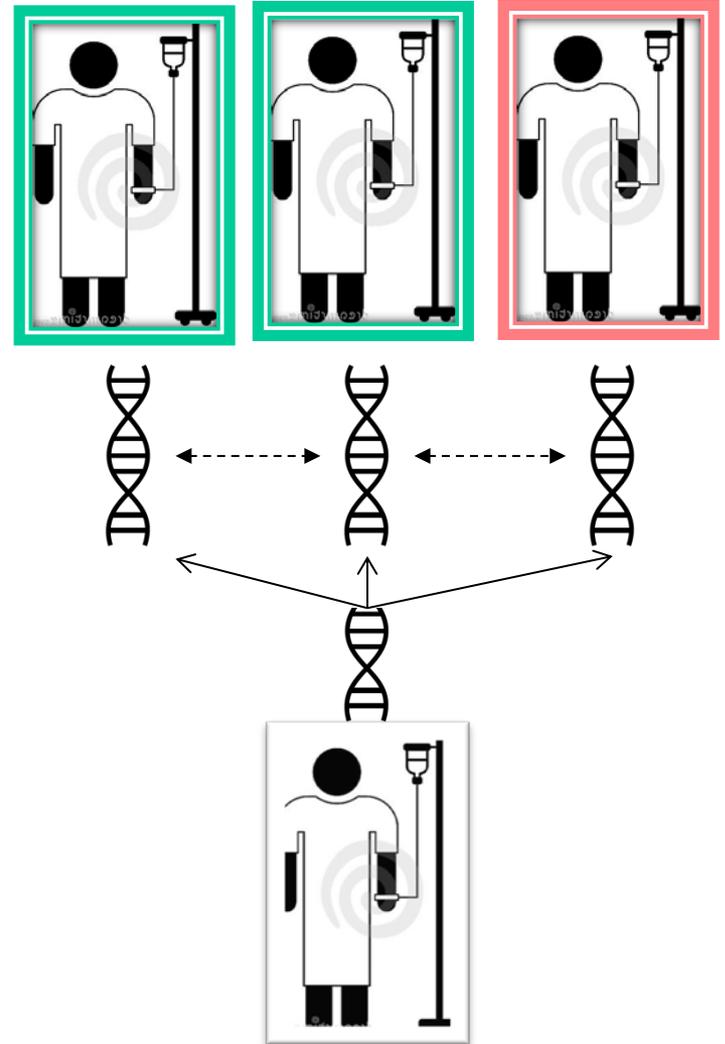
Example Application: Translational Medicine

- Modern cancer drugs depend on genotype of patients
- Genotype: Mutations in certain cancer genes
- Clinics sequence thousands of human genomes
- Given a set of patient genomes C with known outcome and the sequence of a cancer gene g in a new patient q – what is the most similar occurrence of g in C?
- 1000 genomes -> ~3TB data
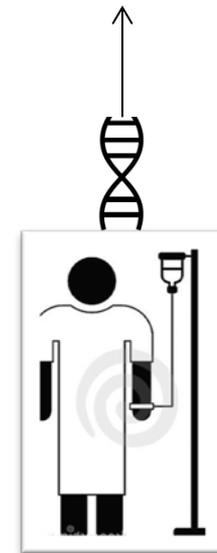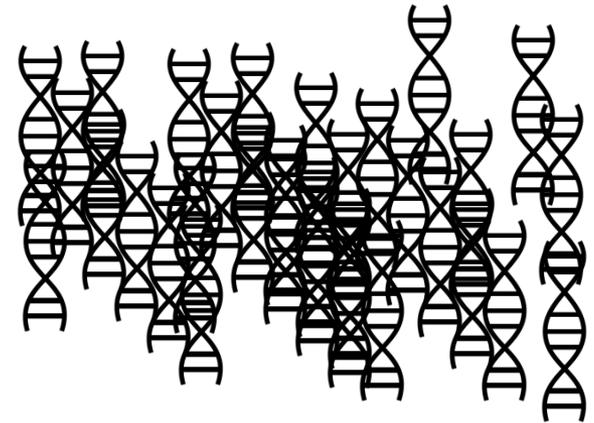
Storing Similar Strings

- Popular idea: Referential Compression
  - Choose a reference string p from C
  - When adding a new string s, only store differences between s and p
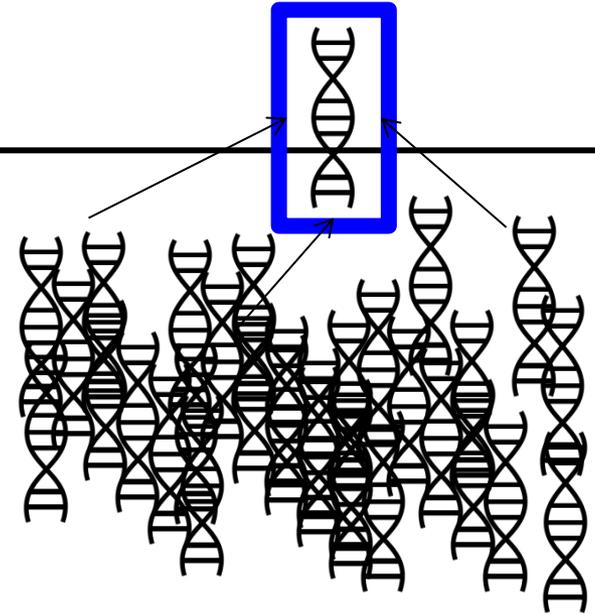
```
          0123456789012345678
p:        Kohala Coast-Hawaii
  s₂:       Kohala Cost
 s₃:      Koala Coast/Hawaii
```

```
    s₂: (p,0,9,s),(p,11,1,_)
s₃: (p,0,2,a),(p,4,8,/),(p,13,6,_)
```

RME: Referential Match Entries

RCSI: Approx. matching in ref. compressed genomes

- Key idea: Find matches in all compressed sequences simultaneously by searching the reference
  - Store reference as suffix tree
  - Search using standard BYP-algorithm
  - For every match, find all RME completely containing the match
    - Build an interval tree over all RMEs
    - If RME X contains match, only children of X may contain other matches

- Problem: Matches not contained in the reference
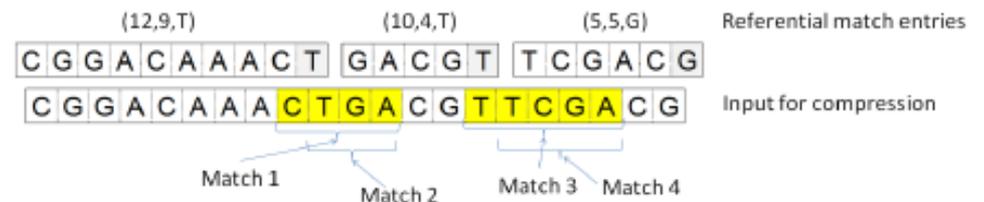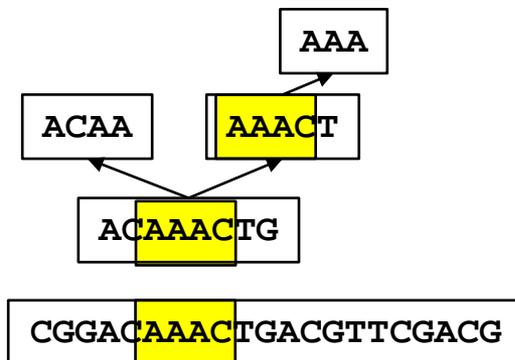
RCSI Approach

- Fix maximal query length q_{max} and maximal k_{max}
- Compute overlap sequences
  - One for every mismatch leading to two consecutive RMEs
- How long must these overlaps be?
  - Answer: $2*|q_{max} + k_{max}|$
  - Very conservative estimation, guaranteed to not loose any match
- Set of overlap sequences is indexed as well
- This index is searched using BYP
  - Additional to reference

Overlap: 2*(4+2-1)+1 symbols

C A A A C **T** G A C G T

C G G A C A A A C **T** G A C G T T C G A C G

(4+2-1) symbols   (4+2-1) symbols
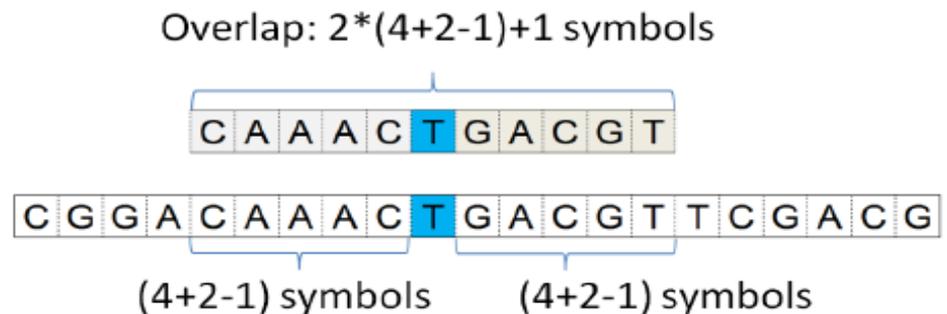
RCSI: Architecture
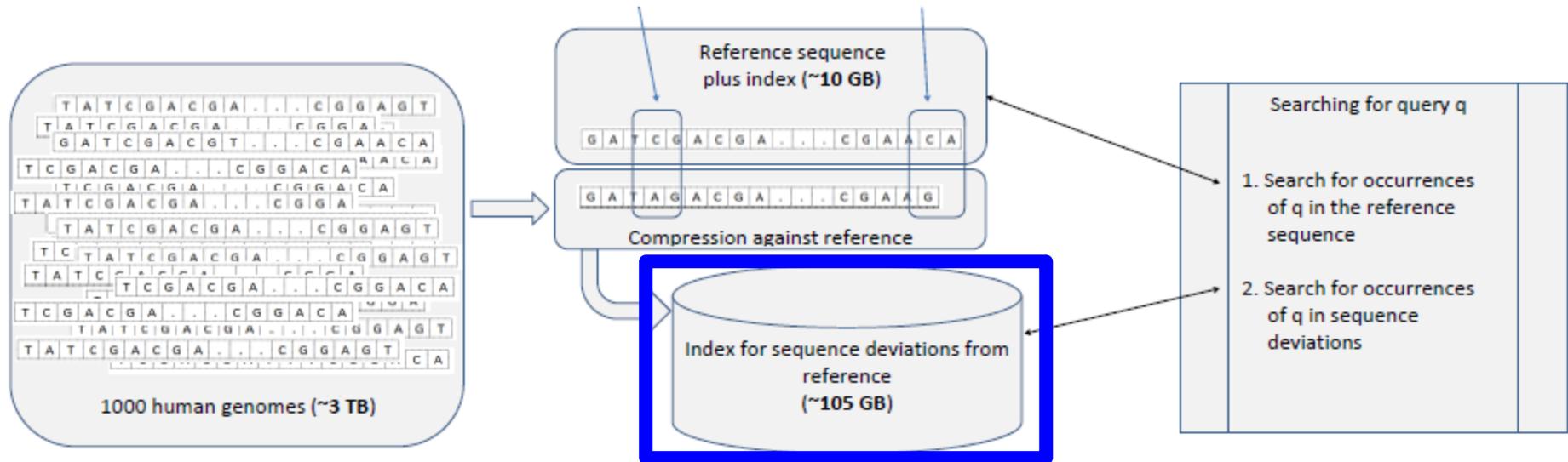
Figure 1: Overview of our Referentially Compressed Search Index.

Evaluation: Indexing time
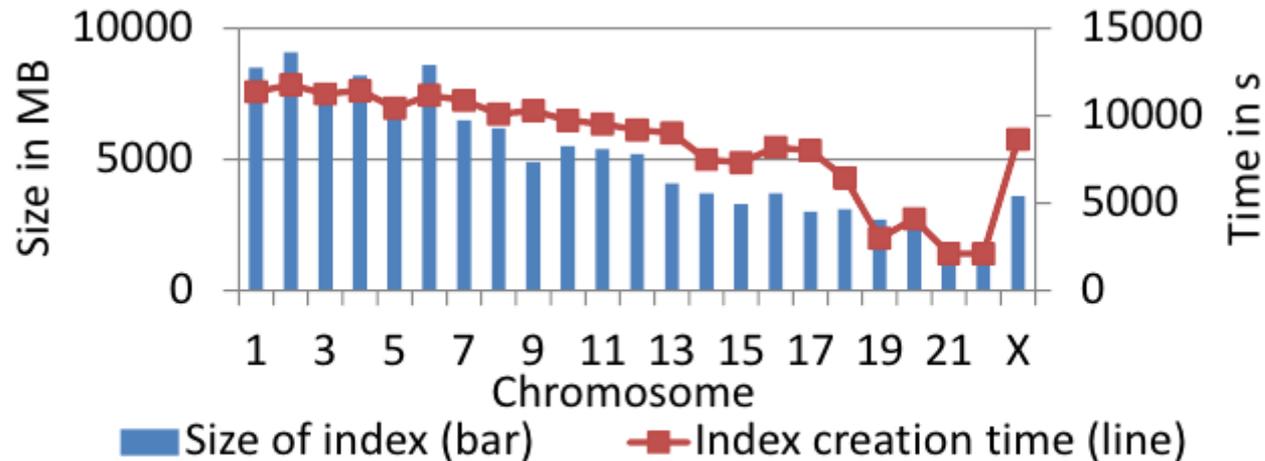
Figure: Size and creation time of RCSI per chromosome.

- Indexing one genome: ~30 sec
- Indexing 1000 genomes: ~8 hours

# Evaluation: Approx. search in 1000 genomes

$(|q| \in [120,170])$
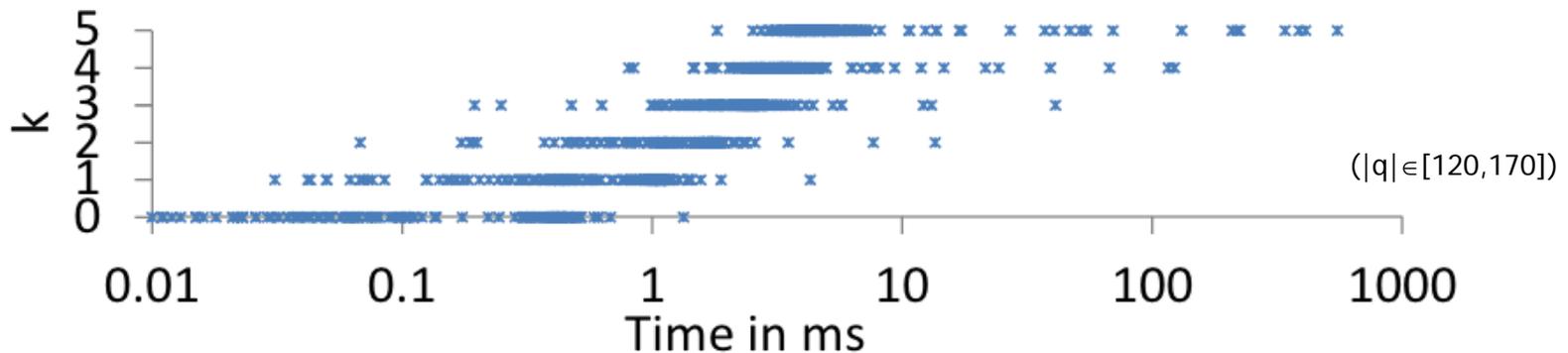
- Until k=5, almost all queries finish in <10ms
- For k=1, almost all queries finish in <1ms
- Outliers: Queries from repetitive regions

Competitors
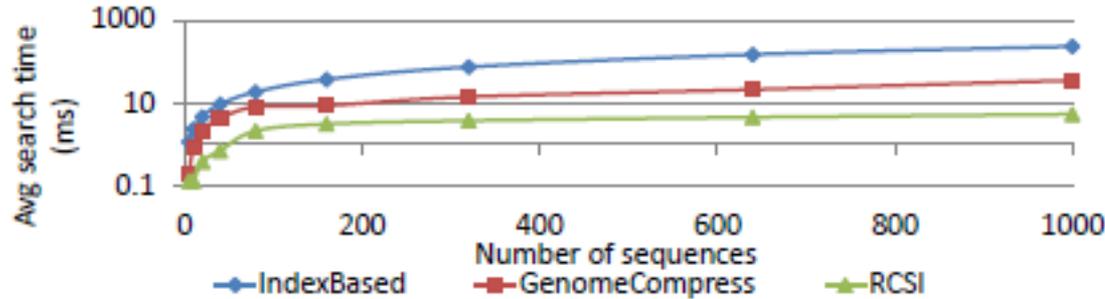
(b) 3-approximate search.
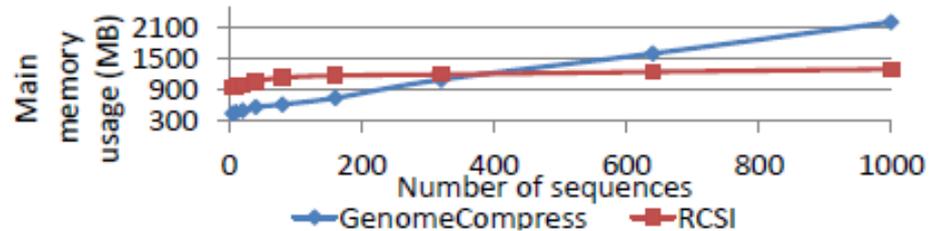
- GC open source code lacks important preprocessing step
  - We could only compare using the data from GC paper
- RCSI between 10 and 100 times faster
  - And computes all results

Content of this Lecture

- Next Generation Sequencing
- Sequence compression
  - Referential compression
  - Four issues
- Approximate search in compressed genomes
- Using multiple references

Collections of Similar Strings

- Often (not always): Strings are similar to each other
  - All human genomes are 99% identical
  - All mammal genomes are >90% identical
  - All elements of a Wikipedia revision histories are highly similar
  - Elements of version histories are very similar (SVN, subversion, …)
  - …

Heterogeneous String Collections

```
     p:        Kohala Coast-Hawaii
        s₂:          Kohala Cost
  s₃:        Koala Coast/Hawaii Islands
  s₄:        Kohala Coast-Hawaii Islands
        s₅:          Orchid Island
        s₆:          Orchied Island
```

Kohala Coast-Hawaii

„compressed against"

Kohala Cost

Kohala Coast/Hawaii Islands

Koala Coast/Hawaii Islands

# Heterogeneous String Collections

```
       p:         Kohala Coast-Hawaii
            s₂:        Kohala Cost
   s₃:      Koala Coast/Hawaii Islands
   s₄:      Kohala Coast-Hawaii Islands
            s₅:        Orchid Island
            s₆:        Orchied Island
```
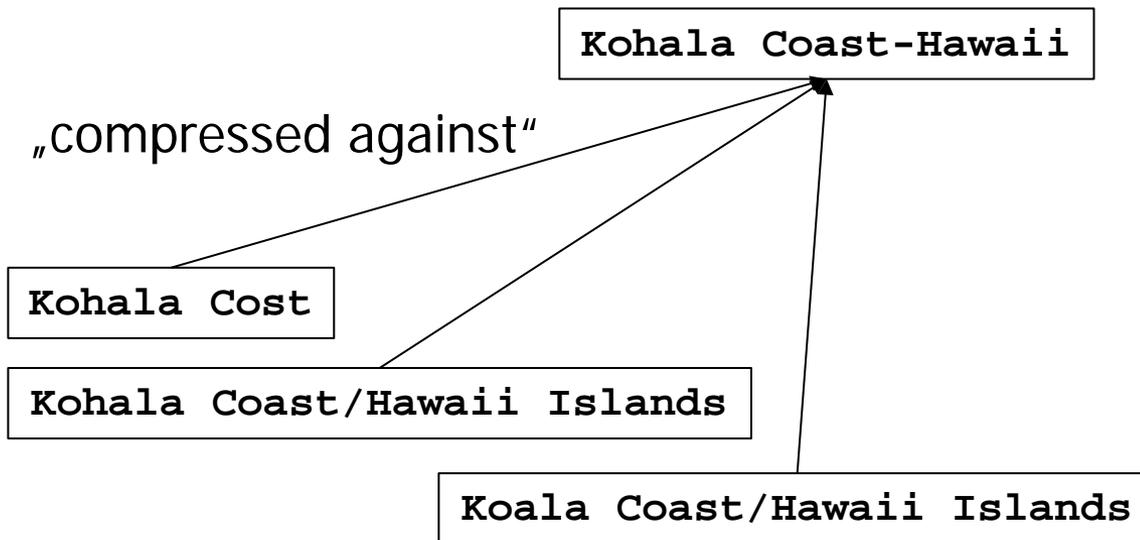
Kohala Coast-Hawaii

Kohala Cost

Kohala Coast/Hawaii Islands

Koala Coast/Hawaii Islands

Orchid Island

Orchied Island
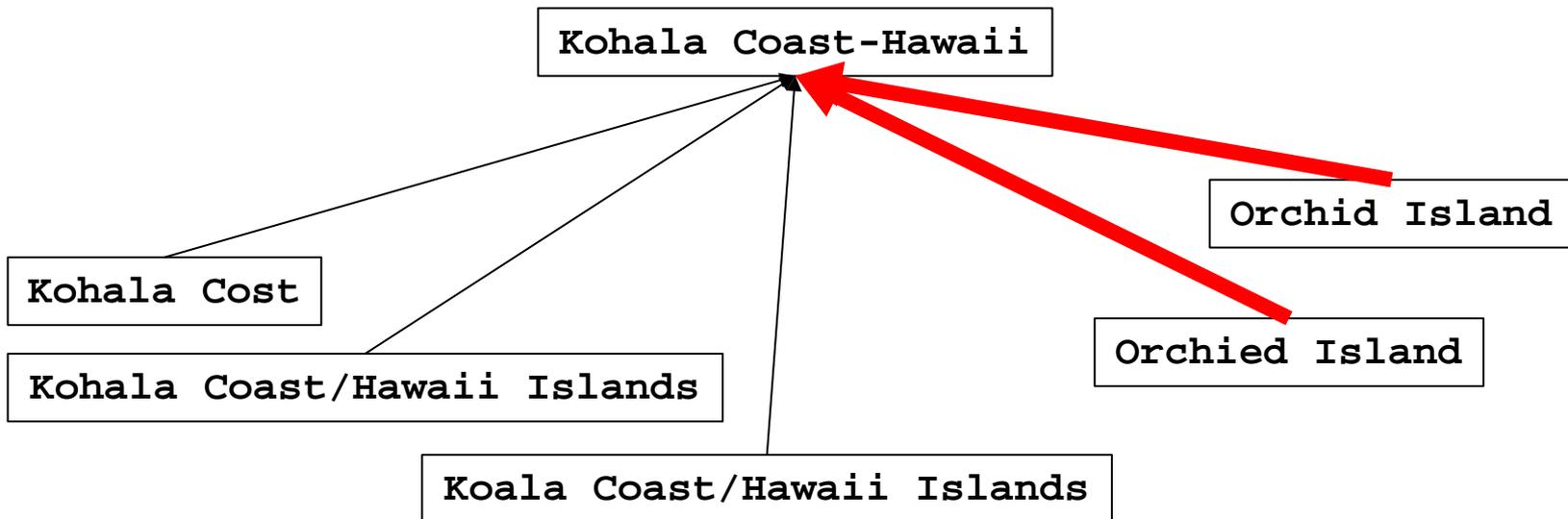
Novel Idea: Use Multiple References

Strings are compressed against different references

Challenge: Which are the best references?
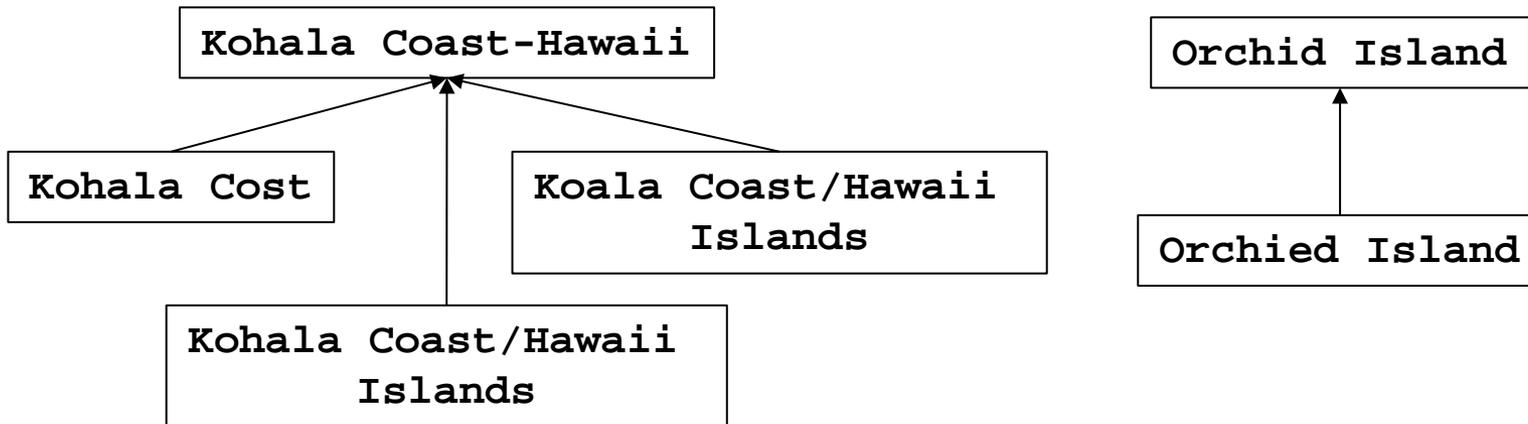
Novel Idea: Allow Hierarchical Compressions

Compression dependencies can form hierarchies

Challenge: Which is the best parent?
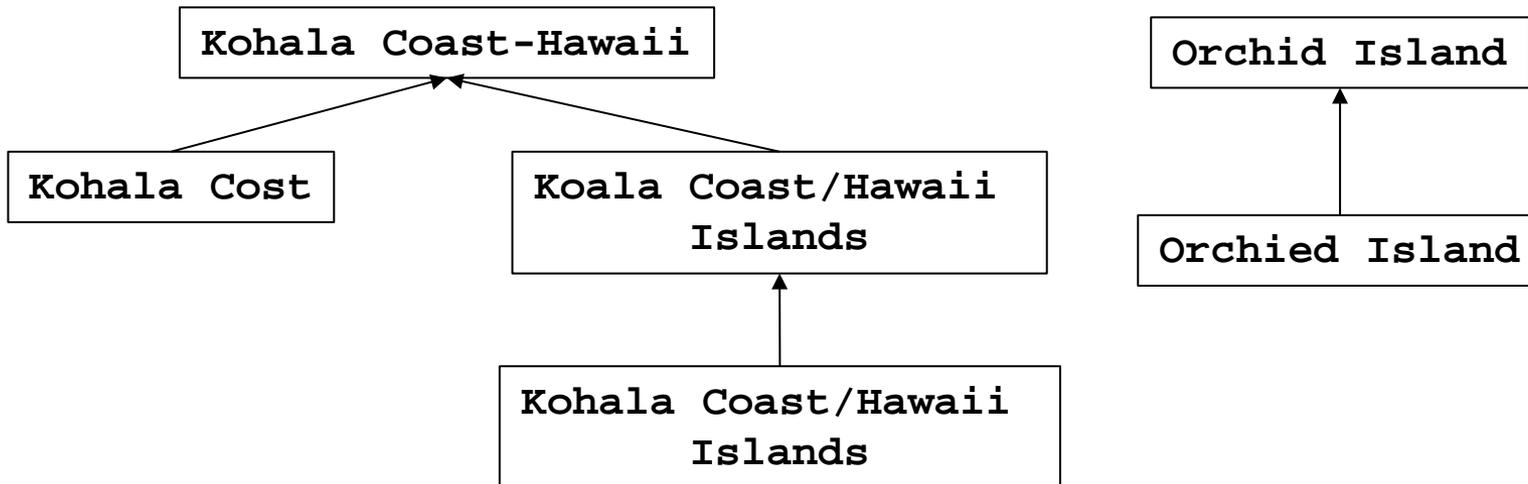
Novel Idea: Compress against Multiple References

Strings are compressed against multiple other strings

Challenge: Which is the best set of parents?
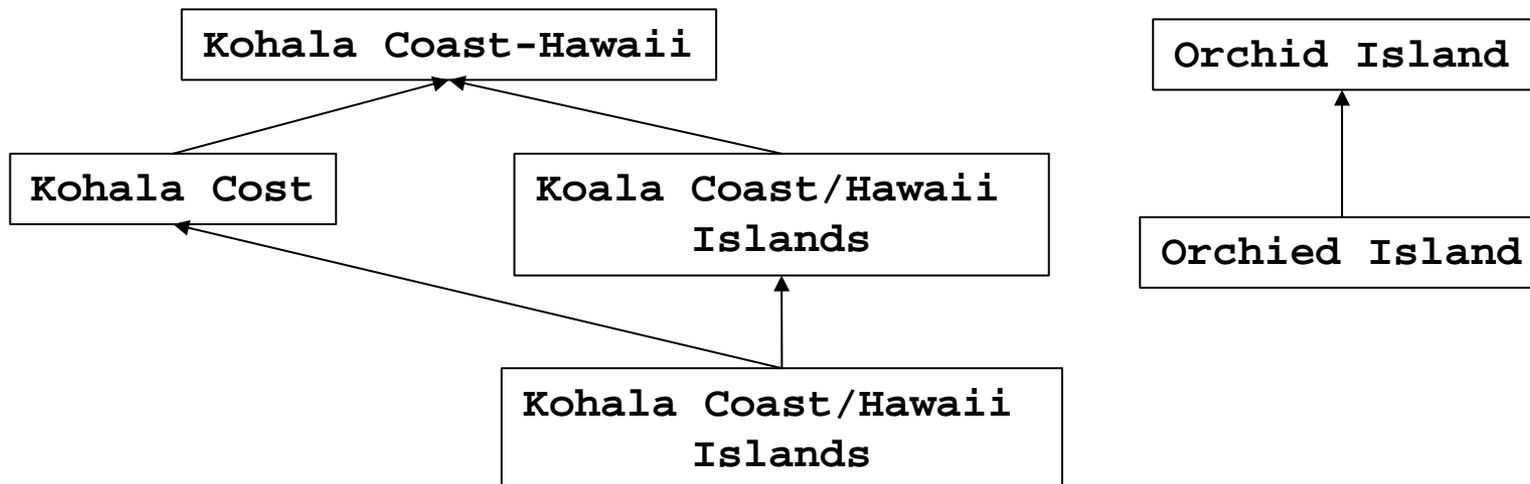
MRSCI: Multiple References Compression

- Challenges during compression
  - Which strings should be references – and how many?
  - How can we efficiently find good parents?
  - What is the optimal compression hierarchy?
- How to perform k-approximate search in a multi-reference compression hierarchy?
- Findings
  - Proof that finding an optimal compression hierarchy is NP-hard
  - Three heuristics to build increasingly complex CHs
    - Increasingly better compression rates
    - Moderate increase in indexing time, roughly same search speed

Overview
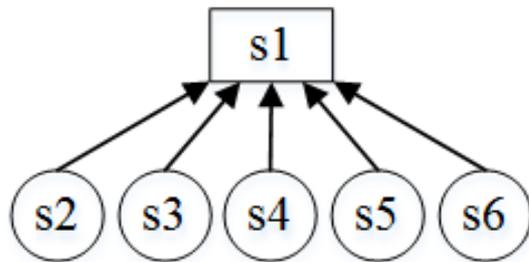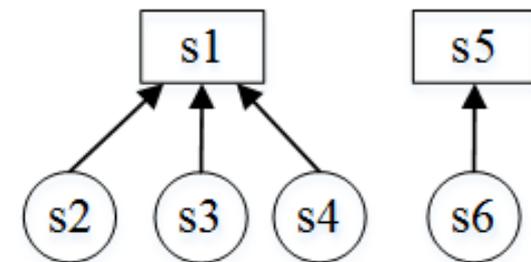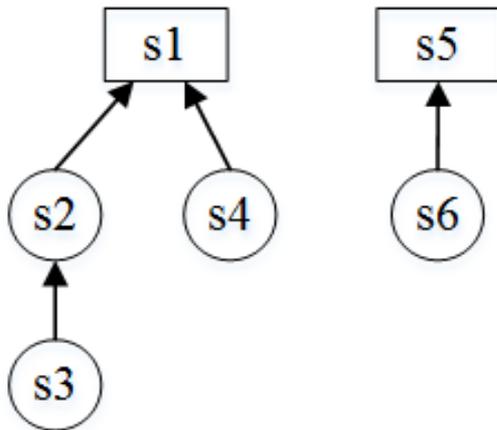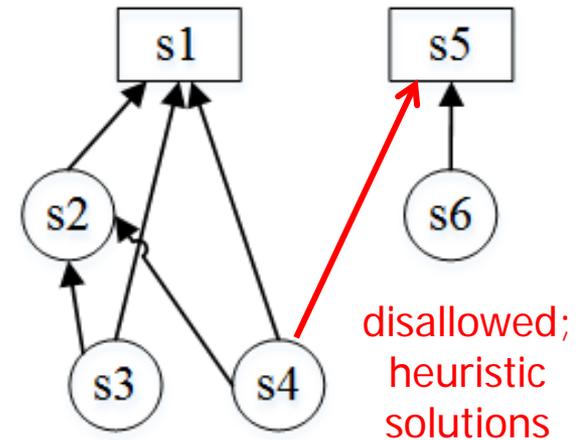
CPart: Using Multiple References
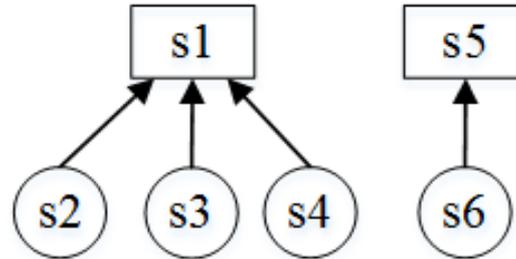
- Iteratively and greedily compress strings from C
  - Choose first string as first reference p, set P={p}
  - Compress all other strings s one-by-one
    - Find reference p′ from P "most similar" to s
    - If p′ and s are sufficiently similar – compress s against p′
    - If not, add s to P (new reference, new root)
- Needs fast method for assessing string similarity
- Essentially performs a greedy clustering of C

Competitors

- Sweet spot: Strong and fast compression, fast search
- Two classes of competitors
  - Pure indexer: ESA, CST: Large memory footprint, fast search
  - Pure compressors: Strong compression, slow search
  - Variations we built: Compressors with additional search indexes
    - RLZ / Tong after modification: iRLZ, iTong

Evaluation: Indexing WikiPedia Revisions

Wikipedia Helsinki, ~3K versions
577 MB

Wikipedia GW Bush, ~45K versions
1400 MB

| HEL		Index size (MB)			
	|Strings|	40	160	640	2560
Compression only	RLZ.025	2.9	5.3	17.4	48.0
	RLZ.05	3.1	8.9	30.8	84.7
	RLZ.1	4.5	16.6	58.3	160.2
	Tong.025	7.3	2.5	6.3	18.0
	Tong.05	1.9	2.5	7.1	21.0
	Tong.1	1.4	2.9	9.4	26.1
Index-based	ConcCST	38.7	151.1	533.9	1,473.5
	ConcESA	443.2	1,722.6	6,077.8	16,642.7
	USConcCST	18.1	23.6	43.3	119.8
	USConcESA	169.1	221.5	406.4	1,121.5
	iRLZ.025	6.2	11.5	37.9	107.8
	iRLZ.05	6.3	18.6	63.9	180.3
	iRLZ.1	9.3	33.9	118.6	330.3
	iTong.025	14.6	6.4	16.2	51.4
	iTong.05	4.4	6.2	17.7	56.5
	iTong.1	3.4	6.8	21.8	65.5
	RCSI	2.7	5.3	21.7	115.8
	CPart	2.7	5.3	21.7	115.8
	CForest	2.7	4.4	11.3	44.1
	CDAG	2.6	4.1	9.5	31.7

GWB		Index size (MB)			
	|Strings|	80	640	5120	40960
Compression only	RLZ.025	8.9	40.1	246.2	957.9
	RLZ.05	10.1	67.7	444.0	1,702.4
	RLZ.1	18.0	127.6	837.2	3,215.0
	Tong.025	7.2	18.0	110.4	346.5
	Tong.05	4.7	22.6	127.0	NA
	Tong.1	5.2	29.9	152.4	491.2
Index-based	ConcCST	172.5	1,242.5	NA	NA
	ConcESA	1,921.1	13,891.8	NA	NA
	USConcCST	46.9	85.1	NA	NA
	USConcESA	436.4	796.5	NA	NA
	iRLZ.025	17.5	80.8	489.9	1,965.9
	iRLZ.05	21.6	137.7	877.5	3,460.2
	iRLZ.1	37.4	257.5	1,665.6	6,480.3
	iTong.025	14.2	39.2	208.5	768.7
	iTong.05	10.9	48.9	247.9	875.8
	iTong.1	12.2	60.1	302.3	1,034.0
	RCSI	10.5	46.1	530.0	4,421.5
	CPart	11.4	46.1	427.2	2,818.6
	CForest	10.3	22.5	122.7	778.8
	CDAG	10.2	19.9	80.6	390.2

- CDAG strongest of index-based, almost as small as best
- CDAG (or CPart) are fastest (2-4 times faster than iTong)

Evaluation: Searching (HEL)

5000 random queries, length 12-18 char
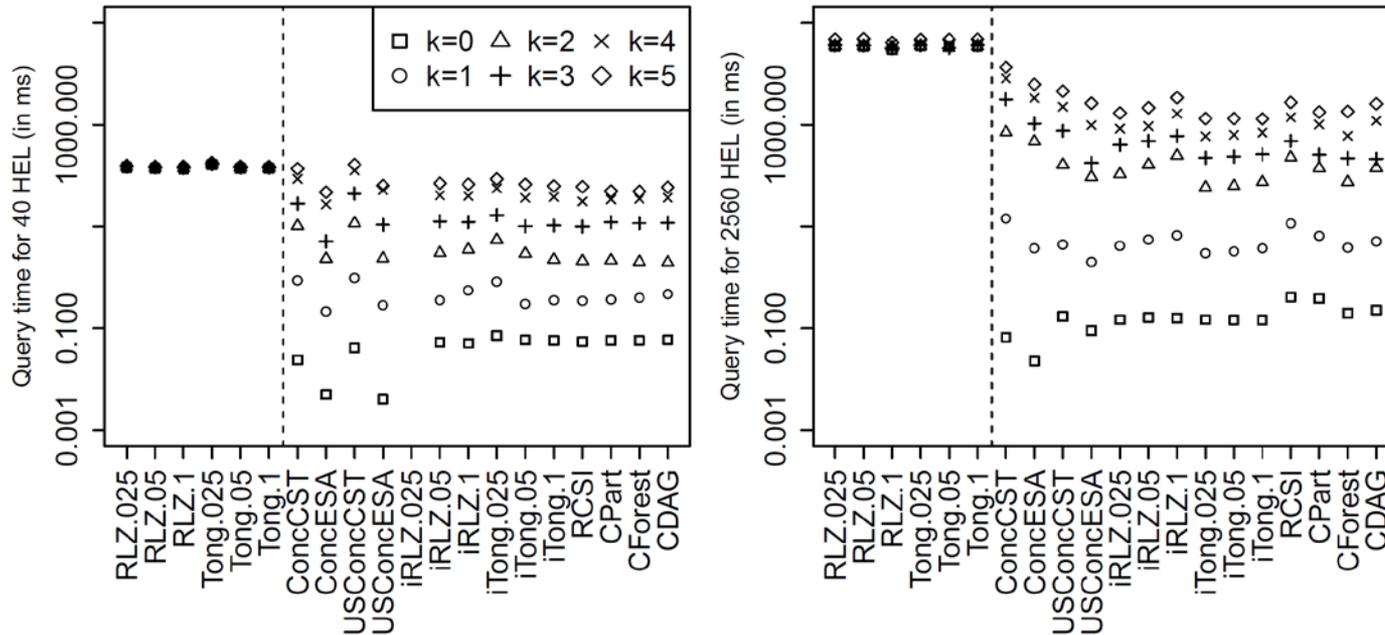
- ESA fastest in search
- All compressing method perform roughly the same

Evaluation: Large Datasets

Human chromosome 21, up to 640 versions, up to 51GB

	HG21	Index size (MB)				Indexing time (s)			
	|Strings|	10	40	160	640	10	40	160	640
Compression only	RLZ.025	175.6	380.9	561.1	**2,039.4**	**218.5**	**853.9**	5,125.8	**67,943.7**
	RLZ.05	**161.3**	332.2	956.9	NA	261.5	965.5	6,607.7	NA
	RLZ.1	178.9	460.8	1,799.8	NA	379.7	1,496.5	10,311.5	NA
	Tong.025	185.5	738.0	225.3	NA	469.1	1,857.9	6,808.7	NA
	Tong.05	183.5	204.4	**223.8**	NA	655.8	2,136.4	10,246.1	NA
	Tong.1	204.0	**131.7**	294.8	NA	1,307.5	4,336.4	22,574.2	NA
Index-based	ConcCST	1,139.6	NA	NA	NA	1,378.7	NA	NA	NA
	ConcESA	12,028.2	NA	NA	NA	1,126.7	NA	NA	NA
	USConcCST	11,729.0	NA	NA	NA	18,555.9	NA	NA	NA
	USConcESA	NA	NA	NA	NA	NA	NA	NA	NA
	iRLZ.025	1,616.4	1,287.9	1,154.5	4,101.7	1,828.9	1,901.6	5,678.5	68,463.0
	iRLZ.05	1,233.4	911.3	1,965.0	NA	1,391.0	1,332.3	6,869.0	NA
	iRLZ.1	958.7	1,008.9	3,691.0	NA	1,080.3	1,630.9	10,594.1	NA
	iTong.025	2,130.8	2,718.8	512.9	NA	2,544.7	4,511.2	7,309.1	NA
	iTong.05	2,038.7	698.8	516.0	NA	2,745.3	2,792.4	10,747.0	NA
	iTong.1	1,906.1	362.5	664.5	NA	3,218.1	4,526.1	23,058.4	NA
	RCSI	277.5	314.7	380.6	687.0	432.3	**499.4**	693.5	1,562.4
	CPart	277.5	314.7	380.6	687.0	**416.8**	507.8	806.1	1,671.8
	CForest	276.4	309.4	357.0	581.9	435.4	502.8	764.1	1,894.8
	CDAG	**275.9**	**305.6**	**341.5**	**512.9**	433.6	509.0	745.0	1,745.0

Conclusions

- Referential compression beats standard compression tools by orders of magnitude for highly-similar sequences (w.r.t. storage and speed)

- Inherent trade-off between compression ratio and de-/compression speed

- Given a referential index, some (many?) string matching problems can be solved much more efficiently – ample room for further research
  - "Compressive genomics"

References

- Deorowicz, Sebastian, Agnieszka Danek, and Marcin Niemiec. "GDC 2: Compression of large collections of genomes." arXiv preprint arXiv:1503.01624 (2015).

- Wandelt, S. and U. Leser (2014). "MRCSI: Compressing and Searching String Collections with Multiple References". PVLDB. Kona, Hawaii.

- Wandelt, S. and U. Leser (2013). "FRESCO: Referential Compression of Highly-Similar sequences." Transactions on Computational Biology and Bioinformatics 10(5): 1275-1288.

- Wandelt, S. and U. Leser (2012). "Adaptive efficient compression of genomes." Algorithms for Molecular Biology 7(30).

- Wandelt, S., J. Starlinger, M. Bux and U. Leser (2013). "RCSI: Scalable similarity search in thousand(s) of genome"s. PVLDB, Hangzhou, China.

- Wandelt, S., Rheinländer, A., Bux, M., Thalheim, L., Haldemann, B. and Leser, U. (2012). "Data Management Challenges in Next Generation Sequencing." Datenbank Spektrum