UNIVERSITÄT
BIELEFELD

Searching for Genomic Variants in Multiple Genomes at Once

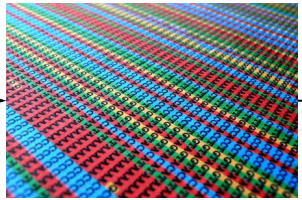
Jens Stoye

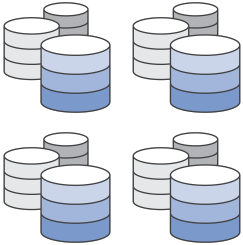
Faculty of Technology, Center for Biotechnology (CeBiTec), Bielefeld Institute for Bioinformatics Infrastructure (BIBI)
Bielefeld University, Germany

(joint work with Tizian Schulz, Roland Wittler, Sven Rahmann, Faraz Hach)

Motivation

Current status in DNA sequence analysis:





- Sequencing technologies are cheap and fast
- Numerous individual genomes per species are sequenced

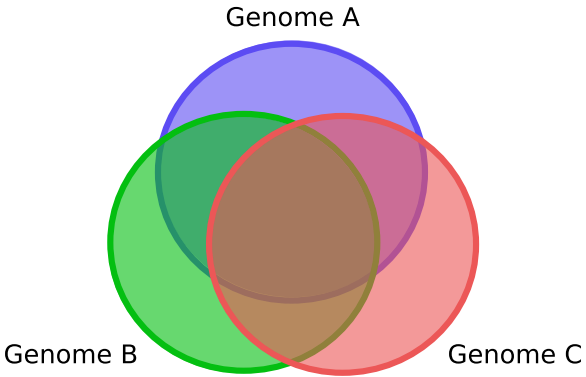
→ Huge amounts of partially redundant sequences

Challenges:

- Efficient storage
- Efficient analysis
- Effective analysis

The sequence-based pangenome

How to efficiently store huge amounts of partially redundant sequences?



Colored de Bruijn graphs

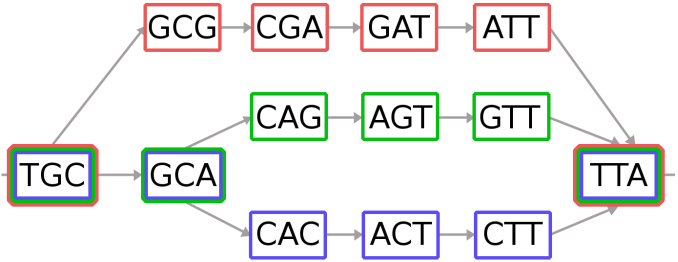
Colored de Bruijn graph (C-DBG)

$k = 3$

S_1 : TGCGATTA

S_2 : TGCAGTTA

S_3 : TGCACTTA



- Vertices represent substrings of length k (k-mers)
- Associated color represents sequence of origin
- Edges between vertices that share a $k - 1$ overlap

Colored de Bruijn graphs

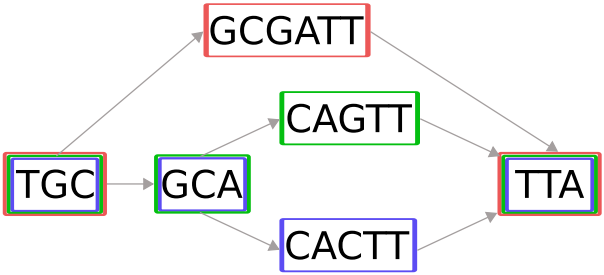
Compacted colored de Bruijn graph

$k = 3$

S_1 : TGCGATTA

S_2 : TGCAGTTA

S_3 : TGCACTTA

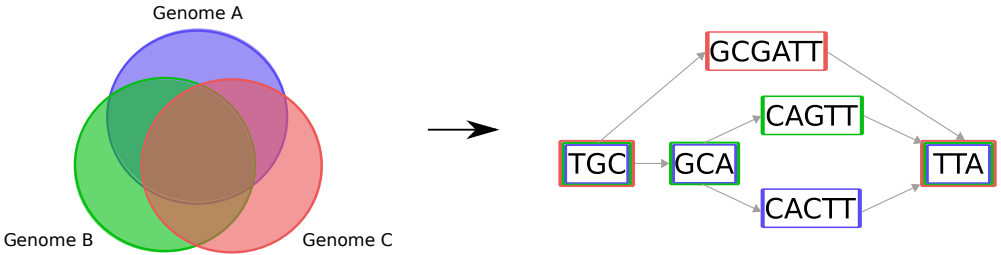


- Vertices of unique paths are replaced by single vertex (unitig)
- Label of unitig is sequence spelled by the path
- Allows even more efficiency

Colored de Bruijn graphs (C-DBGs)

Why to store pan-genomes as compacted colored de Bruijn graphs?

- Reduced memory requirements
- No data preprocessing necessary (assembly)
- Data is stored based on similarity
- Allows analysis of multiple sequences in parallel



Querying a sequence DB - BLAST

BLAST – Basic Local Alignment Search Tool

Developed by Stephen Altschul *et al.* in 1990 (second version in 1997)
Is used to query databases of DNA and protein sequences

Different flavors are available:

- DNA→DNA (blastn)
- protein→protein (blastp)
- translated DNA→protein (blastx)
- protein→translated DNA (tblastn)
- ...

Algorithmic steps:

1. Finding BLAST hits
2. Hit extension *X-drop algorithm*
3. Gapped alignment calculation

Querying a sequence DB - BLAST

BLAST – Basic Local Alignment Search Tool

Developed by Stephen Altschul *et al.* in 1990 (second version in 1997)
Is used to query databases of DNA and protein sequences

Problem statement:

Given a query sequence $x \in \Sigma^*$ and a sequence database Y, find all highest scoring local alignments between x and $y \in Y$ above a certain significance value and return them along with their score.

Querying a colored de Bruijn graph - PLAST

How to query a C-DBG to analyze the pangenome it represents?

Idea: Develop a method that queries a C-DBG in a BLAST-like manner to find local alignments between a query sequence and the sequences stored in the C-DBG.

$\rightarrow$ *Pangenome* Local Alignment Search Tool – PLAST

Problem statement:
Given a query sequence $x \in \Sigma^*$ and a set of sequences Y represented as a C-DBG, find all highest scoring local alignments between x and $y \in Y$ above a certain significance value and return them along with their score.

Querying a colored de Bruijn graph - PLAST

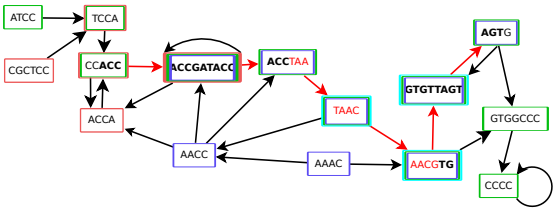
How to query a C-DBG to analyze the pangenome it represents?

Idea: Develop a method that queries a C-DBG in a BLAST-like manner to find local alignments between a query sequence and the sequences stored in the C-DBG.

→ *Pangenome* Local Alignment Search Tool – PLAST

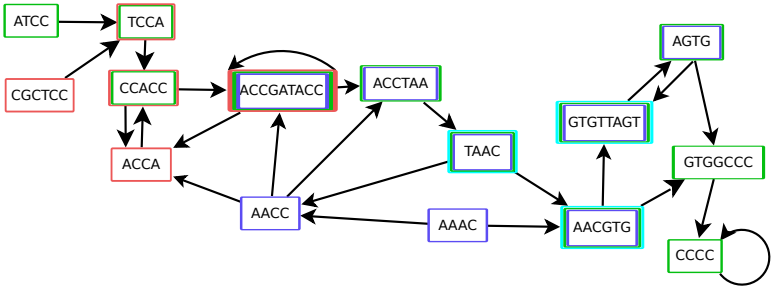
Procedure of PLAST:

1. Seed detection
2. Seed extension
3. Gapped alignment calculation

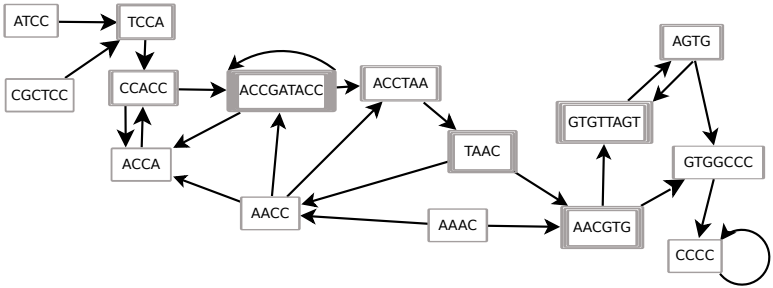


x: A G G **A C C G** - **T A** - - T A A C G **G G G A A G T** A C T
y: **A C C G A T A C C** T A A C G **T G T T A G T**

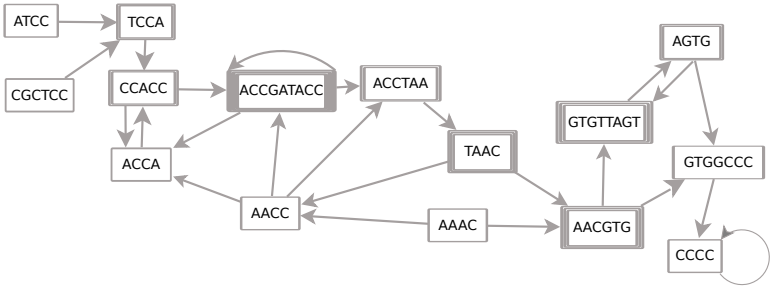
Querying a colored de Bruijn graph - PLAST



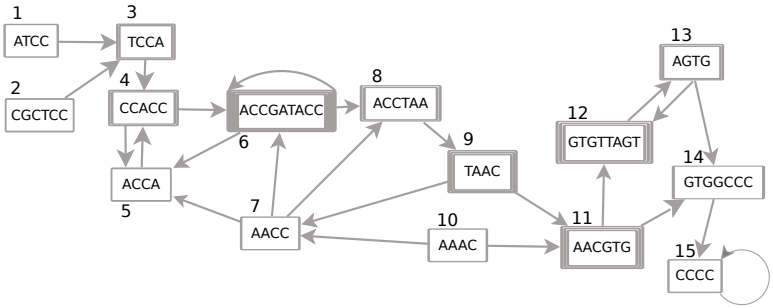
Querying a colored de Bruijn graph - PLAST



Querying a colored de Bruijn graph - PLAST



Querying a colored de Bruijn graph - PLAST

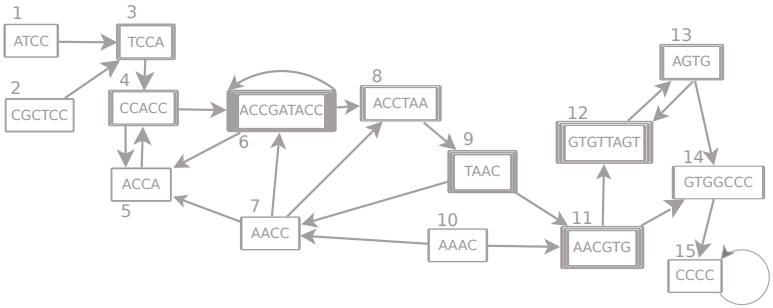


(minimal exact match length $s = 3$)

		(vertex_id, offset)
AAA	1	(10,0)
AAC	3	(7,0)
AAG	3	(11,0)
AAT	3	(5,0)
ACA	3	(6,0)
ACC	6	(8,0)
ACG	7	(11,1)
ACT	7	...
	...	...
CCC	13	(15,0)
CCG	14	(6,1)
CCT	15	(8,1)
	...	...
GTT	27	(12,2)
TAA	28	(9,0)
TAC	29	(6,5)
	...	...

Querying a colored de Bruijn graph - PLAST

1. Seed detection:



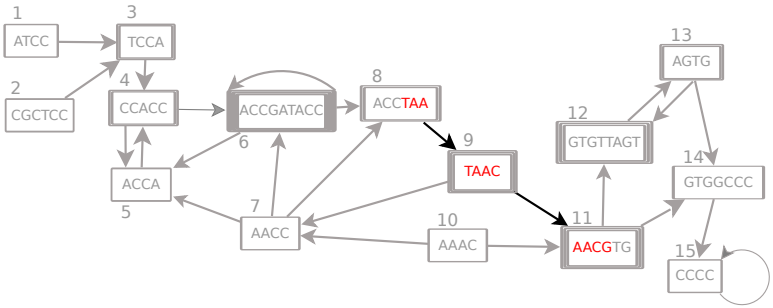
		(vertex_id, offset)
AAA	1	
AAC	3	(10,0)
AAG	3	(7,0)
AAT	3	(11,0)
ACA	3	(5,0)
ACC	6	(6,0)
ACG	7	(8,0)
ACT	7	(11,1)
	...	...
CCC	13	(15,0)
CCG	14	(6,1)
CCT	15	(8,1)
	...	...
GTT	27	(12,2)
TAA	28	(9,0)
TAC	29	(6,5)
	...	...

x: A G G A C C G T A T A A C G G G G A A G T A C T

(minimal exact match length $s = 3$)

Querying a colored de Bruijn graph - PLAST

1. Seed detection:



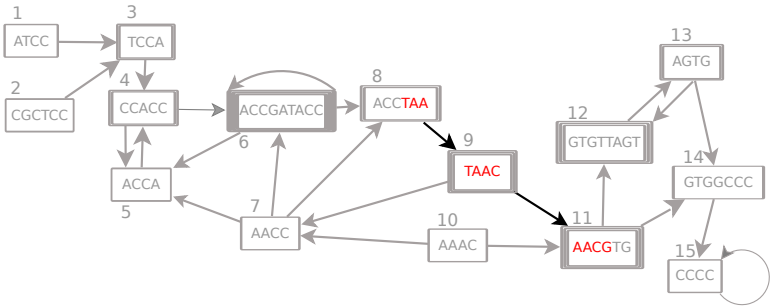
x: A G G A C C G T A **T A A C G** G G G A A G T A C T

(minimal exact match length $s = 3$)

		(vertex_id, offset)
AAA	1	(10,0)
AAC	3	(7,0)
AAG	3	(11,0)
AAT	3	(5,0)
ACA	3	(6,0)
ACC	6	(8,0)
ACG	7	(11,1)
ACT	7	...
	...	(15,0)
CCC	13	(6,1)
CCG	14	(8,1)
CCT	15	...
	...	(12,2)
GTT	27	(9,0)
TAA	28	(6,5)
TAC	29	...
	...	

Querying a colored de Bruijn graph - PLAST

1. Seed detection:



x: A G G A C C G T A **T A A C G** G G G A A G T A C T

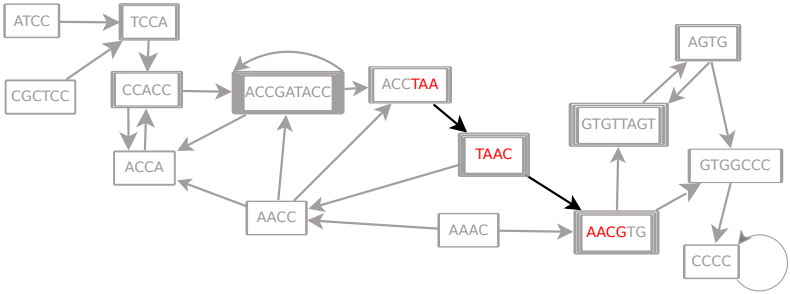
(minimal exact match length $s = 3$)

		(vertex_id, offset)
AAA	1	(10,0)
AAC	3	(7,0)
AAG	3	(11,0)
AAT	3	(5,0)
ACA	3	(6,0)
ACC	6	(8,0)
ACG	7	(11,1)
ACT	7	...
	...	(15,0)
CCC	13	(6,1)
CCG	14	(8,1)
CCT	15	...
	...	(12,2)
GTT	27	(9,0)
TAA	28	(6,5)
TAC	29	...
	...	

Possible in the same way BLAST does, using a precalculated index of the graph

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



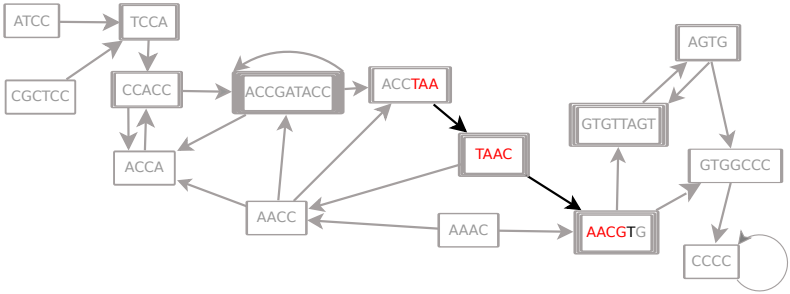
x: A G G A C C G T A T A A C G G G G A A G T A C T

T A A C G

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



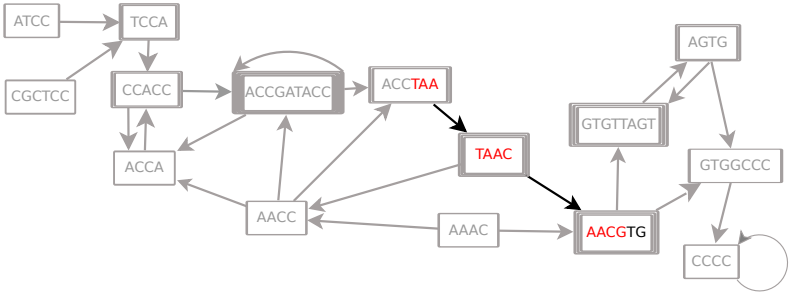
x: A G G A C C G T A T A A C G G G G A A G T A C T

T A A C G T

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



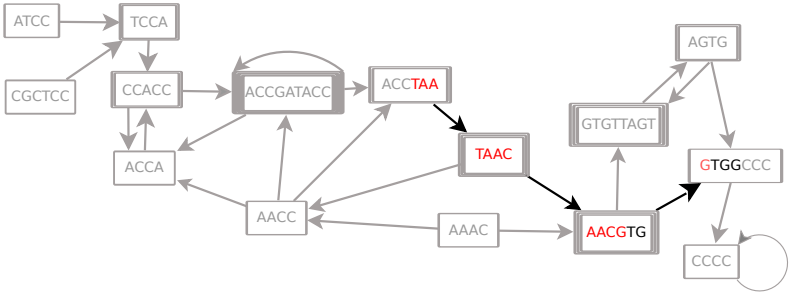
x: A G G A C C G T A T A A C G G G A A G T A C T

T A A C G T G

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



x: A G G A C C G T A T A A C G G G G A A G T A C T

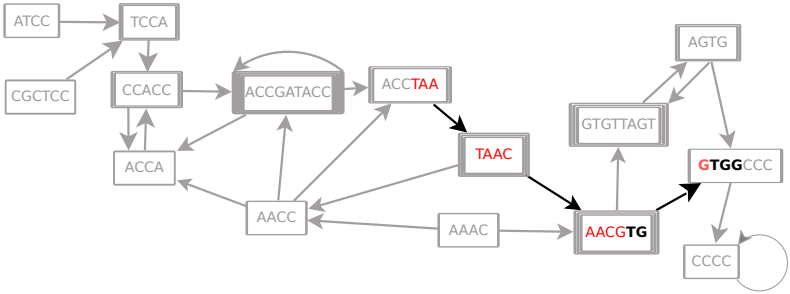
T A A C G T G G

\- + +

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



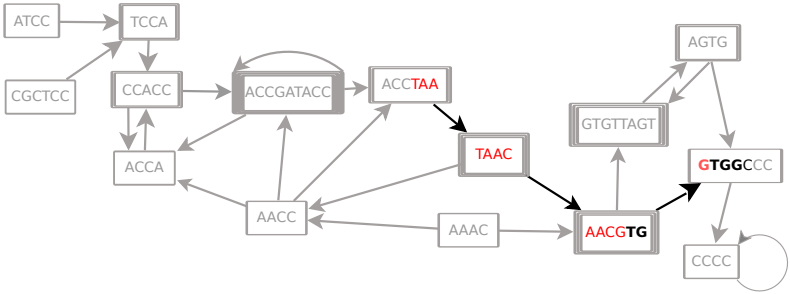
x: A G G A C C G T A T A A C G **G G G** A A G T A C T

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

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



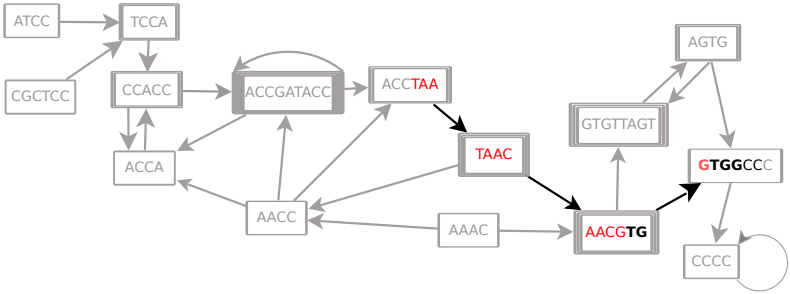
x: A G G A C C G T A T A A C G G G G A A G T A C T

T A A C G T G G C

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



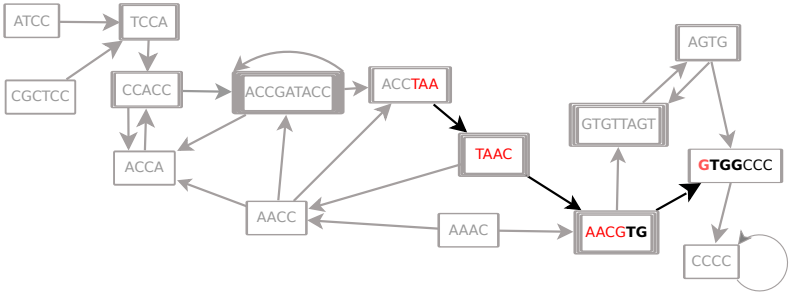
x: A G G A C C G T A T A A C G G G G A A G T A C T

T A A C G T G G C C

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



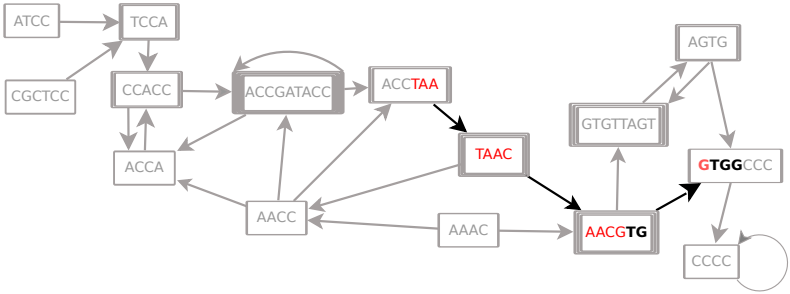
x: A G G A C C G T A T A A C G G G G A A G T A C T

T A A C G T G G C C C

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



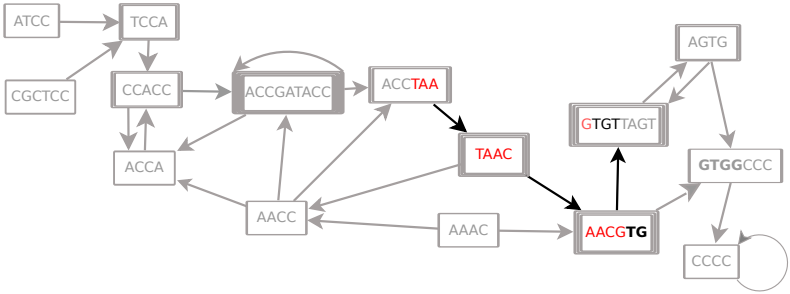
x: A G G A C C G T A T A A C G **G G G** A A G T A C T

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

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



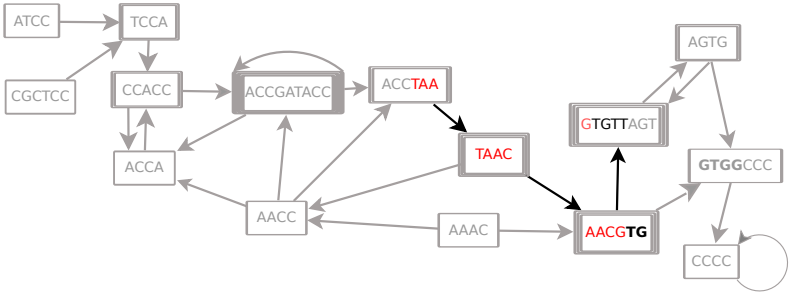
x: A G G A C C G T A T A A C G G G G A A G T A C T

T A A C G T G T

$X = 3$ T G G

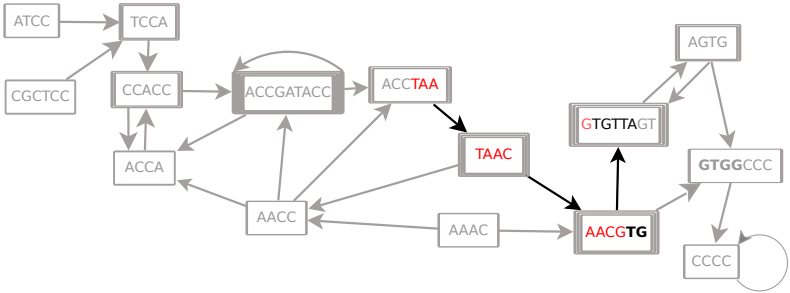
Querying a colored de Bruijn graph - PLAST

2. Seed extension:



Querying a colored de Bruijn graph - PLAST

2. Seed extension:



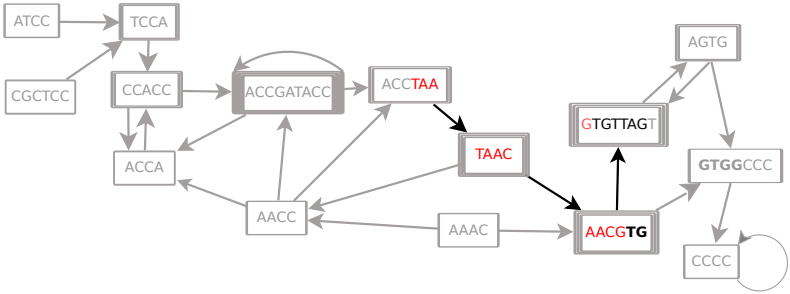
x: A G G A C C G T A T A A C G G G G A A G T A C T

T A A C G T G T T A

$X = 3$ T G G

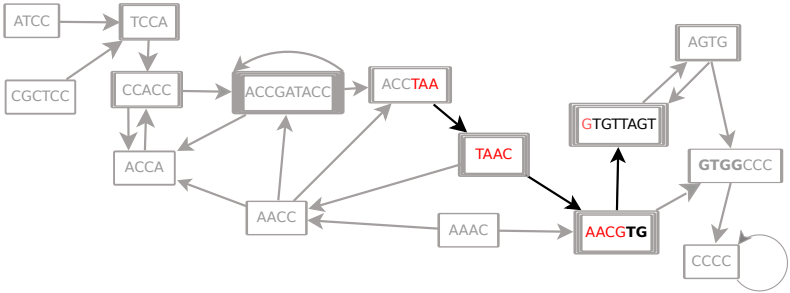
Querying a colored de Bruijn graph - PLAST

2. Seed extension:



Querying a colored de Bruijn graph - PLAST

2. Seed extension:



x: A G G A C C G T A T A A C G G G G A A G T A C T

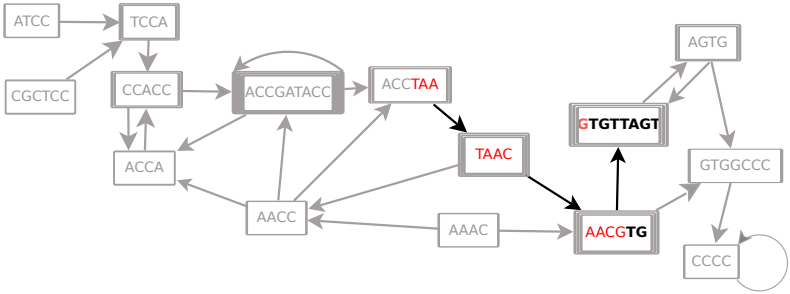
T A A C G T G T T A G T

$X = 3$ T G G

- + - - + + +

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



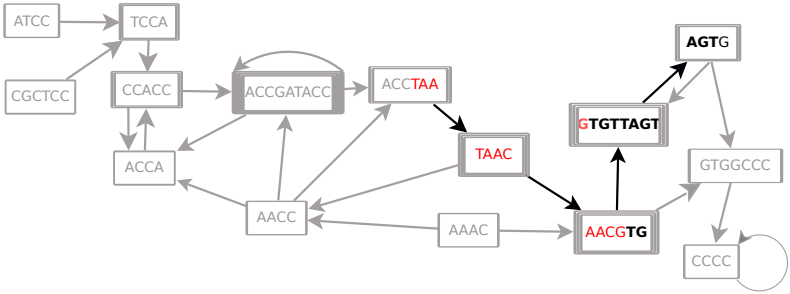
x: A G G A C C G T A **T A A C G G G A A G T** A C T

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

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



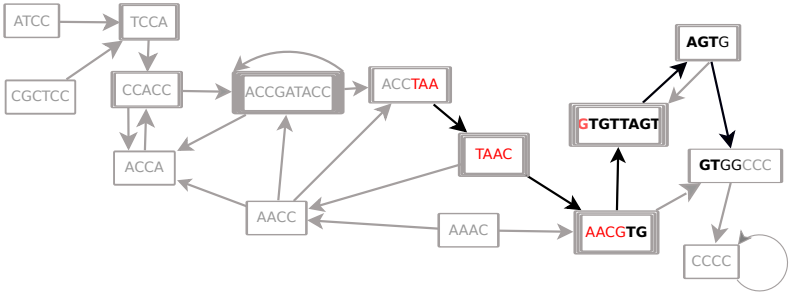
x: A G G A C C G T A T A A C G G G G A A G T A C T

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

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



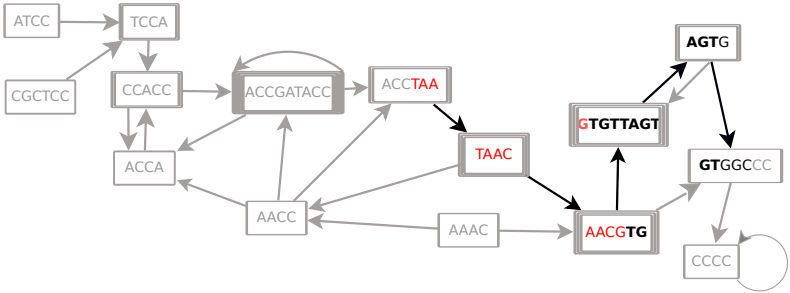
x: A G G A C C G T A T A A C G G G G A A G T A C T

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

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



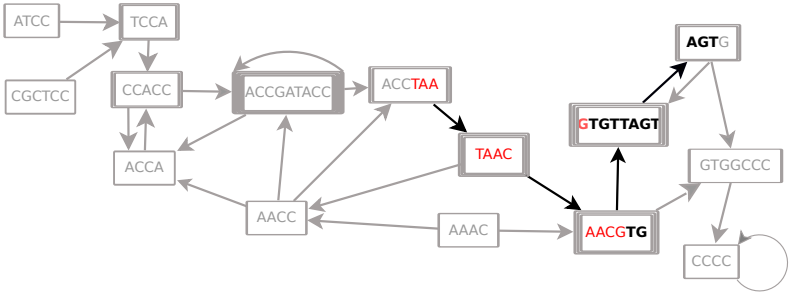
x: A G G A C C G T A T A A C G G G G A A G T A C T

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

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



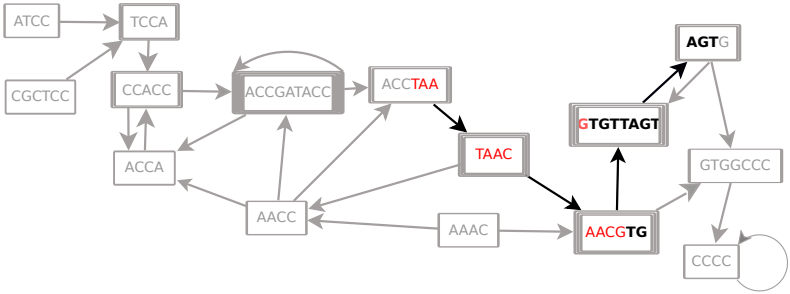
x: A G G A C C G T A T A A C G G G G A A G T A C T

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

$X = 3$

Querying a colored de Bruijn graph - PLAST

2. Seed extension:



x: A G G A C C G T A T A A C G G G G A A G T A C T

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

$X = 3$

- Expensive since graph sequence is ambiguous
- Brute-force traversal (DFS) mostly feasible

Querying a colored de Bruijn graph - PLAST

The endless extension problem:

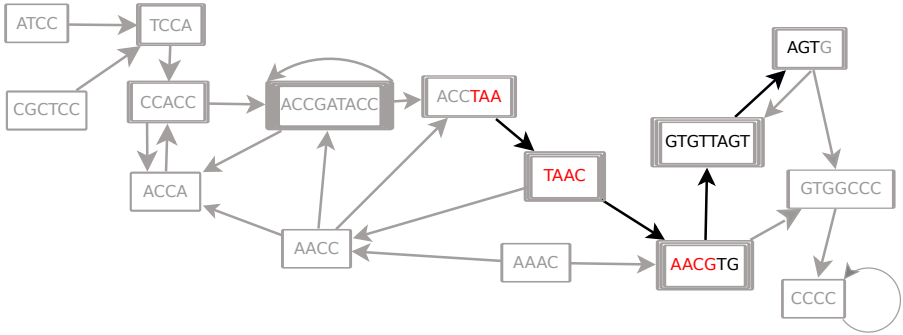
- Graphs may have some collapsed regions
- Regions have
  - many short loops
  - vertices with short sequences and high branching factor

$\rightarrow$ nearly every possible sequence may be generated here

$\rightarrow$ X-drop algorithm is unable to end extensions $\Rightarrow$ hard break

Querying a colored de Bruijn graph - PLAST

3. Gapped alignment of best extended seeds:

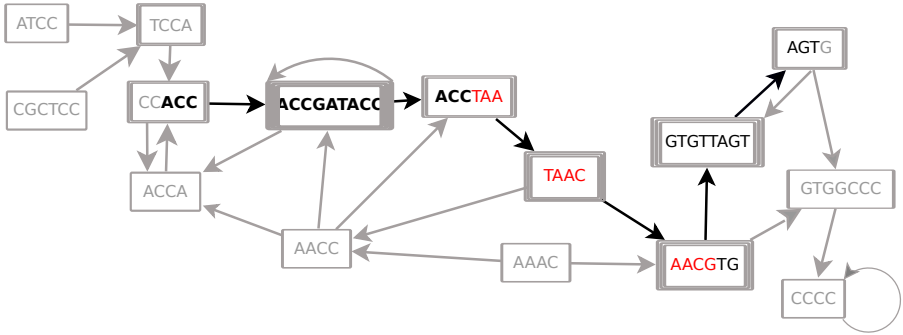


x: A G G A C C G T A T A A C G G G G A A G T A C T

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

Querying a colored de Bruijn graph - PLAST

3. Gapped alignment of best extended seeds:

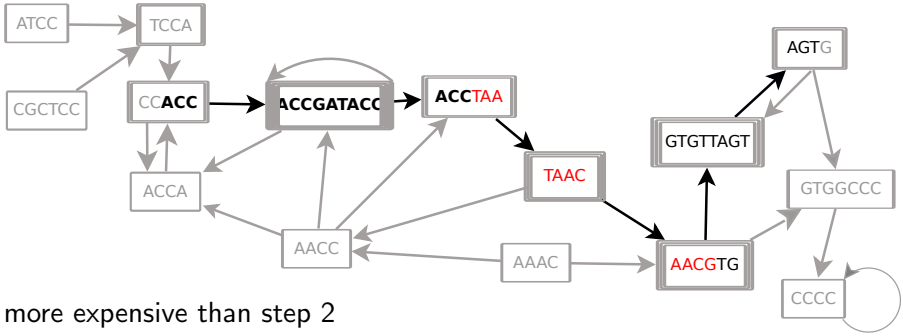


- Banded, gapped alignment calculation following the extension path

x: A G G **A C C G - T A - -** T A A C G G G G A A G T A C T
A C C G A T A C C T A A C G T G T T A G T

Querying a colored de Bruijn graph - PLAST

3. Gapped alignment of best extended seeds:



- Even more expensive than step 2
- Only performed for a small subset of all findings, thus feasible

x: A G G **A C C G - T A - -** T A A C G G G G A A G T A C T
A C C G A T A C C T A A C G T G T T A G T

Usage of quorum and search color set

PLAST allows to find

- highest scoring (consensus) alignments (by default)

Usage of quorum and search color set

PLAST allows to find

- highest scoring (consensus) alignments (by default)
- highest scoring alignments shared by a certain number of genomes (using a **quorum**)

Usage of quorum and search color set

PLAST allows to find

- highest scoring (consensus) alignments (by default)
- highest scoring alignments shared by a certain number of genomes (using a **quorum**)
- highest scoring alignments supported by a subset of genomes (using a **search color set**)

Alignment Statistics

Alignment statistic for BLAST:

Alignment Statistics

Alignment statistic for BLAST:

BLAST ranks findings according to significance values

E-value:

$$E = \underbrace{Kmn}_{C} e^{-\lambda S}$$

P-value:

$$P = 1 - e^{-E}$$

where n = query length, m = database size

Alignment Statistics

Alignment statistic for BLAST:

BLAST ranks findings according to significance values

E-value:

$$E = \underbrace{Kmn}_{C} e^{-\lambda S}$$

P-value:

$$P = 1 - e^{-E}$$

where n = query length, m = database size

How does a sequence to graph alignment statistic work?

Alignment Statistics on a Pangenome

Assumptions:

Alignment Statistics on a Pangenome

Assumptions:

- Graph sequences are highly similar (random hits –)

Alignment Statistics on a Pangenome

Assumptions:

- Graph sequences are highly similar (random hits −)
- High branching factors increase number of possible sequences (random hits +)

Alignment Statistics on a Pangenome

Assumptions:

- Graph sequences are highly similar (random hits $-$)
- High branching factors increase number of possible sequences (random hits $+$)
- Maximum scores follows an exponential distribution

Alignment Statistics on a Pangenome

Assumptions:

- Graph sequences are highly similar (random hits $-$)
- High branching factors increase number of possible sequences (random hits $+$)
- Maximum scores follows an exponential distribution
- λ and C depend on sequence relatedness and diversity within pangenome

But: Details unknown!

Parameter estimations

How to get good estimates for λ and C?

- Our approach: Obtain estimates by random sampling
- Problem:
  - Frequency of high scoring alignments is very low ($< 10^{-6}$)
  - A lot of sampling required
- But: Parameters need to be reliable esp. for high scores

$\rightarrow$ Importance sampling based on *Metropolis-Hastings Markov Chain Monte Carlo* (MCMC) strategy

Importance sampling via MCMC

Idea: Construct Markov chain such that the probability to sample a random sequence with score s is exponentially biased towards higher scores

Procedure:

1. From sequence x propose a new sequence y by substituting/ deleting*/ inserting* a single base in x
2. *Accept* proposal with probability $\min\{1, \exp(\lambda_0 \cdot (s_y - s_x))\}$
3. If y is accepted make a new proposal for y (for x otherwise)

$\rightarrow$ after $2n/3$ accepts, procedure yields new sample with uncorrelated score

$\rightarrow$ Calculate sample's score

* with shifting in/out a base at the sequence borders

Estimation of λ and C

Follow "naive" random sampling strategy to obtain initial estimates for λ and C

Generate many sample sequences using MCMC and score them

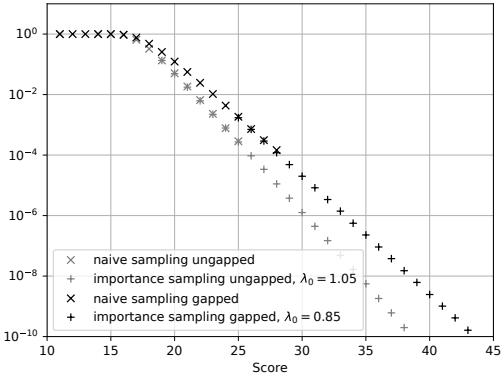
Let R_s be the absolute number of times score s was observed among all samples, then

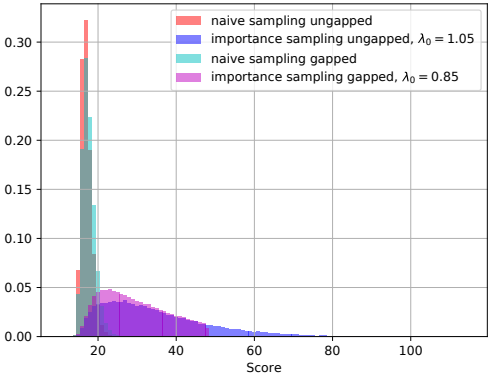
$$T_s := \Sigma_{s' \geq s} R_{s'} \cdot \exp(-\lambda_0 \cdot s')$$

λ can be estimated by fitting line to points $(s, \log T_s)$

C is estimated from 10% of highest scores of naive sampling

Estimation of λ and C for 220 *Salmonella enterica* genomes





Conceptual comparison: BLAST vs. PLAST

BLAST

- Allows comparison of
  - query and each DB sequence
- Processes every DB sequence independent from each other
- Increased runtime if sequence similarity increases

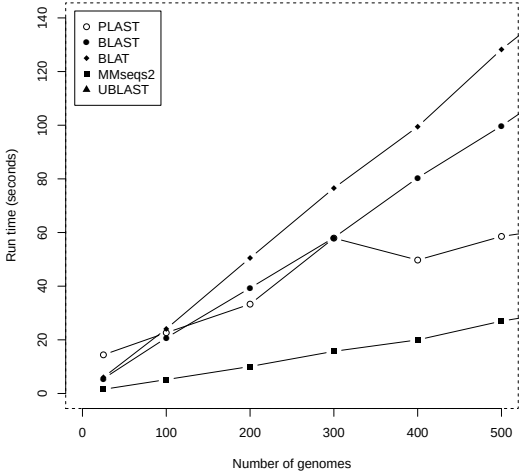
PLAST

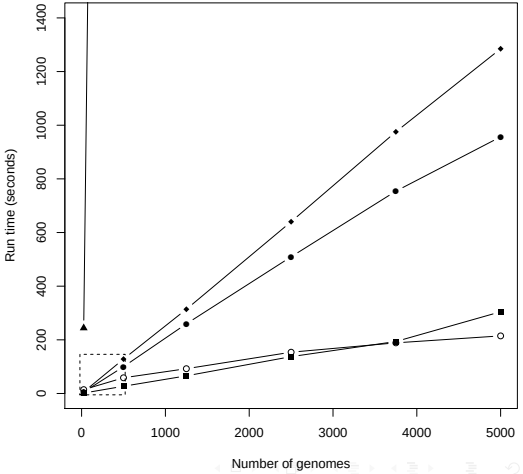
- Allows comparison of
  - query and each graph sequence
  - graph sequences among each other (with respect to query)
- Processes many sequences in parallel most of the time
- Saves run time with increasing similarity of graph sequences

Runtime and memory comparison

Database: 5,000 *Salmonella typhimurium* assemblies (total: 24 GB)

Queries: 100 random substrings of length 1,000 from the *Salmonella* reference genome

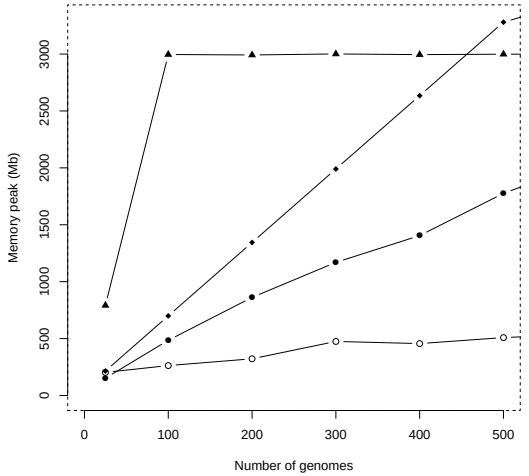


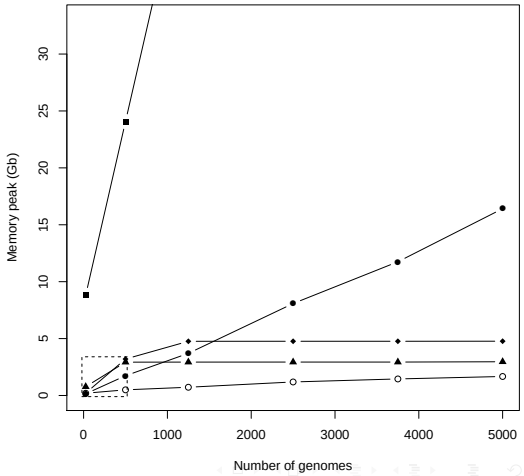


Runtime and memory comparison

Database: 5,000 *Salmonella typhimurium* assemblies (total: 24 GB)

Queries: 100 random substrings of length 1,000 from the *Salmonella* reference genome





Experimental comparison to other tools

Tool	Results	Tool\PLAST	PLAST\Tool	Tool\BLAST	BLAST\Tool
PLAST	7,565	–	–	357 4.72 %	290 0.02 %
BLAST	1,246,221	290 0.02 %	357 4.72 %	–	–
BLAT	457,089	1 0.00 %	456 6.03 %	508 0.11 %	49,798 4.00 %
MMseqs2	695,792	6 0.00 %	322 4.26 %	800 0.12 %	21,022 1.69 %
UBLAST	4,881,509	111,386 2.28 %	272 3.59 %	220,577 4.52 %	5,459 0.44 %

Two results *match* if they overlap by at least 90 % of the shorter alignment.

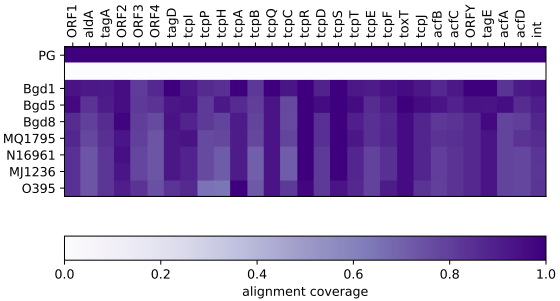
Not included:

- DIAMOND (protein databases only)
- SWORD (proteins only)
- GHOSTZ (index building took too long)
- LAST (index building took too long)
- RAPSearch2 (reported issue)
- LASTZ, YASS, BLASTZ (performed worse in earlier studies)

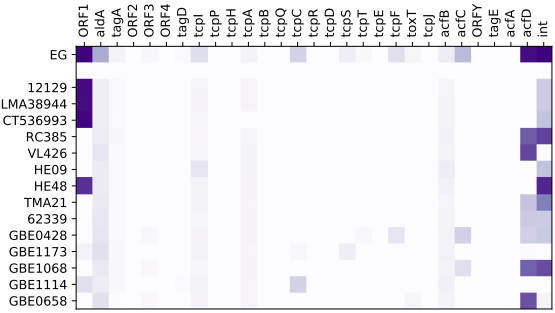
Use case: Pathogenicity islands in *Vibrio cholerae*

Database: *Vibrio cholerae* pangenome consisting of 21 genomes; some assembled, some not

Query: VPI-1 region: 19 genes associated with pathogenicity



results for "pandemic genones" subset



results for "environmental genones" subset

Construction of a human pangenome

We used

- variant information from the 1000 Genomes Project phase 3 and
- human reference genome GRCh37

to construct a pangenome of all 2,504 human individuals for

- chromosome 2 and
- chromosome 15

Beyond microbial pangenomes

Exemplary use case: Investigation of polymorphism within human

- SNP *rs1426654* influences skin pigmentation
  - Reference allele: light skin color (West Eurasian ancestry)
  - Located on exon 3 of gene *SLC24A5* (chr. 15)
- PLAST input:
  - Pangenome of human chromosome 15
  - Reference sequence of exon 3 as query (GRCh37)
- Result:
  - No quorum: perfect match (reference), single mismatch (variant)
  - 99% quorum of all European samples: reference allele exclusively

Beyond microbial pangenomes

Performance on human pangenome: PLAST vs. MMseqs2

- Setup:
  - Data set: Pangenome of chr. 2 ($\approx$ 8% of comp. human genome)
  - Search of 100 1kb queries (randomly drawn from reference)
- PLAST took
  - 317s per query on average
  - and 24GB of memory
  - on a single core
- MMseqs2 (on subset of only 1,000 chromosomes) took
  - 339s per query on average
  - and 259 GB of memory
  - on all available 28 cores
- Sizes of input files on disk:
  - 9.2 GB (PLAST)
  - 2.2 TB (MMseqs2)

Conclusion and Outlook

Conclusions:

- Increasing amounts of available DNA sequences motivate the usage of pangenomic approaches

Conclusion and Outlook

Conclusions:

- Increasing amounts of available DNA sequences motivate the usage of pangenomic approaches
- Querying of C-DBGs in a BLAST-like manner seems to be computationally feasible

Conclusion and Outlook

Conclusions:

- Increasing amounts of available DNA sequences motivate the usage of pangenomic approaches
- Querying of C-DBGs in a BLAST-like manner seems to be computationally feasible
- Searching withing C-DBGs allows comparison of graph sequences in parallel

End

Thank you!

Manuscript on bioRxiv:
https://doi.org/10.1101/2020.09.03.280958