Assembly-free detection of CRISPR-Cas systems in metagenomes through HMM-guided search in de Bruijn graphs

Progress Report

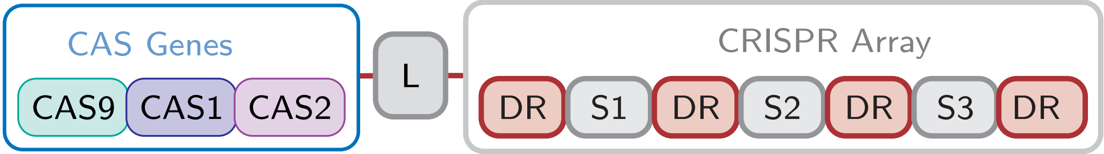
Fikrat Talibli

February 2026

Outline

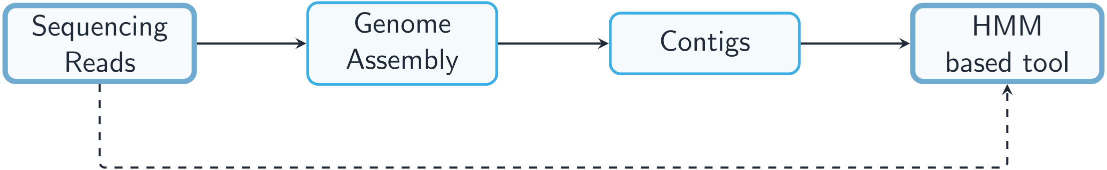
1. Background
2. Cas Detection Workflow
3. Summary

What is CRISPR-Cas?

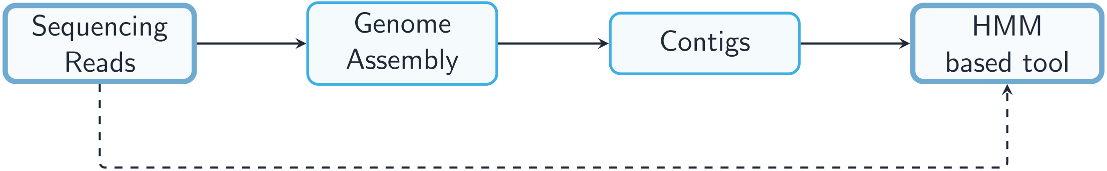


- SubTypes: I-A, I-B, I-C,...,II-A... - 50 subtypes from crisprcasdb
- Typing/subtyping is governed by rules
- Are at the distance $|L|$ either *UPSTREAM* or *DOWNSTREAM*

Metagenomic Datasets

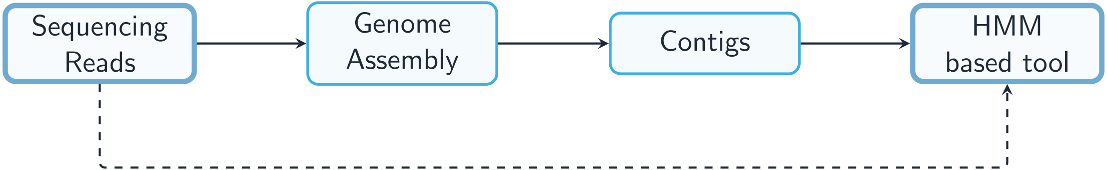


Metagenomic Datasets



Goal: Retrieve CRISPR Systems from metagenomic datasets

Metagenomic Datasets

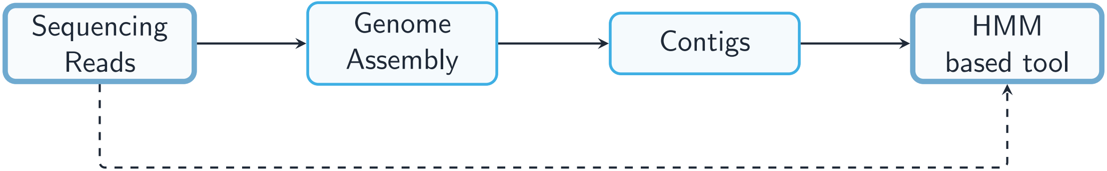


Goal: Retrieve CRISPR Systems from metagenomic datasets

Challenges:

- Short sequencing reads - 125 bps, genes - from 150bps ⇒ not feasible
- Contigs - short, upto 1000s bps ⇒ not enough for casettes

Metagenomic Datasets



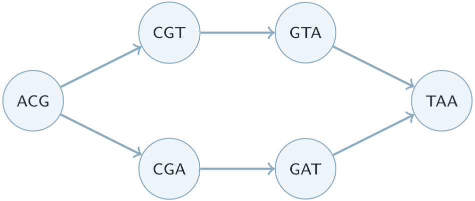
Goal: Retrieve CRISPR Systems from metagenomic datasets

Challenges:

- Short sequencing reads - 125 bps, genes - from 150bps ⇒ not feasible
- Contigs - short, upto 1000s bps ⇒ not enough for casettes

Our approach: Skip assembly, work directly on the **de Bruijn graph**

What is a de Bruijn Graph?



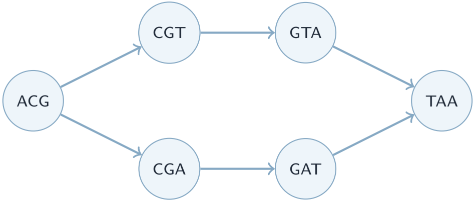
Nodes: `N k-mers (DNA words)`

Edges: `E - overlaps of` $k-1$

Paths: p_i `- possible sequences`

Our cases: `k=23, |E|=[1-18] b., ±10`^{12}

What is a de Bruijn Graph?



Nodes: `N k-mers (DNA words)`

Edges: `E - overlaps of` $k-1$

Paths: p_i `- possible sequences`

Our cases: `k=23, |E|=[1-18] b., ±10`^{12}

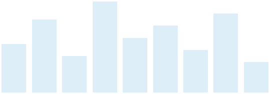
Challenge: Branchings and scale create multiple noisy paths

- Heuristics that do not compromise the accuracy
- Algorithm that corresponds to modern standards
- Elegant to define and fun to develop

How do we recognize a CAS gene in regular contigs

Hidden Markov Models (HMMs) — statistical profiles of protein families

HMM Profile



Position-specific scores

Query Sequence

$\xrightarrow{\text{align}}$ `MKTLLVGNTGSGKS...`

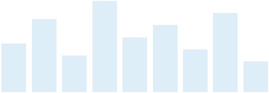
Score: 450 bits, 0.9 bits/AA

⇒ **Cas1 gene**

How do we recognize a CAS gene in regular contigs

Hidden Markov Models (HMMs) — statistical profiles of protein families

HMM Profile



Position-specific scores

Query Sequence

$\xrightarrow{\text{align}}$ `MKTLLVGNTGSGKS...`

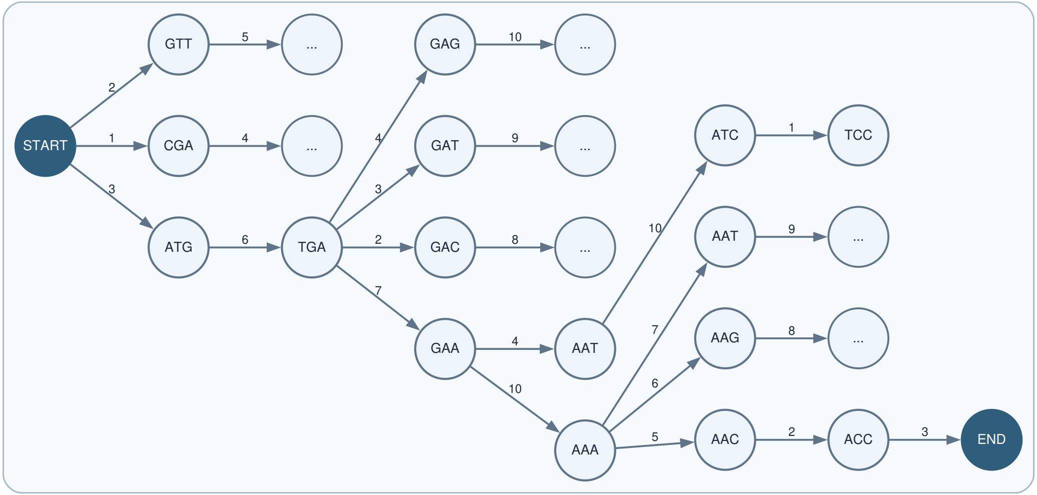
Score: 450 bits, 0.9 bits/AA

⇒ **Cas1 gene**

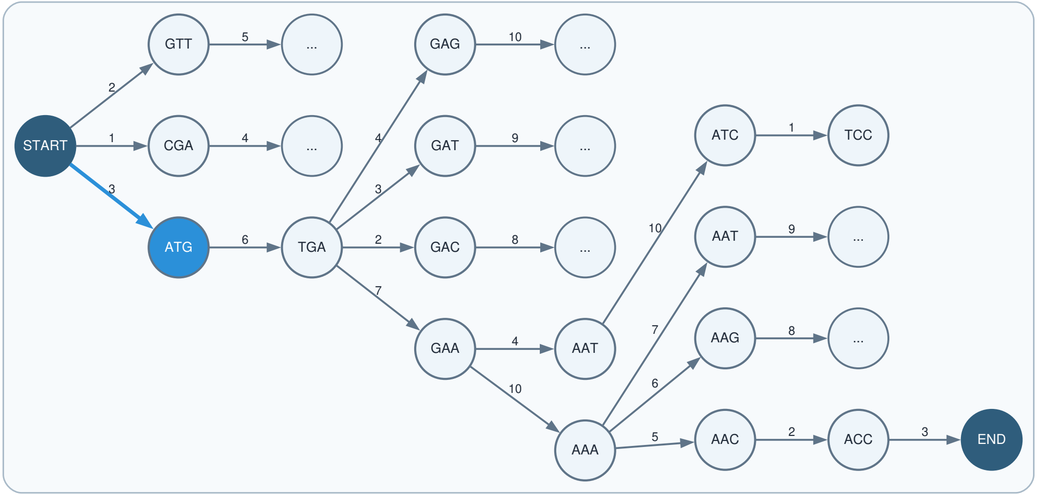
The process:

1. Find a potential START codon (ATG, GTG, TTG)
2. Translate DNA → protein
3. Score against HMM profile
4. High score = CAS gene match

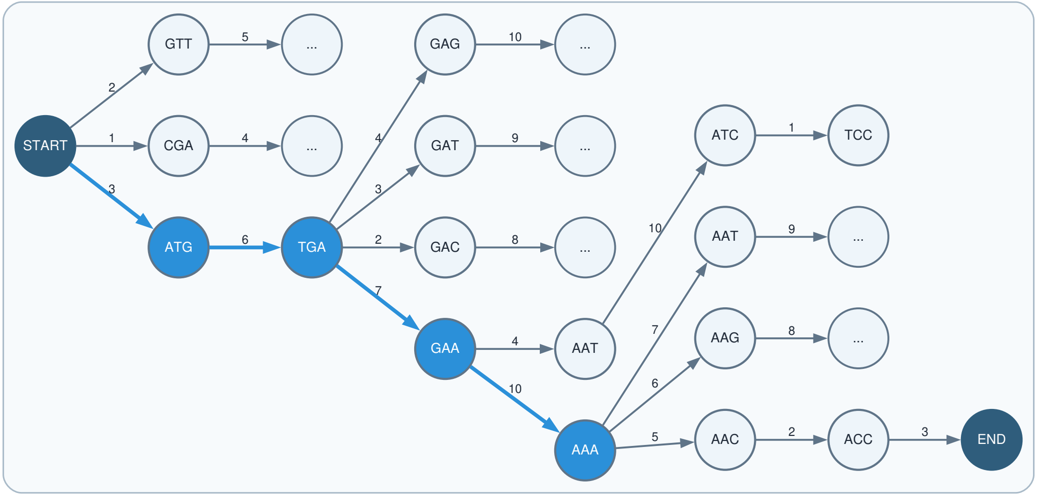
Length-restricted BeamSearch



Length-restricted BeamSearch



Length-restricted BeamSearch



Formulation and definition of BeamSearch

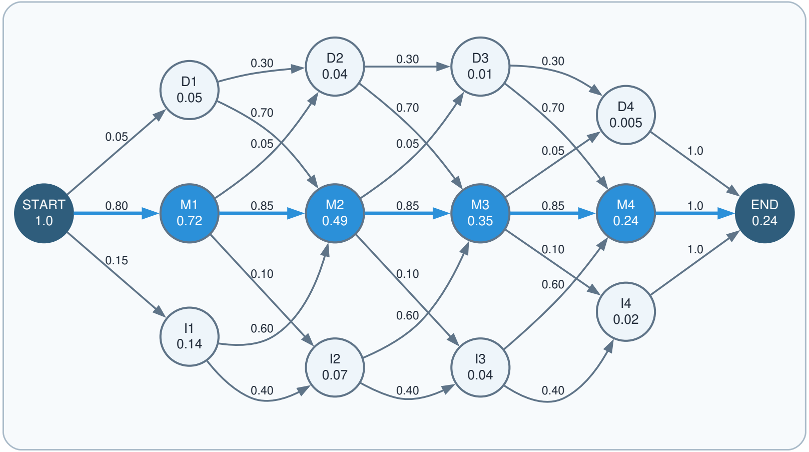
Beam Selection Criterion:

$$\mathcal{B}_N = \underset{\substack{P \subseteq \text{AllPaths} \\ |P|=N}}{\arg\max} \sum_{p \in P} \tau_h(p)$$

Algorithm

```
At each extension step:
  Evaluate each candidate path $p$ using quality metric $\tau_h(p)$
  Select the subset of $N$ paths that maximizes total score
  Maintain beam $\mathcal{B}_N$ of these top $N$ paths
  Retain only paths with the highest cumulative scores
```

Trellis graph



Formulation and definition of Viterbi/Trellis scoring

Viterbi Score Criterion: $$\tau_h(p) = \max_{s_1,\ldots,s_L} \sum_{i=1}^{L} \left[e_h(a_i|s_i) + t_h(s_i|s_{i-1}) \right], \quad h \in \mathcal{H}(\mathcal{T})$$

Algorithm

```
For each path $p$ from graph traversal:
  Translate nucleotide sequence to amino acids $a_1, \ldots, a_L$
  Fetch emission $e_h$ and transition $t_h$ scores from profile $h$
  Find state sequence maximizing total Viterbi score
  $\mathcal{H}(\mathcal{T})$: set of HMM profiles consistent with the current candidate type set $\mathcal{T}$
```

Formulation and definition of Viterbi/Trellis scoring

Viterbi Score Criterion: $$\tau_h(p) = \max_{s_1,\ldots,s_L} \sum_{i=1}^{L} \left[e_h(a_i|s_i) + t_h(s_i|s_{i-1}) \right], \quad h \in \mathcal{H}(\mathcal{T})$$

Algorithm

```
For each path $p$ from graph traversal:
  Translate nucleotide sequence to amino acids $a_1, \ldots, a_L$
  Fetch emission $e_h$ and transition $t_h$ scores from profile $h$
  Find state sequence maximizing total Viterbi score
  $\mathcal{H}(\mathcal{T})$: set of HMM profiles consistent with the current candidate type set $\mathcal{T}$
```

Assign $\tau_h(p)$ as maximum likelihood path score in BeamSearch!

Quality metric in BeamSearch is Viterbi score

$$\mathcal{B}_N = \underset{\substack{P \subseteq \text{AllPaths} \\ |P| = N}}{\arg\max} \sum_{p \in P} \tau_h(p), \quad h \in \mathcal{H}(\mathcal{T})$$

$$\tau_h(p) = \max_{s_1, \ldots, s_L} \sum_{i=1}^{L} \left[e_h(a_i | s_i) + t_h(s_i | s_{i-1}) \right]$$

$$p^*,\ h^* = \underset{\substack{p \in \mathcal{B}_N \\ h \in \mathcal{H}(\mathcal{T})}}{\arg\max}\, \tau_h(p)$$

Quality metric in BeamSearch is Viterbi score

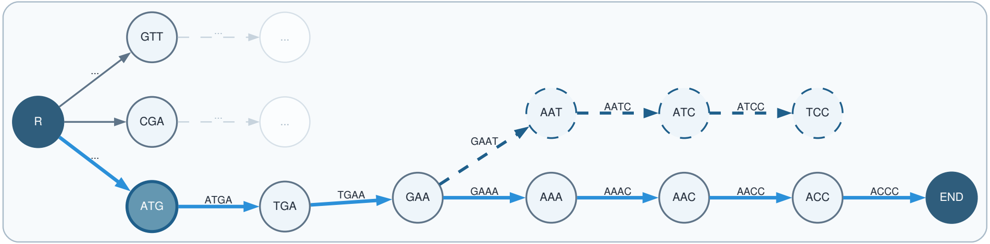
$$\mathcal{B}_N = \operatorname*{arg\,max}_{\substack{P \subseteq \text{AllPaths} \\ |P| = N}} \sum_{p \in P} \tau_h(p), \quad h \in \mathcal{H}(\mathcal{T})$$

$$\tau_h(p) = \max_{s_1,\ldots,s_L} \sum_{i=1}^{L} \left[e_h(a_i | s_i) + t_h(s_i | s_{i-1}) \right]$$

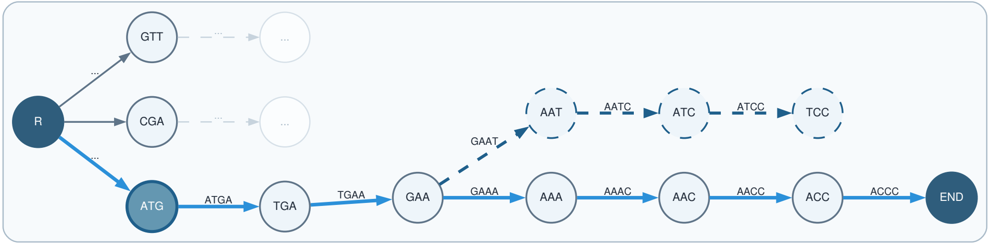
$$p^*,\ h^* = \operatorname*{arg\,max}_{\substack{p \in \mathcal{B}_N \\ h \in \mathcal{H}(\mathcal{T})}} \tau_h(p)$$

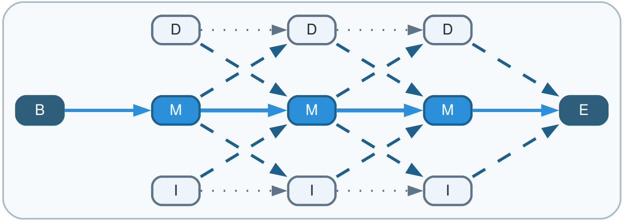
Beam search proposes paths;
Viterbi scoring ranks them.

BeamSearch + Trellis

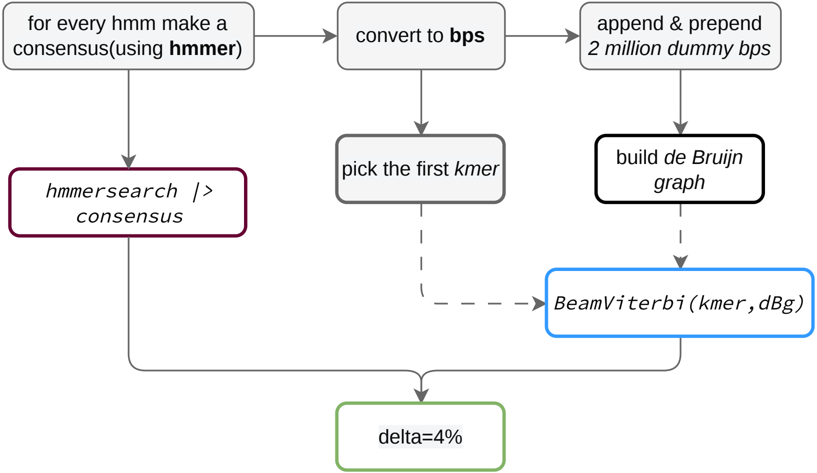


BeamSearch + Trellis





Does it work? How well does it perform?



for every hmm make a consensus(using **hmmer**)

convert to **bps**

append & prepend *2 million dummy bps*

hmmersearch |> consensus

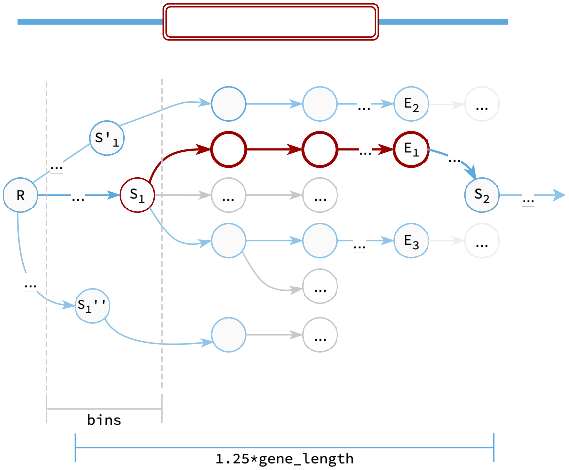
pick the first *kmer*

build *de Bruijn graph*

BeamViterbi(kmer,dBg)

delta=4%

Workflow



1. For $\mathcal{B}_N$, $N = 3$ paths in this example.
2. Two types of path rejection
    - Shallow score rejection: if $\tau_h \leq 0.05$(obvious false positive)
    - Path got rejected due to low τ_h among sibling paths.
3. The final p^*, h^* path and profile scored highest among all paths and profiles.

- Bins and breadth per search are introduced for the UPSTREAM detection case.

CAS detection workflow

1. Detect an anchor gene a from the CRISPR repeat on the SDBG in direction s
2. Initialize cassette $C = [a]$ and candidate type set $\mathcal{T}$ from typing rules
3. Extend cassette while gene count $|C| < L_{\max}$ and span $< B_{\max}$:
    - Search for next gene q on the graph using HMM profiles and candidate types
    - Compute intergenic gap g; if $g \notin [g_{\min}, g_{\max}]$, stop extension
    - Append q to C and narrow $\mathcal{T}$ using the detected family of q
4. Assign final cassette type $\tau^* = \arg\max_{\tau \in \mathcal{T}} \textsc{TypeScore}(\tau, C)$

CAS detection workflow

1. Detect an anchor gene a from the CRISPR repeat on the SDBG in direction s
2. Initialize cassette $C = [a]$ and candidate type set $\mathcal{T}$ from typing rules
3. Extend cassette while gene count $|C| < L_{\text{max}}$ and span $< B_{\text{max}}$:
   - Search for next gene q on the graph using HMM profiles and candidate types
   - Compute intergenic gap g; if $g \notin [g_{\text{min}}, g_{\text{max}}]$, stop extension
   - Append q to C and narrow $\mathcal{T}$ using the detected family of q
4. Assign final cassette type $\tau^* = \arg\max_{\tau \in \mathcal{T}} \textsc{TypeScore}(\tau, C)$

HARD-CODED TERMINATION

NO_NEXT_GENE, GAP_FAIL, or LIMIT_REACHED

Proof-of-concept

- **macsyfinder**
  - Synthetic metagenome consisting of the 24 genomes.
  - Number of repeats is 570(including reverse complement).
  - macsyfinder detected 95 CAS Subtypes.
- **MCAAT**
  - MCAAT detects 82% of the repeats.
  - CAS-Plugin detected and categorized 180 CAS-Subtypes.

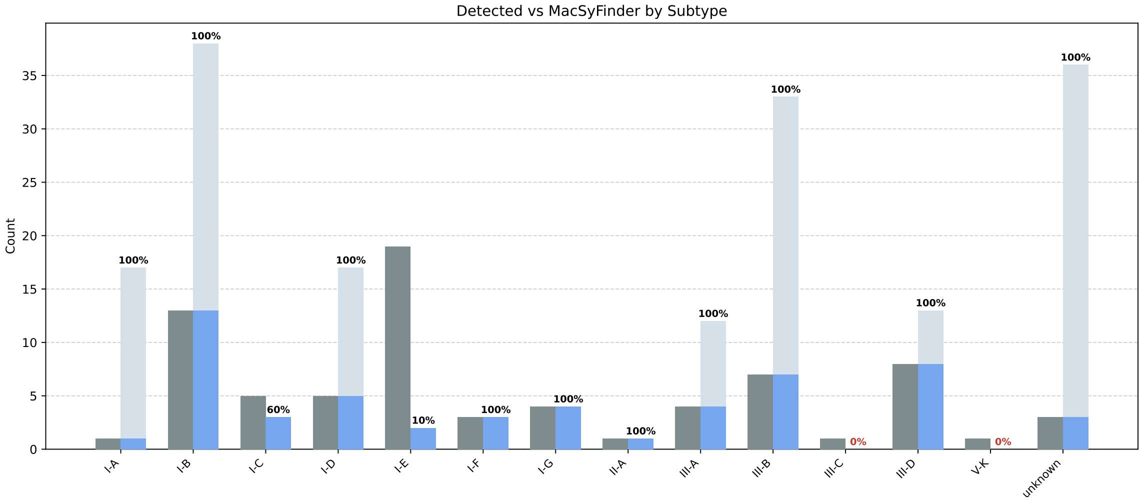
Proof-of-concept

- **macsyfinder**
  - Synthetic metagenome consisting of the 24 genomes.
  - Number of repeats is 570(including reverse complement).
  - macsyfinder detected 95 CAS Subtypes.
- **MCAAT**
  - MCAAT detects 82% of the repeats.
  - CAS-Plugin detected and categorized 180 CAS-Subtypes.

Results

75% **of the systems were detected and categorized correctly.**

Result


Detected vs MacSyFinder by Subtype
Count
0
5
10
15
20
25
30
35
I-A
I-B
I-C
I-D
I-E
I-F
I-G
II-A
III-A
III-B
III-C
III-D
V-K
unknown
100%
100%
60%
100%
10%
100%
100%
100%
100%
100%
0%
100%
0%
100%

Parameters and their effects

Parameter	Value	Effect
Beam width	100	Exploration–precision tradeoff; larger beam improves recall at higher runtime cost
Min. normalised score	0.15 b/pos	Bits per HMM position; gates out random-sequence matches
Shallow threshold	0.05	Fraction of full Viterbi score required in fast pre-scan; prunes bad candidates early
Intergenic gap	[−99, 2000] bp	Allowed nucleotide gap between consecutive genes; negative minimum permits slight overlaps
Max cassette span	60 000 bp	Hard upper bound on total cassette length; terminates runaway extensions
Max gene count	20	Max genes collected per cassette before `LIMIT_REACHED` is raised
First-gene window	[0, 7000] bp	BFS search range for anchor gene relative to the CRISPR repeat node

Progress, outlook and time

Achievements

- We are the first to provide a full **CRISPR-CAS** detection method for metagenomic datasets.
- We improved the filters and ordered spacers successfully in our new working version (special thanks to Max Warkentin).
- We have a new phage detection module that also works.

Progress, outlook and time

Achievements

- We are the first to provide a full **CRISPR-CAS** detection method for metagenomic datasets.
- We improved the filters and ordered spacers successfully in our new working version (special thanks to Max Warkentin).
- We have a new phage detection module that also works.

The evaluation lacks thorough investigation:

- How many genes are detected in total?
- Why so many false positives? Maybe parameter adjustment would help?
- How many genes belonging to each of the systems are detected correctly?
- I-E bothers me **a lot**. A substantial amount of genes from **I-E** *might* have been detected as **I-A** or **I-B**, due to rule similarity.

Thanks everybody!

Björn, Richard, Christoph, Charlene, Caro, Chris, Marius, Qian, Ubi, Max!

I will miss you all!