Ithaka: a taxonomic classifier based on Bloom filters, and spaced seeds

ZAKŁAD INFORMATYKI MEDYCZNEJ I TELEMEDYCYNY
WARSZAWSKIEGO UNIWERSYTETU MEDYCZNEGO

Maciej Sykulski, Karel Břinda, Gregory Kucherov
macieksk@gmail.com, karel.brinda@univ-mlv.fr, gregory.kucherov@univ-mlv.fr

Metagenomics & NGS

Metagenomics is a powerful approach to study genetic material contained in environmental samples, which is revolutionized by high-throughput sequencing technologies. Taxonomic classification of metagenomics data sets is a common step in analysis, a step for which computational cost becomes prohibitive with the growth of metagenomic datasets.

Approaches using sequence alignment algorithms, often based on the Burrows-Wheeler transform, such as Kaiju[1], Centrifuge[2], compete successfully with k-mer based alignment-free comparison methods such as Kraken[3], Clark[4]. Improvements to k-mer based approaches include: extending contiguous k-mers with spaced seeds [5][4], using Bloom filters as an underlying data structure for storing k-mers.[6]

Spaced Seeds

A **spaced seed** is a pattern over alphabet $A = \{\#, -\}$, where
$\#$ matching position, $-$ don't care position.

```
A A C A T T C T
  # # - # - #
A A C C T T C T
```

Previously[5] we demonstrated that spaced seeds allow for a better classification of NGS reads coming from a genome G between two other genomes G_1 and G_2 of the same genus.
Coverage is a number of aligned pairs covered by # from a spaced seed matches while sliding over an alignment (=4 above).
Coverage pattern is the pattern of matches (hits) of a seed over an alignment.
The NP-HARD PROBLEM: Assume a read f comes from genome g. For a given seed, computing $e_{f,g}$ – a minimal number of errors(mutations) in fragment f, given the *coverage pattern* of the read f with hits to g is NP-hard. In other words, answering what is the minimal number of errors in a fragment f which guarantees a certain coverage pattern (eg. found by Ithaka query) turns out to be an **instance of 0-1 integer programming** (an NP-complete problem). We programmed our solution using *coin-Cbc* integer programming library. In practice, it works fast on reads with high coverage, and slows down significantly on reads with low coverage (then heuristic is preferred).

References

[1] Menzel, P, Ng Kim Lee, Krogh A Fast and sensitive taxonomic classification for metagenomics with Kaiju. Nat Commun 7, Nature Publishing Group, apr 2016,

[2] Kim D, Song L, Breitwieser F P, Salzberg S Centrifuge: rapid and sensitive classification of metagenomic sequences Genome 1.2: 2.

[3] Wood, D. E. and Salzberg, S. L. Kraken: ultrafast metagenomic sequence classification using exact alignments. In Genome Biol., 15(3), R46. (2014)

[4] Ounit R, Lonardi S, Higher classification sensitivity of short metagenomic reads with CLARK-S bioRxiv 053462; doi: http://dx.doi.org/10.1101/053462

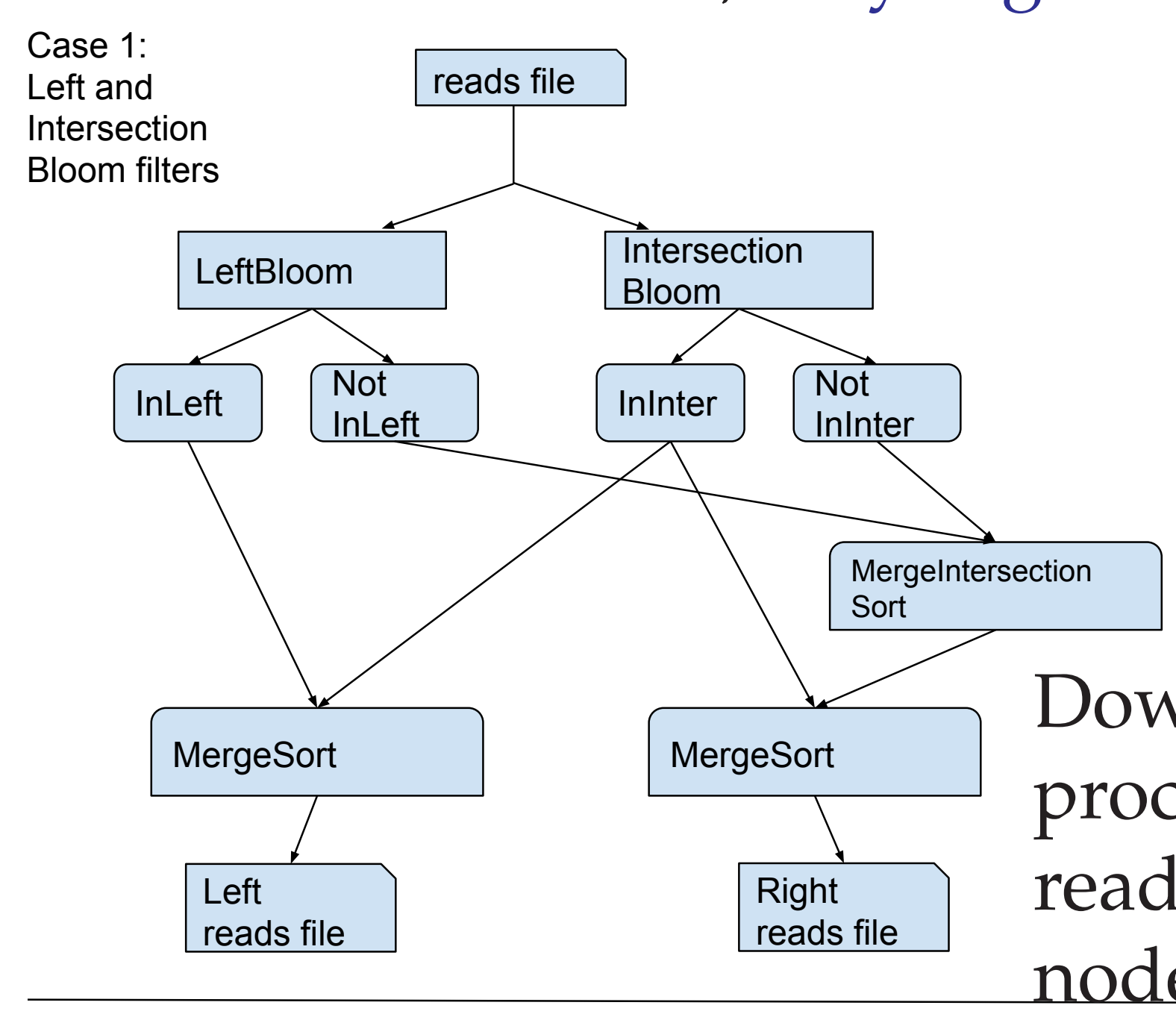
[5] Břinda, K,Sykulski M, Kucherov G. Spaced seeds improve k-mer-based metagenomic classification. Bioinformatics 31.22 (2015): 3584-3592.

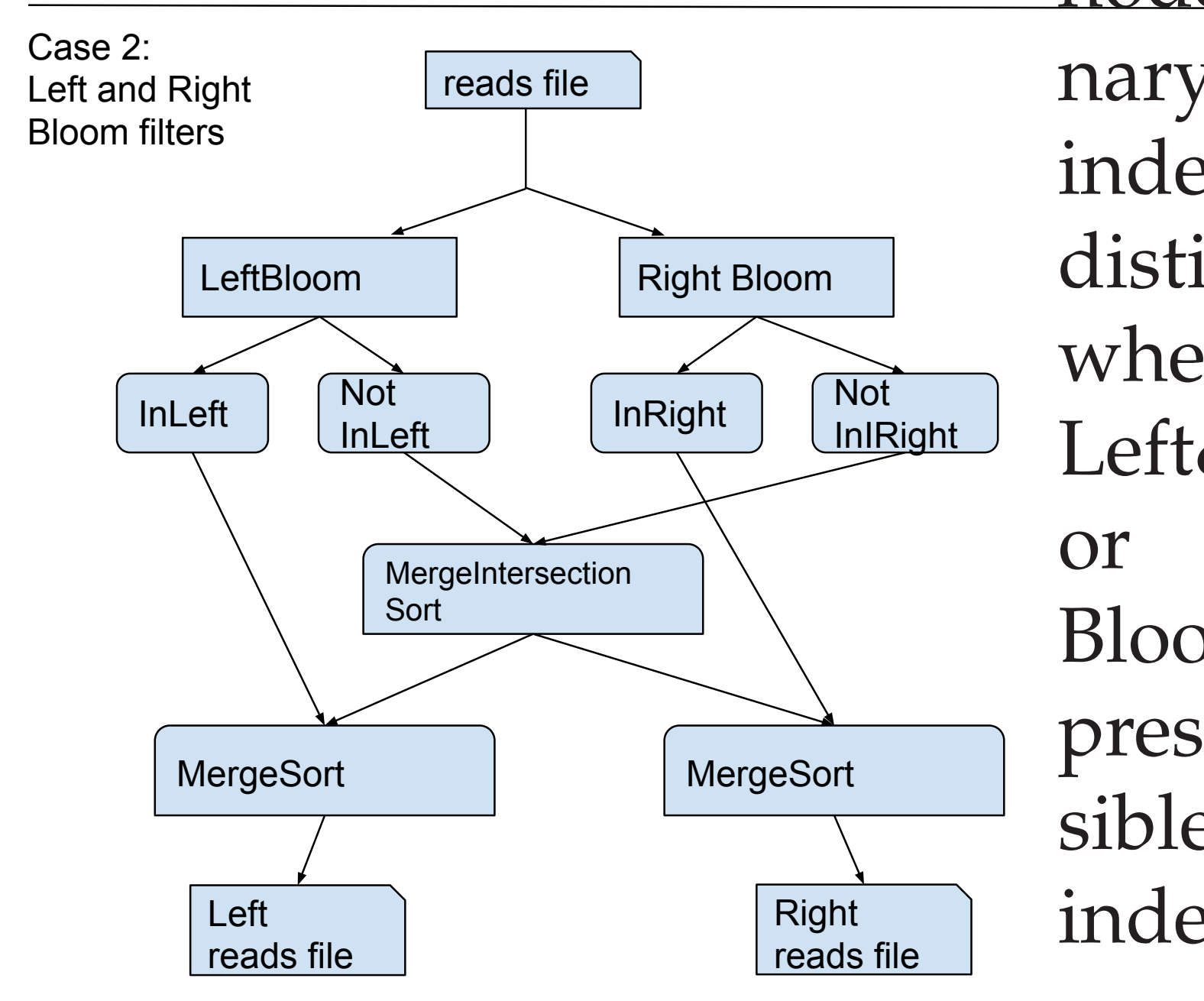
[6] Holley G, Wittler R, Stoye J. Bloom Filter Trie: an alignment-free and reference-free data structure for pan-genome storage. Algorithms for Molecular Biology. 2016;11: 3.

`https://github.com/gregorykucherov/ithaka`
`http://seed-kraken.readthedocs.org`

2outof3 Index

At each node of a **binary tree** we store 2 SMALLER OF OF 3 POSSIBLE p_{error}*–optimal in size* **Bloom filters**: only Left subtree elements, Intersection elements, only Right subtree elems..





Downstream processing of reads at each node of the binary 2outof3 index tree. Two distinct cases, when either Left&Intersection or Left&Right Bloom filters are present. It's possible to shrink index >2x.

Ithaka **2outof3 index** is comparable in size to Kraken: 67GB for bacterial and archaeal, but

- it requires 24 GB to be build (although it builds much faster when more is available) (Kraken requires 120GB to build the db)
- it requires <24 GB to run (!! Kraken req.75GB)
- it contains complete taxonomic information about k-mers (or seeds!), not only LCAs

EM abundance estimation

EM solves two problems at once: abundance estimation, and assignment. EM maximizes likelihood as a function of two types of variables:

1. abundances α_g such that $\sum_{g \in G} \alpha_g = 1$ correspond to proportion of reads from the whole sample which belong to genome g. Here G is the set of all genomes plus U category of unknown/unclassified reads.
2. categorical assignment indicator 0-1 variables $y_{f,g}$, which if equal to 1 mean that read f comes from a genome(or category) g. Matrix of these variables is sparse – most of them are 0 – since we only consider $y_{f,g}$ non zero if there are some hits from genome g in fragment f. $\sum_{g \in G} y_{f,g} = 1$

The **total likelihood** is proportional to

$$\left(\prod_{f \in F} \prod_{g \in G} P(f|g)^{y_{f,g}} \alpha_g^{y_{f,g}} \right) \times P_{prior}(\alpha)$$

$P_{prior}(\alpha)$ is the conjugate prior distribution to categorical distribution: the Dirichlet distribution.
Expectation step goes over $y_{f,g}$ variables (α_g set)
Maximization step goes over α_g variables ($y_{f,g}$ set).

$P(f|g)$ is given, it's probability of obtaining fragment f assuming it comes from genome g, it depends on $\frac{1}{\text{kmer\_richness\_of\_g}}$. Our proposal for it depends also on $e_{f,g}$ – a minimal number of errors/mutations in fragment f, given the coverage pattern (eg. 1111111111110010001101001011001100, see NP-HARD, left panel).

Ithaka Toolchain and Performance

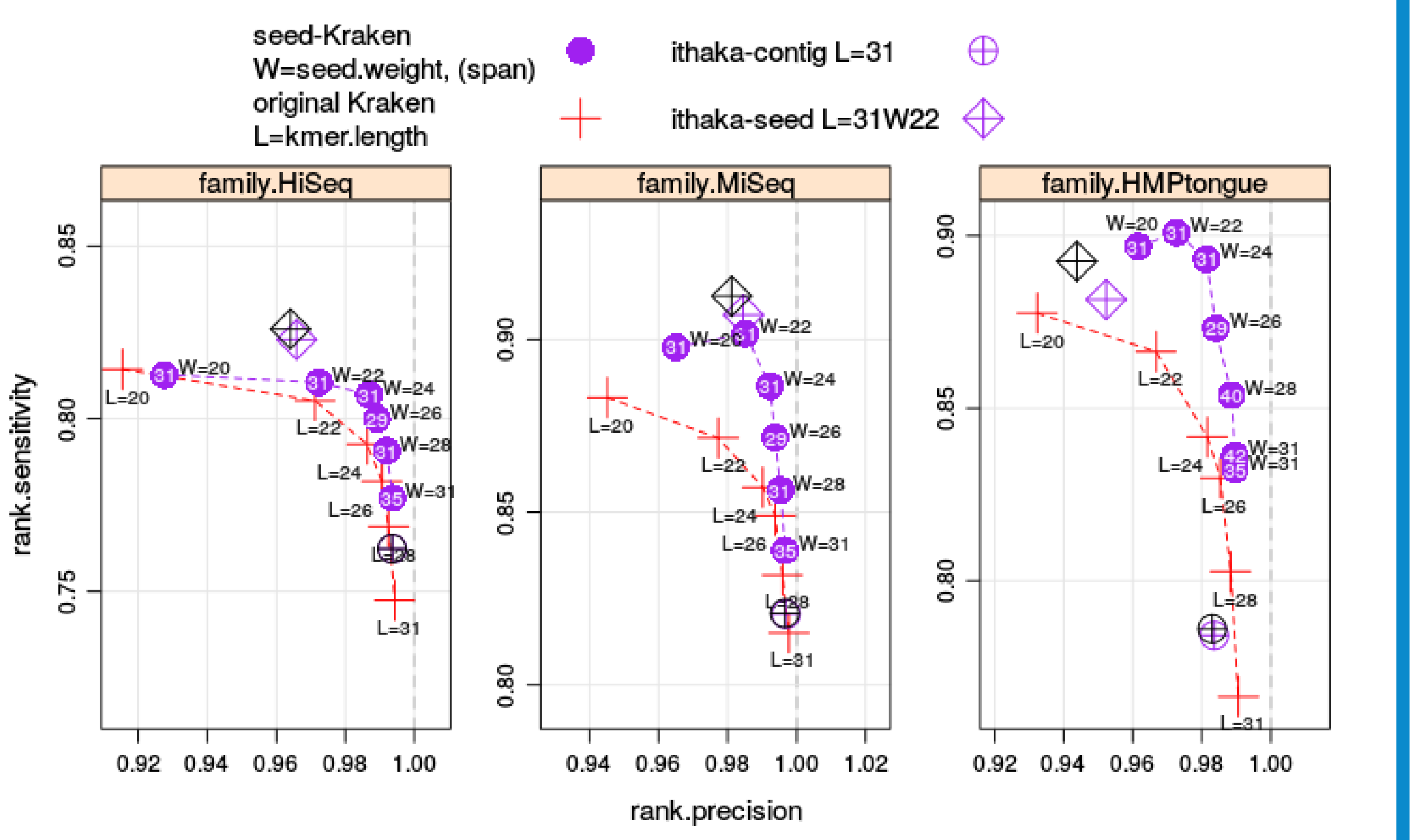
Ithaka tool chain:

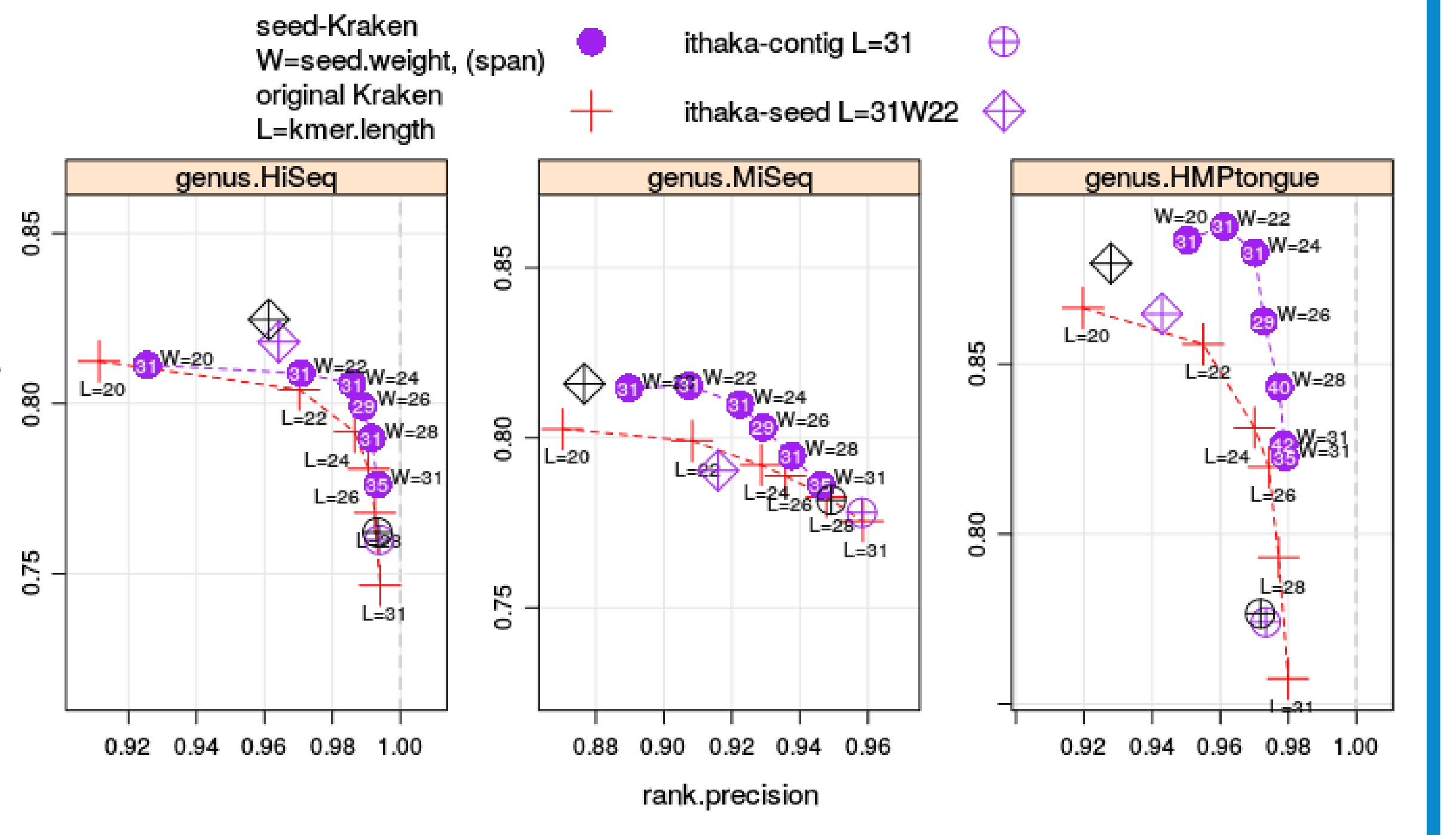
- ithaka-build.py – indexing fasta files, taxonomic tree binarization, creating of **2outof3 index**.
- ithaka-query.py – querying the index with a set of reads.
- ithaka-assigment.py – 2 steps:
  - samtoerr – computing error counts from coverage (NP-hard, *coin-Cbc* lib.),
  - Expectation Maximization(EM) – taxonomic abundance estimation, and read assignment (blazing fast: 1000 iterations of 50K reads (>2 million alignments) in <10 seconds)

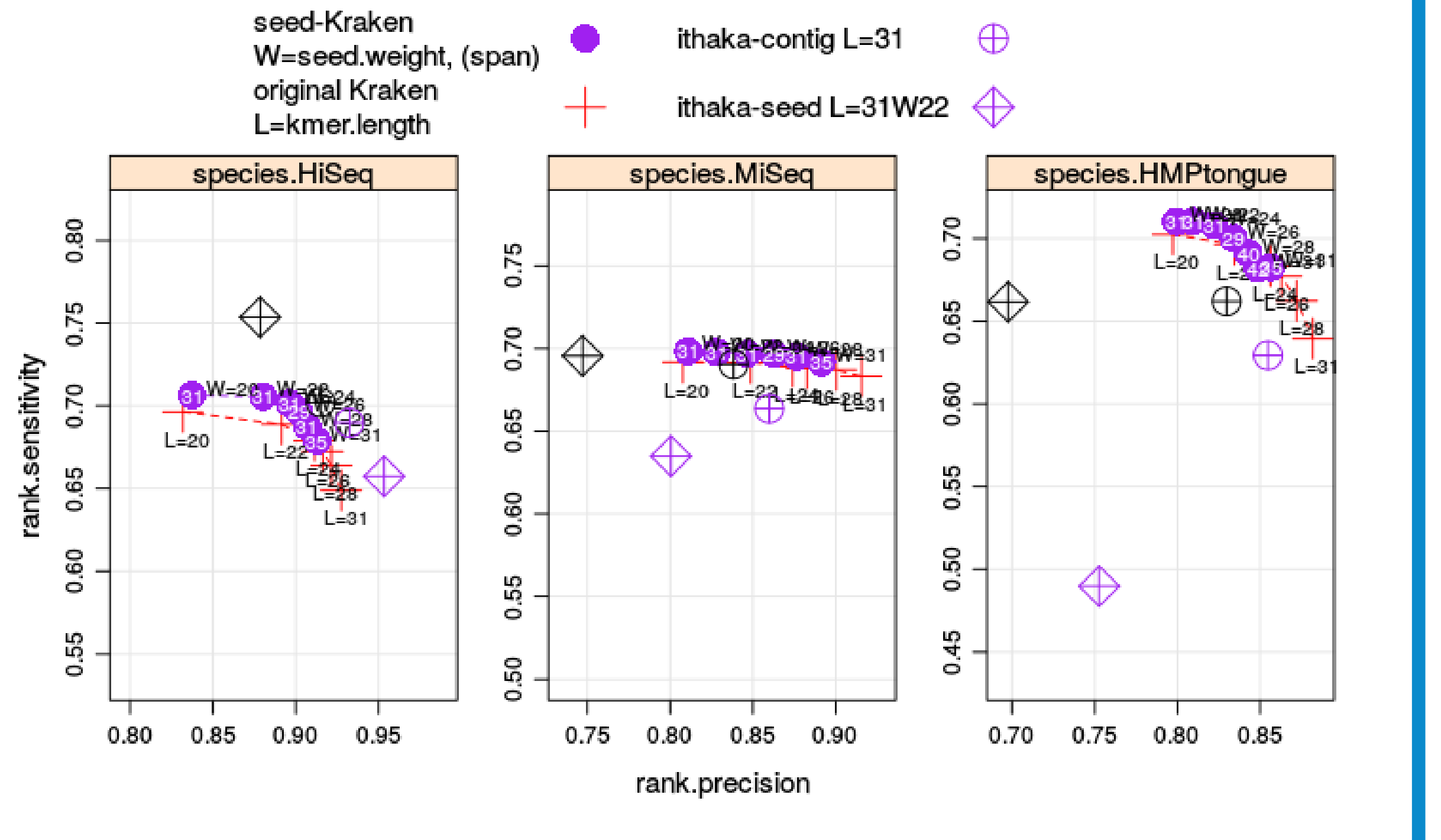
Possible improvements to ithaka-query.py: In development implementation (partially in C++) is still slow: 700 reads (of 220-250len) per second. This can be easily sped up 4x by moving totaly to C++. Further speed up is not possible without cache optimization because 25% of time is spent on querying Bloom filter bitmap: it's the time to access memory randomly with cache misses. Effective implementation using *"cache blocking"* is possible (10x, total 40x speedup).

Performances of classification of Seed-Kraken[5] and Ithaka (with spaced seeds), and original Kraken, were computed on simulated metagenomes (primarily used in [3]): HiSeq, MiSeq, and our HMP set HMPtongue.

Charted are rank precision (positive predictive value) against rank sensitivity (rate of correct assignments). Varying are *k-mer length*, and its spaced seed equivalent *seed weight*, while the *seed span* (in circles) varies from 31 to 40. **Ithaka marks** *black*: only leaves(genomes) fed to EM; *blue*: also inner nodes fed to EM (precision at the cost of sensitivity).


seed-Kraken W=seed.weight, (span)
original Kraken L=kmer.length
ithaka-contig L=31
ithaka-seed L=31W22
family.HiSeq
family.MiSeq
family.HMPtongue
rank.sensitivity
rank.precision


seed-Kraken W=seed.weight, (span)
original Kraken L=kmer.length
ithaka-contig L=31
ithaka-seed L=31W22
genus.HiSeq
genus.MiSeq
genus.HMPtongue
rank.sensitivity
rank.precision


seed-Kraken W=seed.weight, (span)
original Kraken L=kmer.length
ithaka-contig L=31
ithaka-seed L=31W22
species.HiSeq
species.MiSeq
species.HMPtongue
rank.sensitivity
rank.precision