

Optimum Search Schemes for Approximate String Matching Using Bidirectional FM-Index

Kiavash Kianfar¹
Christopher Pockrandt²
Bahman Torkamandi¹
Haochen Luo¹
Knut Reinert²

¹Texas A&M University, College Station, TX, USA



²Freie Universität, Berlin, Germany



slides by Kiavash Kianfar

Introduction

Approximate String Matching (ASM) problem

Given

- a **text** of length T ,
- a string (**read**) of length R ,
- an **alphabet** of size σ ,
- and number of **errors** (Hamming or edit distance), K ,

find substrings in the text whose (Hamming or edit) distance to the read is at most K .

Approximate String Matching (ASM) problem

Given

- a **text** of length T ,
- a string (**read**) of length R ,
- an **alphabet** of size σ ,
- and number of **errors** (Hamming or edit distance), K ,

find substrings in the text whose (Hamming or edit) distance to the read is at most K .

- Fundamental problem in computer science with numerous applications, especially in **Bioinformatics**

Approximate String Matching (ASM) problem

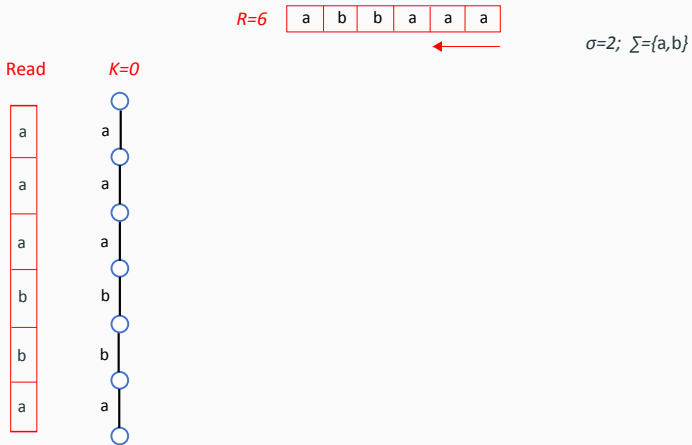
Given

- a **text** of length T ,
- a string (**read**) of length R ,
- an **alphabet** of size σ ,
- and number of **errors** (Hamming or edit distance), K ,

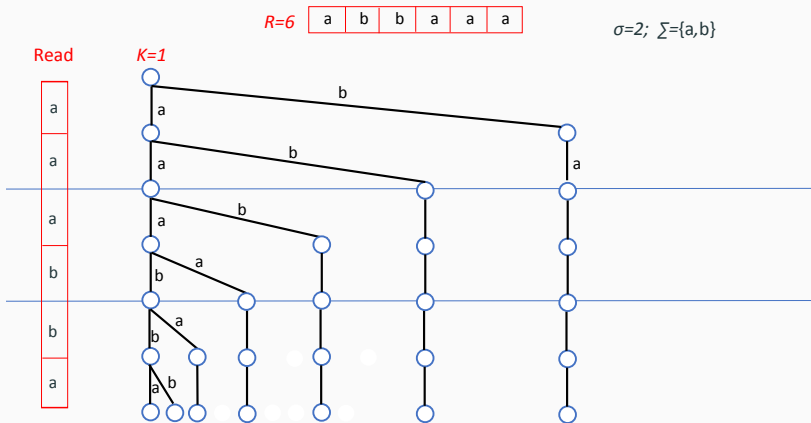
find substrings in the text whose (Hamming or edit) distance to the read is at most K .

- Fundamental problem in computer science with numerous applications, especially in **Bioinformatics**
- Indexing
 - (Enhanced) suffix array [Manber & Myers, 1990, Abouelhoda et al., 2004]
 - Affix array [Maaß, 2003, Strothmann, 2004]
 - **FM-index** [Ferragina & Manzini, 2000] based on **BWT** [Burrows & Wheeler, 1994]

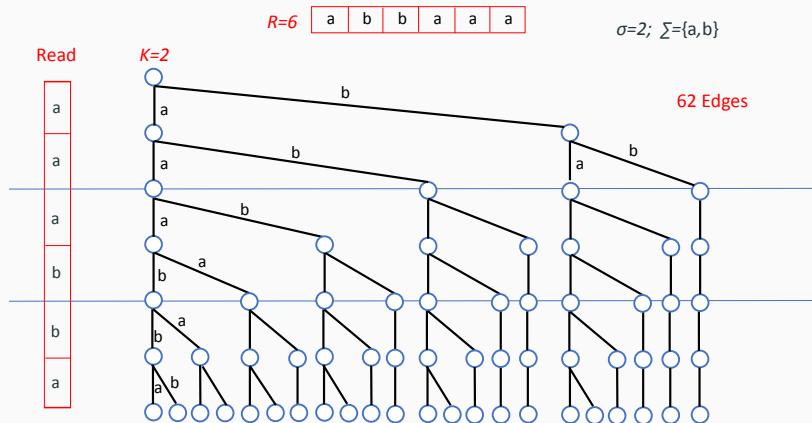
Search in FM-Index: Backtracking (Unidirectional)



Search in FM-Index: Backtracking (Unidirectional)

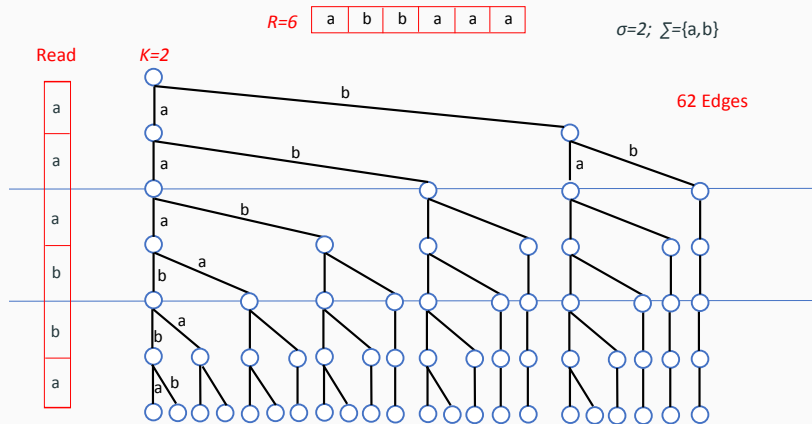


Search in FM-Index: Backtracking (Unidirectional)



- Assume Hamming distance
- Any path from root to a leaf represents a mismatch pattern.
- Assume the text contains all possible read mismatch patterns
- Then, total computational effort: Linear function of **number of edges**

Search in FM-Index: Backtracking (Unidirectional)



How to reduce the number of edges
but not miss any read mismatch pattern?

Bidirectional FM-Index and Search Schemes

[Lam et al., 2009]

- Introduced **bidirectional FM index**
- Search can start anywhere in the read and progress **left or right in any contiguous order**
- Proposed **partitioning** the read and searching them in different orders to cover all **partition mismatch patterns**

[Kucherov et al., 2016]

- Formalized and generalized this idea
- Introduced the concept of **Search Schemes**

Bidirectional FM-Index and Search Schemes

$K=2$

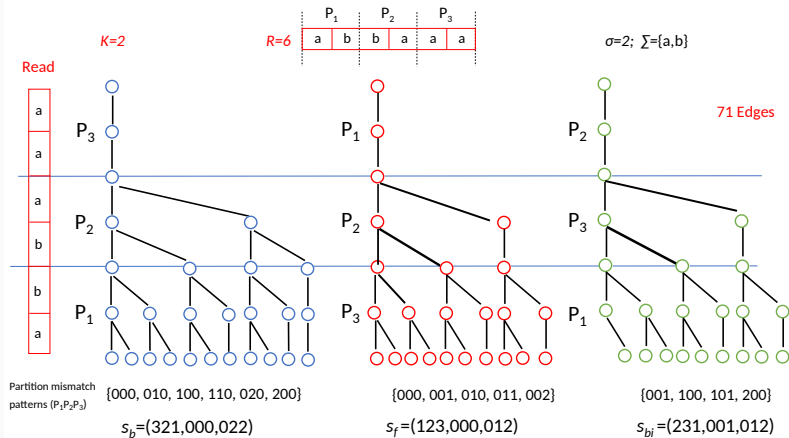
$R=6$

$\sigma=2; \Sigma=\{a,b\}$

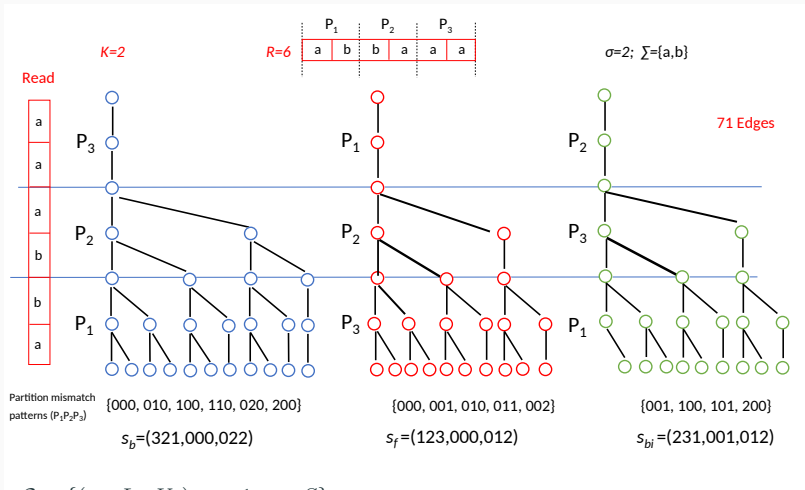
Partition
Mismatch
Patterns

P_1		P_2		P_3	
a	b	b	a	a	a
P_1		P_2		P_3	
0		0		0	
1		0		0	
0		1		0	
0		0		1	
1		1		0	
1		0		1	
0		1		1	
2		0		0	
0		2		0	
0		0		2	

Bidirectional FM-Index and Search Schemes



Bidirectional FM-Index and Search Schemes



- $\mathcal{S} = \{(\pi_s, L_s, U_s), s = 1, \dots, S\}$
- π_s : a permutation of $1, \dots, P$, where P is number of pieces in the partition
- L_s : lower bound on the cumulative No. of mismatches, (string of P numbers)
- U_s : upper bound on the cumulative No. of mismatches, (string of P numbers)

Optimal Search Scheme Problem

Optimal Search Scheme Problem

What is the search scheme that:

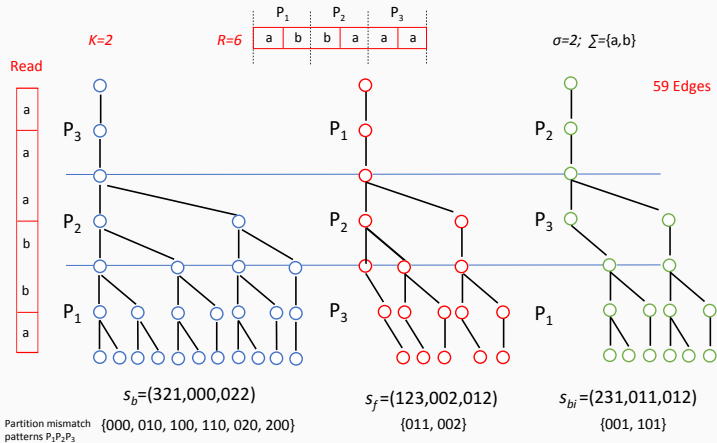
- minimizes number of steps in ASM-B (ASM using bidirectional FM-index)
- and ensures all partition mismatch patterns are covered?

MIP for Optimal Search Scheme Problem

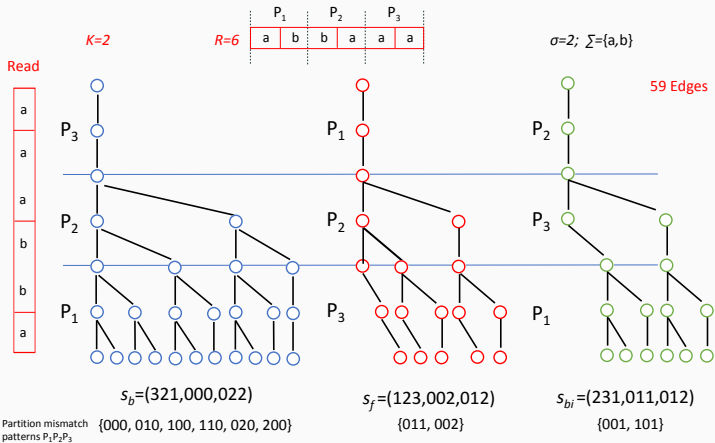
MIP to Solve Optimal Search Scheme Problem

- A general Mixed Integer Programming (MIP) approach to **directly solve** the optimization problem.
- finds the **exact optimal solution** to Optimal Search Scheme problem for **Hamming distance** and **equal-size pieces**.
- Takes K , R , P , and an upper bound on S , as input
- Provides (π_s, L_s, U_s) of all searches in the optimal scheme, as output
- It optimizes across all search schemes with at most S searches.

Optimal Scheme for Our Little Example



Optimal Scheme for Our Little Example



● 62 edges $\Rightarrow$ 71 edges $\Rightarrow$ 59 edges (Optimal)

Computational Comparison

Total Number of Edges: Backtracking vs. Optimal Scheme

Distance	Search Scheme	$K = 1$		$K = 2$		$K = 3$		$K = 4$	
		Edges	Factor	Edges	Factor	Edges	Factor	Edges	Factor
Hamming	Backtracking	15,554	1.00	1,560,854	1.00	116,299,379	1.00	6,862,924,649	1.00
	Optimal ($P = K + 1$)	8,004	0.51	892,769	0.57	67,888,328	0.58	4,064,852,156	0.59
	Optimal ($P = K + 2$)	8,922	0.57	854,303	0.55	65,116,676	0.56	3,916,700,994	0.57
	Optimal ($P = K + 3$)	8,004	0.51	835,213	0.54	64,060,718	0.55	3,887,857,820	0.57
Edit	Backtracking	41,208	1.00	11,154,036	1.00	2,264,515,748	1.00	367,846,294,116	1.00
	Optimal ($P = K + 1$)	20,908	0.51	6,315,779	0.57	1,299,709,022	0.57	213,296,122,595	0.58
	Optimal ($P = K + 2$)	23,356	0.57	6,025,907	0.54	1,246,126,103	0.55	205,509,484,572	0.56
	Optimal ($P = K + 3$)	20,908	0.51	5,892,667	0.53	1,226,903,544	0.54	203,270,363,390	0.55

Computational Comparison

Total Number of Edges: Backtracking vs. Optimal Scheme

Distance	Search Scheme	$K = 1$		$K = 2$		$K = 3$		$K = 4$	
		Edges	Factor	Edges	Factor	Edges	Factor	Edges	Factor
Hamming	Backtracking	15,554	1.00	1,560,854	1.00	116,299,379	1.00	6,862,924,649	1.00
	Optimal ($P = K + 1$)	8,004	0.51	892,769	0.57	67,888,328	0.58	4,064,852,156	0.59
	Optimal ($P = K + 2$)	8,922	0.57	854,303	0.55	65,116,676	0.56	3,916,700,994	0.57
	Optimal ($P = K + 3$)	8,004	0.51	835,213	0.54	64,060,718	0.55	3,887,857,820	0.57
Edit	Backtracking	41,208	1.00	11,154,036	1.00	2,264,515,748	1.00	367,846,294,116	1.00
	Optimal ($P = K + 1$)	20,908	0.51	6,315,779	0.57	1,299,709,022	0.57	213,296,122,595	0.58
	Optimal ($P = K + 2$)	23,356	0.57	6,025,907	0.54	1,246,126,103	0.55	205,509,484,572	0.56
	Optimal ($P = K + 3$)	20,908	0.51	5,892,667	0.53	1,226,903,544	0.54	203,270,363,390	0.55

- Reduction in total number of edges up to a factor of 0.51

Computational Comparison

Total Number of Edges: Backtracking vs. Optimal Scheme

Distance	Search Scheme	$K = 1$		$K = 2$		$K = 3$		$K = 4$	
		Edges	Factor	Edges	Factor	Edges	Factor	Edges	Factor
Hamming	Backtracking	15,554	1.00	1,560,854	1.00	116,299,379	1.00	6,862,924,649	1.00
	Optimal ($P = K + 1$)	8,004	0.51	892,769	0.57	67,888,328	0.58	4,064,852,156	0.59
	Optimal ($P = K + 2$)	8,922	0.57	854,303	0.55	65,116,676	0.56	3,916,700,994	0.57
	Optimal ($P = K + 3$)	8,004	0.51	835,213	0.54	64,060,718	0.55	3,887,857,820	0.57
Edit	Backtracking	41,208	1.00	11,154,036	1.00	2,264,515,748	1.00	367,846,294,116	1.00
	Optimal ($P = K + 1$)	20,908	0.51	6,315,779	0.57	1,299,709,022	0.57	213,296,122,595	0.58
	Optimal ($P = K + 2$)	23,356	0.57	6,025,907	0.54	1,246,126,103	0.55	205,509,484,572	0.56
	Optimal ($P = K + 3$)	20,908	0.51	5,892,667	0.53	1,226,903,544	0.54	203,270,363,390	0.55

- Reduction in total number of edges up to a factor of 0.51
- Optimal solution for Hamming distance has the same improvement effect for Edit distance problem.

Computational Comparison

Total Number of Edges: Backtracking vs. Optimal Scheme

Distance	Search Scheme	$K = 1$		$K = 2$		$K = 3$		$K = 4$	
		Edges	Factor	Edges	Factor	Edges	Factor	Edges	Factor
Hamming	Backtracking	15,554	1.00	1,560,854	1.00	116,299,379	1.00	6,862,924,649	1.00
	Optimal ($P = K + 1$)	8,004	0.51	892,769	0.57	67,888,328	0.58	4,064,852,156	0.59
	Optimal ($P = K + 2$)	8,922	0.57	854,303	0.55	65,116,676	0.56	3,916,700,994	0.57
	Optimal ($P = K + 3$)	8,004	0.51	835,213	0.54	64,060,718	0.55	3,887,857,820	0.57
Edit	Backtracking	41,208	1.00	11,154,036	1.00	2,264,515,748	1.00	367,846,294,116	1.00
	Optimal ($P = K + 1$)	20,908	0.51	6,315,779	0.57	1,299,709,022	0.57	213,296,122,595	0.58
	Optimal ($P = K + 2$)	23,356	0.57	6,025,907	0.54	1,246,126,103	0.55	205,509,484,572	0.56
	Optimal ($P = K + 3$)	20,908	0.51	5,892,667	0.53	1,226,903,544	0.54	203,270,363,390	0.55

- Reduction in total number of edges up to a factor of 0.51
- Optimal solution for Hamming distance has the same improvement effect for Edit distance problem.
- Not all read mismatch patterns in reference genome; Search-in-index speed-up achieved by optimal search schemes for real data can be much more significant.

Computational Comparison

Search in Index for **all** occurrences of 100,000 Illumina reads ($R = 101$)
in human reference genome (hg38)

Dist.	Search Tool	$K = 1$		$K = 2$		$K = 3$	
		Time	Factor	Time	Factor	Time	Factor
Hamm.	Backtracking	22.80s	1.00	269.24s	1.00	2417.06s	1.00
	Optimal-scheme bidirect. ($P = K + 1$)	7.73s	2.95	19.78s	13.61	74.62s	32.39
	Optimal-scheme bidirect. ($P = K + 2$)	7.39s	3.09	18.81s	14.31	68.69s	35.19
Edit	Backtracking	43.59s	1.00	1245.70s	1.00	27889.40s	1.00
	Optimal-scheme bidirect. ($P = K + 1$)	11.21s	3.89	120.70s	10.32	1338.61s	20.83
	Optimal-scheme bidirect. ($P = K + 2$)	10.66s	4.09	112.23s	11.10	1307.23s	21.33

Computational Comparison

Search in Index for **all** occurrences of 100,000 Illumina reads ($R = 101$)
in human reference genome (hg38)

Dist.	Search Tool	$K = 1$		$K = 2$		$K = 3$	
		Time	Factor	Time	Factor	Time	Factor
Hamm.	Backtracking	22.80s	1.00	269.24s	1.00	2417.06s	1.00
	Optimal-scheme bidirect. ($P = K + 1$)	7.73s	2.95	19.78s	13.61	74.62s	32.39
	Optimal-scheme bidirect. ($P = K + 2$)	7.39s	3.09	18.81s	14.31	68.69s	35.19
Edit	Backtracking	43.59s	1.00	1245.70s	1.00	27889.40s	1.00
	Optimal-scheme bidirect. ($P = K + 1$)	11.21s	3.89	120.70s	10.32	1338.61s	20.83
	Optimal-scheme bidirect. ($P = K + 2$)	10.66s	4.09	112.23s	11.10	1307.23s	21.33

- Speed-up of up to 35 times

Computational Comparison

Search in Index for **all** occurrences of 100,000 Illumina reads ($R = 101$)
in human reference genome (hg38)

Dist.	Search Tool	$K = 1$		$K = 2$		$K = 3$	
		Time	Factor	Time	Factor	Time	Factor
Hamm.	Backtracking	22.80s	1.00	269.24s	1.00	2417.06s	1.00
	Optimal-scheme bidirect. ($P = K + 1$)	7.73s	2.95	19.78s	13.61	74.62s	32.39
	Optimal-scheme bidirect. ($P = K + 2$)	7.39s	3.09	18.81s	14.31	68.69s	35.19
Edit	Backtracking	43.59s	1.00	1245.70s	1.00	27889.40s	1.00
	Optimal-scheme bidirect. ($P = K + 1$)	11.21s	3.89	120.70s	10.32	1338.61s	20.83
	Optimal-scheme bidirect. ($P = K + 2$)	10.66s	4.09	112.23s	11.10	1307.23s	21.33

- Speed-up of up to 35 times
- Hamming distance optimal scheme very effective for edit distance as well.

- Can we eliminate the input variables S and P
- Improve MIP formulation (cuts, symmetry elimination)
- Is the optimal solution always a partition of the mismatch patterns?
- What is the best point to stop search in index and start in-text verification?

Optimal Search Schemes

Optimal search schemes are available in our paper for community use

	$K = 1$	$K = 2$	$K = 3$	$K = 4$
Optimal ($P = K + 1$)	(12, 00, 01) (21, 01, 01)	(123, 002, 012) (321, 000, 022) (231, 011, 012)	(1234, 0003, 0233) (2341, 0000, 1223) (3421, 0022, 0033)	(12345, 00004, 03344) (23451, 00000, 22334) (54321, 00033, 00444)
Optimal ($P = K + 2$)	(123, 001, 001) (321, 000, 011)	(2134, 0011, 0022) (3214, 0000, 0112) (4321, 0002, 0122)	(12345, 00022, 00333) (43215, 00000, 11223) (54321, 00003, 02233)	(123456, 000004, 033344) (234561, 000000, 222334) (654321, 000033, 004444)
Optimal ($P = K + 3$)	(1234, 0000, 0011) (4321, 0001, 0011)	(21345, 00011, 00222) (43215, 00000, 00112) (54321, 00002, 01122)	(123456, 000003, 022233) (234561, 000000, 111223) (654321, 000022, 003333)	(1234567, 0111111, 3333334) (1234567, 0000000, 0044444) (7654321, 0000004, 0333344)

Thank you!

Thanks to

- Texas A&M University High Performance Computing Facility
- Max-Planck Research School for Computational Biology and Scientific Computing

References i

- [Abouelhoda et al., 2004] Abouelhoda, M.I., Kurtz, S., Ohlebusch, E.: Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms* 2(1) (2004) 53–86
- [Burrows & Wheeler, 1994] Burrows, M., Wheeler, D.J.: A block-sorting lossless data compression algorithm. Technical Report 124, Digital SRC Research Report (1994)
- [Ferragina & Manzini, 2000] Ferragina, P., Manzini, G.: Opportunistic data structures with applications. In: FOCS '00. (2000) 390–398
- [Kucherov et al., 2016] Kucherov, G., Salikhov, K., Tsur, D.: Approximate string matching using a bidirectional index. *Theoretical Computer Science* **638** (2016) 145–158
- [Lam et al., 2009] Lam, T.W., Li, R., Tam, A., Wong, S., Wu, E., Yiu, S.M.: High throughput short read alignment via bi-directional bwt. In: IEEE BIBM '09. 31–36
- [Li & Durbin, 2009] Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 25(14) (2009) 1754–1760
- [Maaß, 2003] 11. Maaß, M.G.: Linear bidirectional on-line construction of affix trees. *Algorithmica* 37(1) (2003) 43–74

- [Manber & Myers, 1990] Manber, U., Myers, E.W.: Suffix arrays: a new method for on-line string searches. In: SODA '90. (1990) 319–327
- [Pockrandt et al., 2017] Pockrandt, C., Ehrhardt, M., Reinert, K.: EPR-Dictionaries: A Practical and Fast Data Structure for Constant Time Searches in Unidirectional and Bidirectional FM Indices. In: RECOMB '17. (2017) 190–206
- [Reinert et al., 2015] Reinert, K., Langmead, B., Weese, D., Evers, D.J.: Alignment of Next-Generation Sequencing Reads. *Annual review of genomics and human genetics* 16 (2015) 133–151
- [Siragusa, 2015] Siragusa, E.: Approximate string matching for high-throughput sequencing. PhD thesis, Freie Universität Berlin (2015)
- [Strothmann, 2004] Strothmann, D.: The affix array data structure and its applications to RNA secondary structure analysis. *Theoretical Computer Science* 389(1-2) (2007) 278–294
- [Vroland et al., 2016] Vroland, C., Salson, M., Bini, S., Touzet, H.: Approximate search of short patterns with high error rates using the 01*0 lossless seeds. *Journal of Discrete Algorithms* (2016) 3–16

MIP Main Variables

- $n_{s,l,d}$: Number of edges at level l of the trie of search s with d cumulative mismatches
- $x_{s,i,j}$: Assignment of piece j to position (iteration) i in search s
- $L_{s,i}$: Lowerbound on number of cumulative mismatches for search s at iteration i
- $U_{s,i}$: Upperbound on number of cumulative mismatches for search s at iteration i

MIP Objective Function

$$\min \sum_{s=1}^{\bar{S}} \sum_{l=1}^R \sum_{d=0}^K n_{s,l,d}$$

- Minimize total number of edges.
- [Kucherov et al., 2016] presented a weighting of edges assuming read and text are randomly and independently drawn from the alphabet.
- For ASM of DNA sequence reads to reference genomes, this is far from reality.
- Currently, it is not known how to weight, so we used total number of edges.
- Our MIP can be easily modified to incorporate any weighting scenario.

MIP Constraints

subject to

$$\sum_{i=1}^P x_{s,i,j} = 1 \quad \text{for all } s \text{ and } j \quad (1)$$

$$\sum_{j=1}^P x_{s,i,j} = 1 \quad \text{for all } s \text{ and } i \quad (2)$$

$$\sum_{h=1}^i x_{s,h,j} - \sum_{h=1}^i x_{s,h,j-1} = t_{s,i,j}^+ - t_{s,i,j}^- \quad \text{for all } s, i = 2, \dots, P-1, j = 1, \dots, P+1 \quad (3)$$

$$\sum_{j=1}^{P+1} (t_{s,i,j}^+ + t_{s,i,j}^-) = 2 \quad \text{for all } s, i = 2, \dots, P-1 \quad (4)$$

- Constraints (1)-(2) make sure one-to-one assignment of pieces to iterations.
- Constraints (3)-(4) ensure connectivity of pieces.

MIP Constraints

$$d - (L_{s, \lceil \frac{l}{m} \rceil} - m \lceil \frac{l}{m} \rceil + l) + 1 \leq (R + 1)z_{s,l,d} \quad \text{for all } s, l, \text{ and } d \quad (5)$$

$$U_{s, \lceil \frac{l}{m} \rceil} + 1 - d \leq (K + 1)\bar{z}_{s,l,d} \quad \text{for all } s, l, \text{ and } d \quad (6)$$

$$\binom{l}{d}(\sigma - 1)^d(\bar{z}_{s,l,d} + z_{s,l,d} - 2) \leq n_{s,l,d} - n_{s,l-1,d} - (\sigma - 1)n_{s,l-1,d-1} \quad \text{for all } s, l, \text{ and } d \quad (7)$$

$$L_{s,i} \leq L_{s,i+1} \quad \text{for all } s, \text{ and } i = 1, \dots, P - 1 \quad (8)$$

$$U_{s,i} \leq U_{s,i+1} \quad \text{for all } s, \text{ and } i = 1, \dots, P - 1 \quad (9)$$

- Constraints (5)-(7) enforce calculation of $n_{s,l,d}$ based on recursive equation $n_{s,l,d} = n_{s,l,d} + (\sigma - 1)n_{s,l-1,d-1}$ [Kuchеров et al., 2016].
- Constraints (8)-(9) ensure $L_{s,i}$ and $U_{s,i}$ are non-decreasing.

MIP Constraints

$$L_{s,i} + K(\lambda_{q,s} - 1) \leq \sum_{h=1}^i \sum_{j=1}^P a_{q,j} x_{s,h,j} \leq U_{s,i} + K(1 - \lambda_{q,s}) \quad \text{for all } q, s, \text{ and } i \quad (10)$$

$$\sum_{s=1}^{\bar{S}} \lambda_{q,s} \geq 1 \quad \text{for all } q \quad (11)$$

$$x_{1PP} = 1 \quad (12)$$

$$\sum_{t=s}^{\bar{S}} \sum_{k=1}^{j-1} x_{t,1,k} \leq (\bar{S} - s + 1)(1 - x_{s,1,j}) \quad \text{for all } s \text{ and } j = 2, \dots, P \quad (13)$$

$$\sum_{j=1}^{P-i+1} x_{sij} + \sum_{j=i}^P x_{sij} = 1 \quad \text{for all } s \text{ and } i \geq \lceil P/2 \rceil + 1 \quad (14)$$

$$n_{s,l,d} \geq 0 \quad \text{for all } q, s, i, j, l, \text{ and } d \quad (15)$$

$$L_{s,i}, U_{s,i} \geq 0 \quad \text{Integer} \quad \text{for all } s \text{ and } i \quad (16)$$

$$x_{s,i,j}, \lambda_{q,s}, \bar{z}_{s,l,d}, \underline{z}_{s,l,d}, t_{s,i,j}^+, t_{s,i,j}^- \in \{0, 1\} \quad \text{for all } q, s, i, j, l, \text{ and } d \quad (17)$$

- Constraints (10)-(11) ensure search scheme covers all partition mismatch patterns.
- Constraints (12)-(14) eliminate some symmetry in solution space.
- Constraints (15)-(17) set the sign and type of variables.