Boyer Moore exact string matching algorithm variations

Milica Vlasonjić 3094/2021
School of Electrical Engineering, Belgrade

Heuristics 1 – Horspool Sunday 2[1]

- Idea: when a mismatch occur at any position then the shift value is determined by Next-to-last character and Last character of text corresponding to pattern, that is *T[i+m]* and *T[i+m-1]* where m is length of pattern
- Algorithm:
  - Start matching pattern from right to left
  - If mismatch occurs at any position than consider Next-to-Last character (*T[i+m]*) of text (let's call it *x*) and find its position in pattern:
    1) If not in pattern than right shift by *m + 1*.
    2) If occurs at first position than right shift by *m*.
    3) If occurs other than first position, then consider Last character of pattern (let's call it *y*) and search for *yx* sequence:
       - If *yx* occurs in P right shift by last occurrence of *y* + 1.
       - Else right shift by *m + 1*.

[1] Lin quan Xie, Xiao ming Liu and Guangxue Yue, "Improved Pattern Matching Algorithm of BMHS", 978-0-7695-4360-4/10 2010 IEEE DOI 10.1109/ISISE..2010.154

Heuristics 1 – Horspool Sunday 2

	S	T	R	I	N	G	M	A	T	C	H	I	N	G	I	S	T	O	F	I	N	D	T	H	E	P	A	T	T	E	R	N
1	P	A	T	T	E	R	N																									
2									P	A	T	T	E	R	N																	
3																	P	A	T	T	E	R	N								
4																									P	A	T	T	E	R	N	
5																										P	A	T	T	E	R	N

- Time complexity: worst time $O(mn)$, best time $O(n/m)$, average $O(n)$
- Space complexity: $O(n)$~$O(n)$
- Preprocessing: time $O(m)$; space $O(m)$

Heuristics 2 – Composite Rule[2]

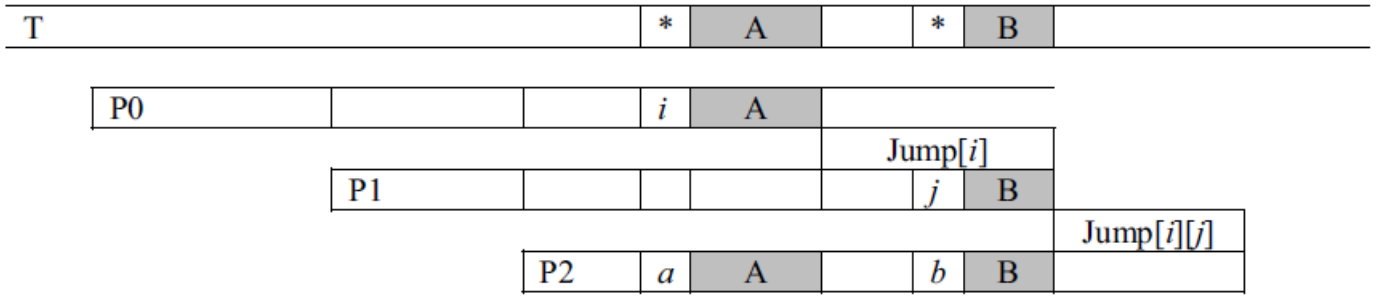
- Idea: use the comparison history achieved at previous iteration
- Example:
  - For the pattern *P=101101*, suppose the mismatch appears at the end of P at previous iteration, so the character *a* is not *1*. Suppose the mismatch also appears at the end of *P* at current iteration, so the character *b* is not *1* either. According to BM, *P* should right shift for *1* character. But, in fact, the results of the comparison of the two iterations show that none of the characters in P that correspond to *a* and *b* are *1*. For *P* does not consist continuous two characters that are not *1*, *P* can be right moved out of the previous position, that is, it can right shift for 6 characters.

T	*	*	*	*	*	*	*a*	*b*	*	*	*	*	*	*	*
Location for P at previous iteration		1	0	1	1	0	1								
Location for P at current iteration			1	0	1	1	0	1							
Shift location for P according to BM				1	0	1	1	0	1						
Improved shift location for P									1	0	1	1	0	1	

[2] Zhengda Xiong," *A Composite Boyer-Moore Algorithm for the String Matching Problem"*, 978-0- 7695-4287-4/10 2010 IEEE, DOI 10.1109/PDCAT.2010.58

Heuristics 2 – Composite Rule

- Algorithm: construct a 2D array *Jump[m][m]*, where *Jump[i][j]* is the shift distance of pattern P when the mismatch occurs at *P[j]* at current iteration and at *P[i]* at previous iteration
- Only pattern is needed for construction of *Jump* table

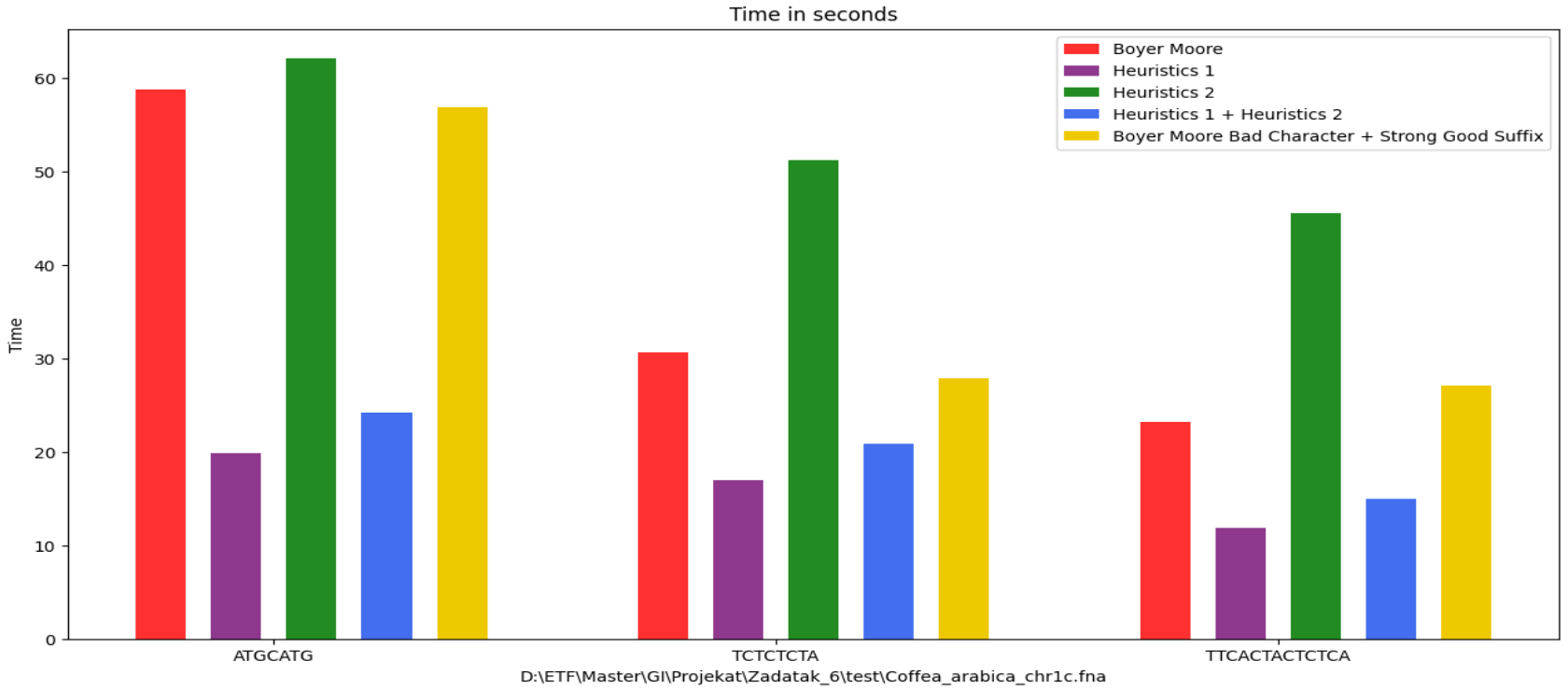


Heuristics 2 – Composite Rule

```
JUMPS(pattern, i, j):
        jmp ← strong_good_suffix_rule(j)
        m ←  len(pattern)
        isMatch ← True
        for jump in range(jmp + 1, m + 1):
            isMatch ← True
            k ← m - 1
            while k > j and k >= jump:
                if p[k] ≠ p[k - jump]
                    isMatch ← False
                    break
                k ← k - 1
            if not isMatch:
                continue
            if (j ≥ jump) and (p[j] = p[j - jump]):
                continue
            isMatch ← True
            delta ← jump + SJump[i]
            k ← m - 1
            while k > i and k >= delta:
                if p[k] ≠ p[k - delta]:
                    isMatch = False
                    break
                k ← k - 1
            if not isMatch:
                continue
            if (i ≥ delta) and (p[i] = p[i - delta]):
                continue
            return jump
        return jump
```

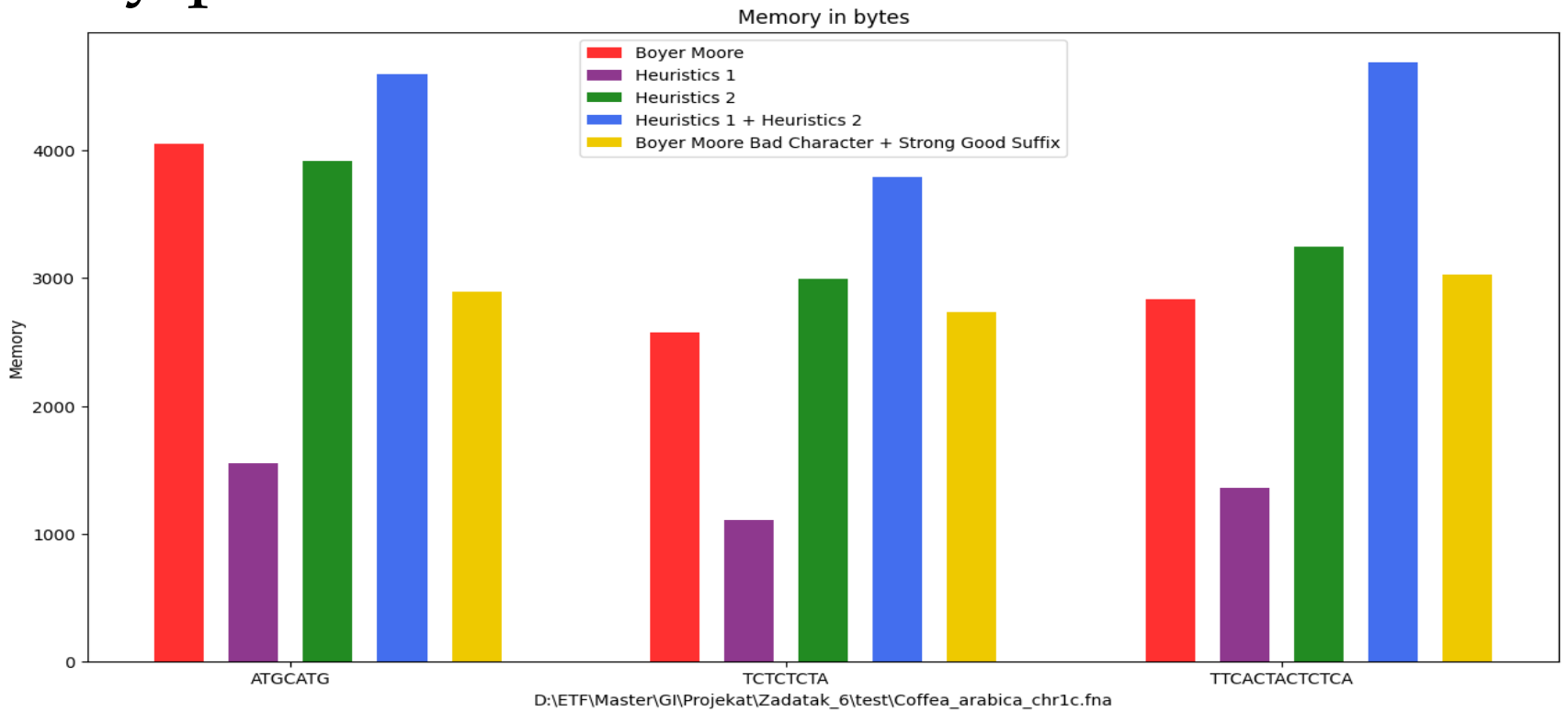
- Time complexity: worst time $O(mn)$, best time $O(n/m)$, average $O(n)$
- Space complexity: $O(n + m^2) \sim O(n)$
- Preprocessing: time $O(m^2)$; space $O(m^2)$

Time Performance – Coffea Arabica chr1c



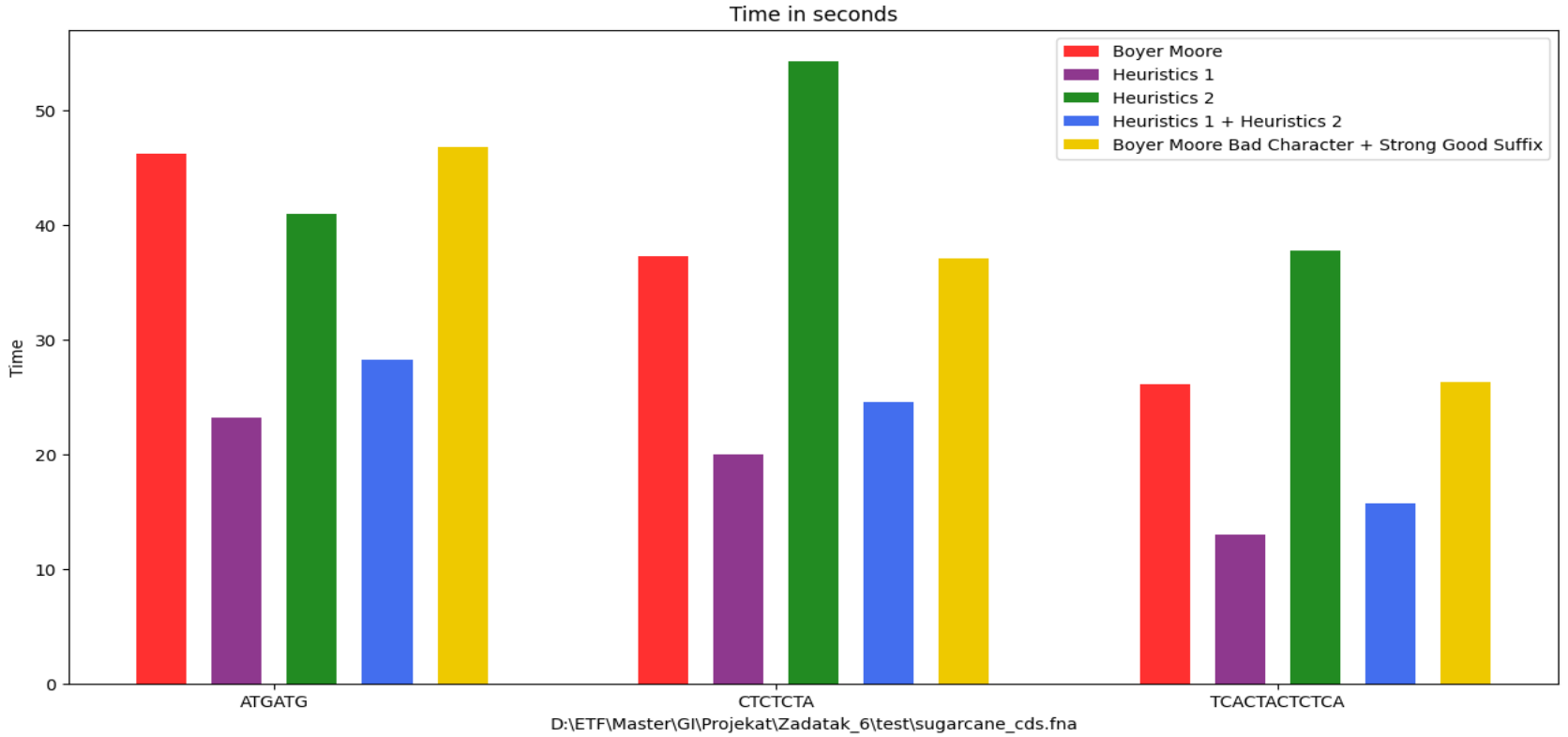
	BM	Heur 1	Heur 2	Heur 1 + Heur 2	Strong BM
ATGCATG	58.76	19.9	62.1	24.19	56.87
TCTCTCTA	30.63	16.98	51.25	20.86	27.84
TTCACTACTCTCA	23.27	11.86	45.57	14.97	27.14

Memory performance - Coffea Arabica chr1c



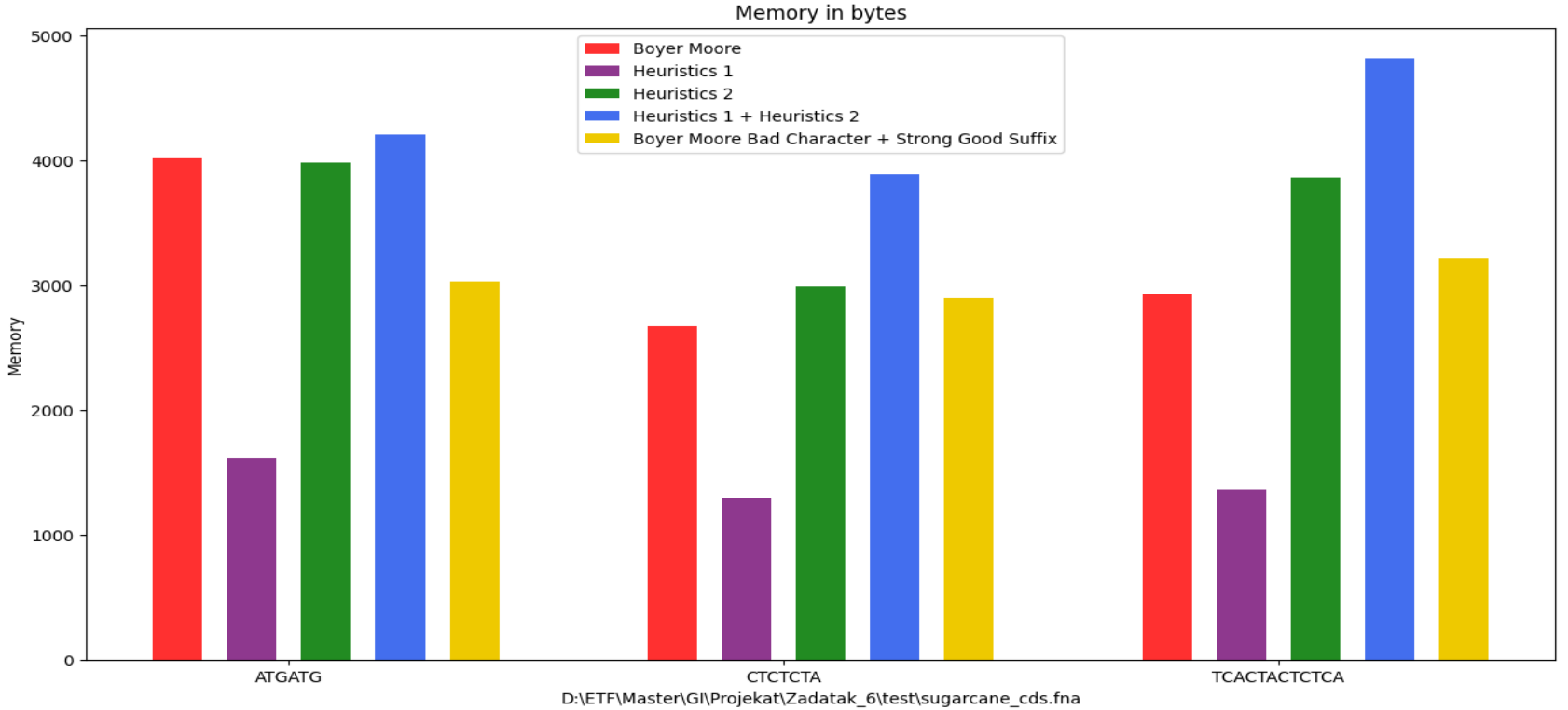
	BM	Heur 1	Heur 2	Heur 1 + Heur 2	Strong BM
ATGCATG	4048.0	1584.0	4496.0	4576.0	2960.0
TCTCTCTA	2640.0	1104.0	2992.0	3824.0	2800.0
TTCACTACTCTCA	2896.0	1360.0	3280.0	4752.0	3088.0

Time Performance – Sugarcane



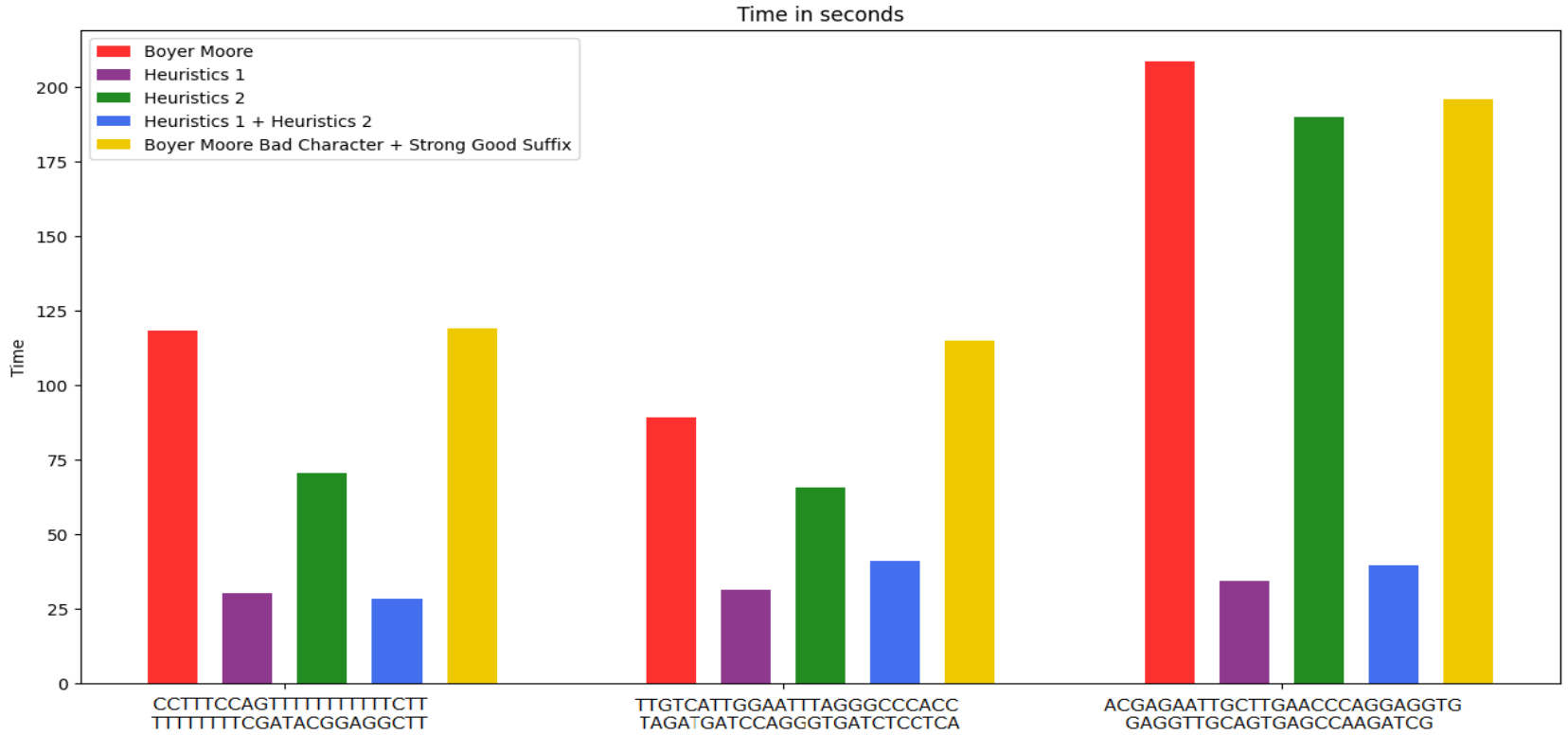
	BM	Heur 1	Heur 2	Heur 1 + Heur 2	Strong BM
ATGATG	46.18	23.18	40.97	28.25	46.83
CTCTCTA	37.29	19.99	54.26	24.54	37.09
TCACTACTCTCA	26.15	12.97	37.75	15.73	26.27

Memory performance - Sugarcane



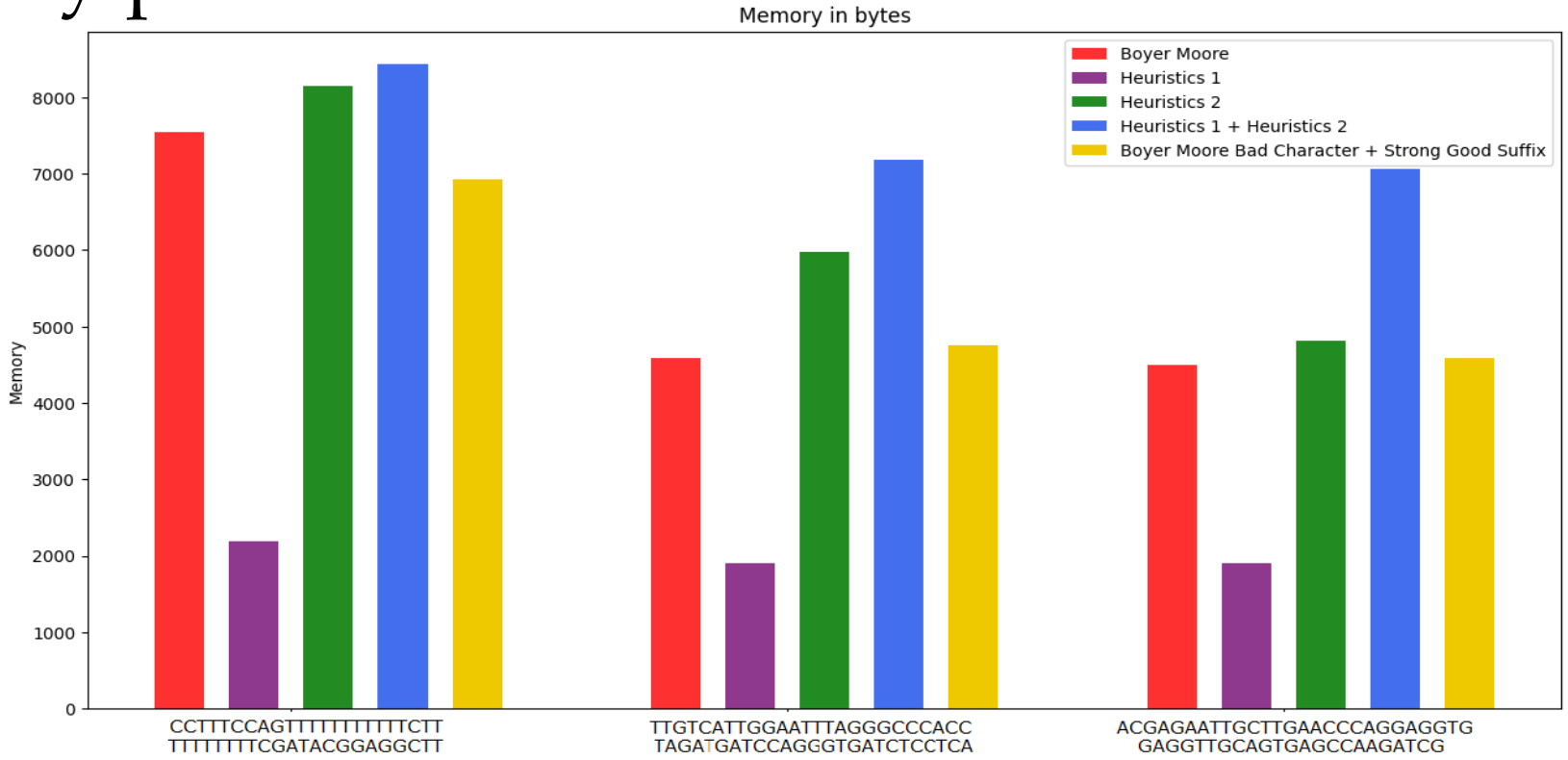
	BM	Heur 1	Heur 2	Heur 1 + Heur 2	Strong BM
ATGATG	4016.0	1616.0	3984.0	4208.0	3024.0
CTCTCTA	2672.0	1296.0	2992.0	3888.0	2896.0
TCACTACTCTCA	2928.0	1360.0	3856.0	4816.0	3216.0

Time Performance – Human Genome chrX



	BM	Heur 1	Heur 2	Heur 1 + Heur 2	Strong BM
CCTTTCCAGTTTTTTTTTTCTTTTTTTTTTCGATACGGAGGCTT	118.33	30.31	70.64	28.55	119.03
TTGTCATTGGAATTTAGGGCCCACCTAGATGATCCAGGGTGATCTCCTCA	89.16	31.4	65.66	41.1	114.98
ACGAGAATTGCTTGAACCCAGGAGGTGGAGGTTGCAGTGAGCCAAGATCG	208.45	34.28	189.92	39.51	195.77

Memory performance – Human Genome chrX



	BM	Heur 1	Heur 2	Heur 1 + Heur 2	Strong BM
CCTTTCCAGTTTTTTTTTTCTTTTTTTTTTCGATACGGAGGCTT	7536.0	2192.0	8144.0	8432.0	6928.0
TTGTCATTGGAATTTAGGGCCCACCTAGATGATCCAGGGTGATCTCCTCA	4592.0	1904.0	5968.0	7184.0	4752.0
ACGAGAATTGCTTGAACCCAGGAGGTGGAGGTTGCAGTGAGCCAAGATCG	4496.0	1904.0	4816.0	7056.0	4592.0

Thank you for your attention