Client side web based application for search space reduction in approximate circular pattern matching

Costas S. Iliopoulos[1], M. Samiruzzaman[1], M Sohel Rahman[2] and Steven Watts[1]

[1] Department of Informatics, King's College London, Strand,WC2R 2LS, England
[2] Department of CSE, Bangladesh University of Engineering & Technology, Dhaka-1215, Bangladesh

Abstract. This paper deals with a client side development approach for Approximate Circular Pattern Matching (ACPM) problem. ACPM appears as an interesting problem in many biological contexts. ACPM is to find all possible occurrences of the rotations of a pattern $\mathcal{P}$ of length m in a text $\mathcal{T}$ of length n with k mismatch. We have developed a lightweight and fast web based tools for users. Much of the speed of our approach can be attributed to the fact that actual computation is done in client machine instead of uploading big chunks of data in the server. We have uploaded our web tools on http://www.icloudberry.com/upload/index.html [1]. The source codes including the input patterns and text files are available in our Github repository https://github.com/steven31415/circular-string-filter [2]

1 Our Contribution

We are extending the work from [4] and [5] by developing a client based tools for circular pattern matching. The main contribution of this paper is a fast and efficient browser based tools which dramatically reduces the size of data by using filter based approximate circular pattern matching approach. The idea behind our approach is quite simple and intuitive. The users do not need to install any software or do not need to upload the big file in the server. Our development approach works with big data in client side by using lower memory working as chunk by chunk without uploading any data to web server. Instead of program running in the web server, the java script code is transported in the browser in run-time and does the computation. The user doesn't require to install any software either because the the computation is done in the browser.

2 Filtering Algorithm and our browser based client side approach

The circular pattern, denoted $\mathcal{C}(\mathcal{P})$, corresponding to a given pattern $\mathcal{P} = \mathcal{P}_1 \dots \mathcal{P}_m$, is formed by connecting $\mathcal{P}_1$ with $\mathcal{P}_m$ and forming a cycle. Due to

the size restriction of our short abstract, the readers are referred to read in details from [3, 6–13] and references therein.

We consider the DNA alphabet, i.e., $\Sigma = \{a, c, g, t\}$. We assign the numbers from the range $[1...|\Sigma|]$ to the characters of Σ following their inherent lexicographical order. We use $num(x), x \in \Sigma$ to denote the numeric value of the character x. So, we have $num(a) = 1, num(c) = 2, num(g) = 3$ and $num(t) = 4$,. For a string S, we use the notation S_N to denote the numeric representation of the string S; and $S_N[i]$ denotes the numeric value of the character $S[i]$. So, if $S[i] = g$ then $S_N[i] = num(g) = 3$. The concept of circular string and their rotations also apply naturally on their numeric representations as is illustrated in Example 1 below.

Example 1. Suppose we have a pattern $\mathcal{P} = atcgatg$. The numeric representation of $\mathcal{P}$ is $\mathcal{P}_N = 1423143$. And this numeric representation has the following rotations: $\mathcal{P}_N^1 = 4231431$, $\mathcal{P}_N^2 = 2314314$, $\mathcal{P}_N^3 = 3143142$, $\mathcal{P}_N^4 = 1431423$, $\mathcal{P}_N^5 = 4314231$, $\mathcal{P}_N^6 = 3142314$.

Filter 1 We define the function *sum* on a string $\mathcal{P}$ of length m as follows: $sum(\mathcal{P}) = \sum_{i=1}^{m} P_N[i]$. Our first filter, Filter 1, is based on this *sum* function. We have the following observation.

Observation 1 *Consider a circular string $\mathcal{P}$ and a linear string $\mathcal{T}$ both having length n. If $\mathcal{C}(\mathcal{P}) \equiv_k \mathcal{T}$, where $0 \leq k < n$, then we must have*

$$sum(\mathcal{T}) - k \times 4 + k \times 1 \leq sum(\mathcal{C}(\mathcal{P})) \leq sum(\mathcal{T}) + k \times 4 - k \times 1.$$

Example 2. Consider $\mathcal{P} = atcgatg$. We can easily calculate that $sum(\mathcal{C}(\mathcal{P})) = 18$. Now, consider $\mathcal{T}1 = aacgatg$, slightly different from $\mathcal{P}$, i.e, $\mathcal{P}[2] = t \neq \mathcal{T}1[2] = a$. As can be easily verified, here $\mathcal{P} \equiv_1 \mathcal{T}1$. According to Observation 1, in this case the lower (upper) bound is 15 (18). Indeed, we have $\mathcal{T}1_N = 1123143$ and $sum(\mathcal{T}1) = 15$, which is within the bounds. Now consider $\mathcal{T}2 = ttcgatg$, slightly different from $\mathcal{P}$, i.e, $\mathcal{P}[1] = a \neq \mathcal{T}2[1] = t$. As can be easily verified, here $\mathcal{P} \equiv_1 \mathcal{T}2$. Therefore, in this case as well, the lower and upper bound mentioned above hold. And indeed we have $\mathcal{T}2_N = 4423143$ and $sum(\mathcal{T}2) = 21$, which is within the bounds. Finally, consider another string $\mathcal{T}' = atagctg$. It can be easily verified that $\mathcal{C}(\mathcal{P}) \not\equiv_1 \mathcal{T}'$. Again, the previous bounds hold in this case and we find that $\mathcal{T}'_N = 1413243$ and $sum(\mathcal{T}') = 18$. Clearly this is within the bounds of Observation 1 and in fact it is exactly equal to $sum(\mathcal{C}(\mathcal{P}))$. This is an example of a false positive with respect to Filter 1.

3 Experimental Results

We have implemented our approximate sum filtering algorithm so that it may process data on the client size, eliminating the need for data upload. We conduct experiments on various text sizes and pattern sizes (m), recording the time taken to perform approximate sum filtering. An approximation parameter k is additionally used to specify the level of approximation acceptable in the filtering process. The final size indicates the reduced text search space after filtering.

Table 1. Average elapsed times for upload and logic (algorithm processing time) of the approximate sum filter for varying pattern sizes (considering three approximation parameters k) over a text of size 250MB.

	k = 0			k =1			k = 2		
m	Load	Logic	Final Size	Load	Logic	Final Size	Load	Logic	Final Size
(kb)	(s)	(s)	(mb)	(s)	(s)	(mb)	(s)	(s)	(mb)
2	0	31.4	1.31	0	31.1	9.16	0	30.5	17.01
5	0	30.0	0.87	0	30.2	6.08	0	30.0	11.28
10	0	31.5	0.84	0	30.9	5.87	0	30.0	10.91

Table 2. Average elapsed times for upload and logic (algorithm processing time) of the approximate sum filter for varying pattern sizes (considering three approximation parameters k) over a text of size 500MB.

	k = 0			k =1			k = 2		
m	Load	Logic	Final Size	Load	Logic	Final Size	Load	Logic	Final Size
(kb)	(s)	(s)	(mb)	(s)	(s)	(mb)	(s)	(s)	(mb)
2	0	60.0	2.62	0	62.0	18.32	0	60.4	34.00
5	0	61.8	1.75	0	60.4	12.23	0	62.3	22.70
10	0	59.0	1.66	0	61.0	11.61	0	60.9	21.55

Table 3. Average elapsed times for upload and logic (algorithm processing time) of the approximate sum filter for varying pattern sizes (considering three approximation parameters k) over a text of size 750MB.

	k = 0			k =1			k = 2		
m	Load	Logic	Final Size	Load	Logic	Final Size	Load	Logic	Final Size
(kb)	(s)	(s)	(mb)	(s)	(s)	(mb)	(s)	(s)	(mb)
2	0	91.9	3.92	0	98.4	27.41	0	96.9	50.90
5	0	97.2	2.61	0	98.9	18.32	0	99.5	34.01
10	0	97.2	2.47	0	96.6	17.28	0	88.0	32.11

Table 4. Average elapsed times for upload and logic (algorithm processing time) of the approximate sum filter for varying pattern sizes (considering three approximation parameters k) over a text of size 1000MB.

	k = 0			k =1			k = 2		
m	Load	Logic	Final Size	Load	Logic	Final Size	Load	Logic	Final Size
(kb)	(s)	(s)	(mb)	(s)	(s)	(mb)	(s)	(s)	(mb)
2	0	120.2	5.23	0	123.2	36.57	0	127.3	67.90
5	0	123.2	3.49	0	124.7	24.41	0	127.0	45.32
10	0	125.6	3.30	0	127.5	23.16	0	125.0	43.03

4 Conclusions

We presented a web based tool for an effective lightweight filtering technique to reduce the search space of the Approximate Circular Pattern Matching (ACPM) problem. We worked on 1 GB data but it can easily go higher because the memory doesn't load the data at a time. Our algorithm works as chunk by chunk and performs the computation. Much of the speed of our algorithm comes from the fact the user does not need to upload the data to the web server. At this moment our tool is producing significantly reduced size of text. Future works include to work on more lightweight filters as well as enhancing our current web based tool.

References

1. `http://www.icloudberry.com/upload/index.html`.
2. `https://github.com/steven31415/circular-string-filter`.
3. T. Allers and M. Mevarech. Archaeal genetics – the third way. *Nat Rev Genet*, 6:58–73, 2005.
4. M. A. R. Azim, C. S. Iliopoulos, M. S. Rahman, and M. Samiruzzaman. A fast and lightweight filter-based algorithm for circular pattern matching. pages 621–622, 2014.
5. M. A. R. Azim, C. S. Iliopoulos, M. S. Rahman, and M. Samiruzzaman. A filter-based approach for approximate circular pattern matching. pages 24–35, 2015.
6. R. Dulbecco and M. Vogt. Evidence for a ring structure of polyoma virus DNA. *Proc Natl Acad Sci*, 50(2):236–243, 1963.
7. F. Fernandes, L. Pereira, and A. Freitas. CSA: An efficient algorithm to improve circular DNA multiple alignment. *BMC Bioinformatics*, 10:1–13, 2009.
8. D. Gusfield. *Algorithms on Strings, Trees and Sequences.* New York, NY, USA: Cambridge University Press, 1997.
9. T. Lee, J. Na, H. Park, K. Park, and J. Sim. Finding optimal alignment and consensus of circular strings. *Proceedings of the 21st annual Conference on Combinatorial Pattern Matching*, pages 310–322, 2010.
10. G. Lipps. *Plasmids: Current Research and Future Trends.* Norfolk, UK: Caister Academic Press, 2008.
11. A. Mosig, I. Hofacker, P. Stadler, and A. Zell. Comparative analysis of cyclic sequences: viroids and other small circular RNAs. *German Conference on Bioinformatics, Volume 83 of LNI*, pages 93–102, 2006.
12. M. Thanbichler, S. Wang, and L. Shapiro. The bacterial nucleoid: A highly organized and dynamic structure. *J Cell Biochem*, 96(3):506–521, 2005. [http://dx.doi.org/10.1002/jcb.20519].
13. R. Weil and J. Vinograd. The cyclic helix and cyclic coil forms of polyoma viral DNA. *Proc Natl Acad Sci*, 50(4):730–738, 1963.