dx.doi.org/10.24254/CNIB.14.31

An improved consensus algorithm for approximate string matching

Miguel E. Rubio-Rincon, Alfonso Alba, Edgar R. Arce-Santana, and Martin O. Mendez-Garcia.

***Abstract*— One of the fundamental tasks in bioinformatics consists in searching for patterns, in a protein or DNA sequence, that are sufficiently similar to a given motif. This problem is known as approximate string matching (ASM) and has several applications besides bioinformatics. The similarity between strings of symbols is typically evaluated by metrics such as the Hamming distance, the Levensthein distance, or correlation or consensus techniques. In this paper, a refinement of a recently introduced consensus algorithm is proposed and evaluated with real protein sequences from plants. Preliminary tests with real protein sequences from plants show that the proposed refinement can significantly increase the localization accuracy by up to 95%, while further reducing the number of false positives by around 80%. Thus, the proposed algorithm could be a useful tool in many biological applications.**

I. Introduction

Approximate string matching (ASM) is one of the fundamental tasks in bioinformatics. Given a finite alphabet A, a pattern string $a = a_1 a_2 \dots a_m \in A^*$ of length m and a search string $b = b_1 b_2 \dots b_n \in A^*$ of length n (also called the *search text*), where * represents the Kleene star operation, the problem consists in finding all the substrings in b which are sufficiently similar to a. This requires defining a similarity measure between strings. The most popular measures are the Hamming distance [1], which is equal to the number of mismatching symbols between two strings of equal length, or infinite if the strings have different lengths, and the Levensthein distance [2] (also called edit distance), which is equal to the number of edit operations (insertions, deletions, and substitutions of symbols) that must be performed in order to transform one of the strings into the other, where the strings may have different lengths. Other similarity measures are based on correlation methods [3], although these may not have a direct interpretation in terms of editing operations such as symbol insertions or substitutions. Given a similarity measure $d(x, y)$, where x and y are two strings, and a similarity threshold k, the ASM problem can be formally defined as follows: find all the indices r such that $d(a_1 a_2 \dots a_m,\ b_r b_{r+1} \dots b_{r+s-1}) \leq k$ for some non-negative integer s that depends on r. Ideally, each value of r should correspond to the first symbol of an approximate instance of the pattern string. However, solving this problem often results in a large number of false positives, often due to the fact that multiple consecutive indices may correspond to the same instance of the pattern string. For example, under the Levensthein measure, if a match is detected at position r in the search string with an edit distance $d < k$, then a match will also be reported at positions $r - 1$ and $r + 1$ (at least) with edit distance $d + 1 \leq k$. Depending on the size of the alphabet, the length of the pattern string, and the similarity threshold, a typical search within a large DNA database may return between tens of thousands to several millions of false positives [4]. For this reason, a filtering stage is often performed to discard any matches that either correspond to an already reported instance, or are irrelevant for the application.

Recently, a new ASM algorithm was proposed, which uses a consensus measure to assess the degree of similarity between strings, and also applies a post-filtering stage to significantly reduce the number of false positives with little computational cost [5,6]. The algorithm, however, has an important shortcoming: in practice, a reported index r is often several positions (up to k) apart from the true beginning of an instance.

In this paper, an improved version of the consensus algorithm is presented, where a second filtering stage is performed, based on the dynamic programming technique that is used to compute the Levensthein distance. The proposed method significantly improves the localization of the true beginning of each instance, while further reducing the number of false positives.

The article is organized as follows: in Section II, the basic consensus algorithm, as introduced in [6], is briefly described, along with the original post-filtering stage. Section III will describe the proposed refinements. In Section IV, some preliminary results using real protein sequences will be presented and discussed. Finally, Section V will summarize our conclusions.

II. Basic consensus algorithm

A. Baeza-Yates and Perleberg algorithm

The consensus method proposed in [6] is based on a modification of an algorithm introduced by Baeza-Yates and Perleberg to efficiently estimate the Hamming distance [7]. In this first stage of this algorithm, one computes, for each symbol $s \in A$, the set $\alpha(s)$ of zero-based positions where s appears in the pattern string; that is, $\alpha(s) = \{j : a_{j+1} = s, 0 \leq j \leq m - 1\}$. Note that $\alpha(s)$ is empty if s does not belong in the pattern string. In the second stage of the algorithm, the search string is traversed, and for each symbol $b_j,\ j = 1, \dots, n$ one increases a counter c_{j-q} for all $q \in \alpha(b_j)$. The rationale behind this algorithm is that if a symbol b_j in the search string appears in the pattern string, then an instance of the pattern string might begin at positions $j - q, q \in \alpha(b_j)$ in the search string. After the search string is traversed, the counter $c_j,\ j = 1, \dots, n$ indicate the number of matching

Research supported by CONACyT grant CB-2010-154623.

All authors are with the Facultad de Ciencias, Universidad Autonoma de San Luis Potosi.

Corresponding author: A. Alba, Facultad de Ciencias, UASLP, Av. Salvador Nava Mtz. S/N, Zona Universitaria, San Luis Potosi, SLP, 78290, Mexico. Phone: +52 (444) 8262486 x 2906. E-mail: fac@fc.uaslp.mx

symbols between the pattern string and the substring $b_j \ldots b_{j+m-1}$. The Hamming distance between these strings can simply be computed as $m - c_j$.

B. Generalization to insertions and deletions of symbols

The Baeza-Yates and Perleberg algorithm, as described above, can only detect symbol mismatches, but is unable to take insertions and deletions of symbols into account. To overcome this problem, a simple strategy is introduced in [5] and [6]. Since any insertions or deletions might offset the length of an instance of the pattern string by up to k symbols (where k is the maximum edit distance allowed), the authors suggest to increase the counters c_{j-q+r} for each $q \in \alpha(b_j)$ and for all $r = -k, \ldots, k$. We have found that a better strategy is to simply replace the alpha-sets by an expanded set $\hat{\alpha}(s)$, which can be obtained as

$$\hat{\alpha}(s) = \{q + r : q \in \alpha(s), r = -k, \ldots, k\}.$$

The expanded set $\hat{\alpha}(s)$ can then be used instead of the set $\alpha(s)$ in the Baeza-Yates and Perleberg algorithm described in the previous subsection.

Note that, with this modification, the counters c_j are no longer related to the Hamming distance nor they are related to the Levensthein distance. Instead, these counters conform a voting system (i.e., a consensus) related to the likelihood that an approximate instance of the pattern string begins at a given position in the search string. If the $\hat{\alpha}(s)$ sets are used, the counters take values between 0 and m. If a counter c_j exceeds a given threshold U (not necessarily related to k), the algorithm reports position j as a positive.

C. Reduction of false positives

As in many other ASM algorithms, the consensus algorithm described in the previous sections may report a large number of false positives. However, most of these are adjacent positions, corresponding to the same instances of the pattern string within the search string. In order to eliminate these false positives, one can perform a post-processing stage where for each sequence of consecutive positions reported as positives $j, \ldots, j + r$, such that $c_j, \ldots, c_{j+r} \geq U$, one can choose the first position p, $j \leq p \leq j + r$ such that $c_p \geq c_j, \ldots, c_{j+r}$, and report p as the only positive of the sequence. This filtering stage may reduce the number of false positives by 80% to 90% with very little computational cost.

III. Proposed Refinations

One disadvantage of the consensus algorithm described in the previous section is that the reported positives may have a localization error of up to k positions with respect to the true beginning of the pattern instances. This is because the maximum counter value in a group of consecutive positives does not always correspond to the start of an instance. To overcome this problem, one can compute the Levensthein distance between the pattern string and the search substring $b_{p-k} \ldots b_{p+m-1+k}$, for each positive p reported by the consensus method. One way to do this is by means of the classic dynamic programming algorithm originally adapted by Sellers [8] for string matching. We specifically use the text-searching version reported in [9]. In this algorithm, one computes a matrix C in column-wise or row-wise order using the following recursive formula:

$$C_{i,j} = \min(C_{i-1,j-1} + \delta(a_i, b_{p-k-1+j}), C_{i-1,j} + 1, C_{i,j-1} + 1),$$

where $d(u, v) = 0$ if $u = v$ and 1 otherwise. It is also necessary to define $C_{i,0} = 1$ for all i and $C_{0,j} = 0$ for all j. Once this matrix has been computed, the bottom row contains the Levensthein distance between the pattern string and the text substring which ends at the corresponding column. For our application, we are interested in the beginning of the pattern instances, rather than the ending position; therefore, we apply the dynamic programming algorithm to the inverted input strings. Those values in the bottom row which result to be less or equal than k will correspond to matching positions; however, at this point we already know that the text substring corresponds to a single instance of the pattern, so one must only choose one of the positions with distance lower or equal than k. We have found that the floor of the average among the matching positions yields a much better localization than the original consensus method. Moreover, this refinement can further reduce the number of false positives in two ways: (1) when a false positive from the consensus method does not correspond to an actual match, the dynamic programming refinement will report no matches, and (2) when two or more false positives from the consensus method correspond to the same instance of the pattern, the proposed refinement will likely report the same matching position for all of them, effectively reducing multiple positives into a single one.

IV. Preliminary Results with Real Protein Sequences

To evaluate the proposed refinements, we have tested a C++ implementation of the proposed algorithm, as well as the original consensus method, with six different protein sequences from the following plants: *Opuntia streptacantha*, *Arabidopsis thaliana*, and *A. lyrata* [10,11,12]. Dehydrin proteins (DHN) are characterized by the presence of one or more K-segments consisting of approximately 15 amino acid residues [EKKGIM(E/D)KIKEKLPG], which form a putative amphiphatic α-helix. DHN proteins may also contain S-segments [LHRSGS$_{4\text{-}10}$(E/D)$_3$] formed by a stretch of 4-10 serine residues, and Y-segments [(V/T)D(E/Q)YGNP] that are located near the N-terminus [12,13]. DHN proteins, in general, share low sequence identities, so proper identification of conserved motifs is important for the classification of these proteins into the different DHN sub-types.

Each of the six protein sequences contains multiple approximate instances of at least one of the patterns (K-segment, S-segment or Y-segment), whose true locations (i.e., the position of the first symbol of each instance) were manually annotated by an expert biologist, and thus are known in advance. Among the six sequences, there are 4 S-segments, 17 K-segments, and 8 Y-segments, for a total of 29 instances. In this preliminary evaluation, we apply both versions of the consensus algorithm (with and without the refinement proposed in Section III) and determine the following indices:

- True positives (TP), the number of reported matches which correspond to a true instance

- False positives (FP), the number of reported matches which do not correspond to a true instance
- False negatives (FN), the number of true instances that do not correspond to any of the reported matches
- Localization error (LE), the average absolute difference between a reported true positive, and the position of the corresponding true instance.
- Computation time (CT), the total search time for all the six protein sequences and all three patterns.

In all cases, the edit distance threshold k was chosen to be around $m/3$, where m is the length of the pattern string; specifically, $k = 2$ for the S-segment, $k = 3$ for the Y-segment, and $k = 6$ for the K-segment. Also, following [7], the consensus threshold U was chosen to be around $U = (2/3)m$. Slight adjustments were made in order to achieve a 100% true positive rate, thus increasing the number of false positives.

The resulting indices, for each version of the algorithm, are reported in Table I. Moreover, Figure 1 shows the six protein sequences and the positions that the proposed algorithm (including the refinement stage) reported as positives. The localization error (LE) is significantly lower for the refined version of the consensus algorithm with respect to the basic version. This is because the refined algorithm is able to find the exact location on 22 of the 29 pattern instances, and finds 7 more instances with a localization error of only 1. On the other hand, the basic algorithm yields a localization error of up to 6 for some instances (of the K-segment, where $k = 6$ is used). Thus, in this experiment the proposed refinement reduces the localization error by 80% to 95%. Note also that the total number of false positives is 131 for the basic consensus method, while using the proposed refinement reports only 5 false positives; that is, a reduction of 95% of false positives.

Finally, we also measured the total computation time, for the search of the three patterns within the six protein sequences. Computation times with an Intel Core2Duo CPU running at 2.4 GHz were 500 μs for the basic algorithm, and 800 μs for the refined version; therefore, in this test the proposed refinement increases computation time by around 60%; however, this figure will depend on the number of positives reported by the basic consensus algorithm.

V. Conclusions

A refinement for the consensus method for approximate string matching, originally introduced in [5,6], was presented. This refinement, which uses the classic dynamic programming algorithm for computing the edit distance between two strings, significantly reduces the localization error and also reduces the number of false positives with an acceptable computational cost. Further research will focus on a more detailed evaluation, and a comparison with other state-of-the-art methods.

References

The authors would like to thank Margarita Rodriguez-Kessler for providing the protein sequences used in this study and for the fruitful discussions.

References

[1] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, pp. 147-160, 1950.
[2] V. I. Levensthein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707-710, 1966.
[3] A. Alba, M. Rodriguez-Kessler, E. R. Arce-Santana and M. O. Mendez, "Approximate string matching using phase correlation," in *Proc. 34th Annual Int. Conf. of the IEEE EMBS*, pp. 6309-6312, 2012.
[4] J. Buhler, "Efficient large-scale sequence comparison by locality sensitive hashing," *Bioinformatics*, vol. 17, pp. 21-27, 2001.
[5] M. Rubio, A. Alba, M. Mendez, E. Arce-Santana, M. Rodriguez-Kessler, "A Consensus Algorithm for Approximate String Matching," *Procedia Technology*, vol. 7, pp. 322-327, 2013.
[6] A. Alba, M. Rubio-Rincon, M. Rodriguez-Kessler, E. R. Arce-Santana, M. O. Mendez, "Un algoritmo de consenso para la búsqueda aproximada de patrones en cadenas de proteínas," *Revista Mexicana de Ingeniería Biomédica*, vol. 33, pp. 87-99, 2012.
[7] R. A. Baeza-Yates, C. H. Perleberg, "Fast and practical approximate string matching," *Inf. Process. Lett.*, vol. 59, pp. 21-27, 1996.
[8] P. H. Sellers, "The theory and computation of evolutionary distances: pattern recognition," *Journal of algorithms*, vol. 1, pp. 359-373, 1980.
[9] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, pp-31-88, 2001.
[10] A. Ochoa-Alfaro, M. Rodriguez-Kessler, M. Perez-Morales, P. Delgado-Sanchez, C. Cuevas-Velazquez, G. Gomez-Anduro, J. Jimenez-Bremont, "Functional characterization of an acidic SK3 dehydrin isolated from an Opuntia streptacantha cDNA library," *Planta*, vol. 235, pp. 565-578, 2012.
[11] M. Hundertmark, D. K. Hincha, "LEA (late embryogenesis abundant) proteins and their encoding genes in Arabidopsis thaliana," *BMC Genomics*, vol. 9, pp. 118, 2008.
[12] J. F. Jimenez-Bremont, I. Maruri-Lopez, A. Ochoa-Alfaro, P. Delgado-Sanchez, J. Bravo, M. Rodriguez-Kessler, "LEA gene introns: is the intron of dehydrin genes a characteristic of the serine-segment?," *Plant Mol Biol Rep*., vol. 31, pp., 128-140, 2013.
[13] C. R. Allagulova, F. R. Gimalov, F. M. Shakirova, V. A. Vakhitov, "The plant dehydrins: structure and putative functions," *Biochemistry (Moscow)*, vol. 68, pp. 945-951, 2003.

TABLE I. Matching results for six protein sequences using the basic consensus algorithm and the proposed refinement

	S-segments		K-segments		Y-segments	
	LHRSGS		EKKGIMDKIKEKLPG		DEYGNP	
	$k = 2, U = 4$		$k = 6, U = 11$		$k = 3, U = 4$	
	Basic	Refined	Basic	Refined	Basic	Refined
TP	4	4	17	17	8	8
FP	30	**0**	29	**2**	72	**3**
FN	0	0	0	0	0	0
LE	2.25	**0.50**	4.05	**0.17**	3.00	**0.25**

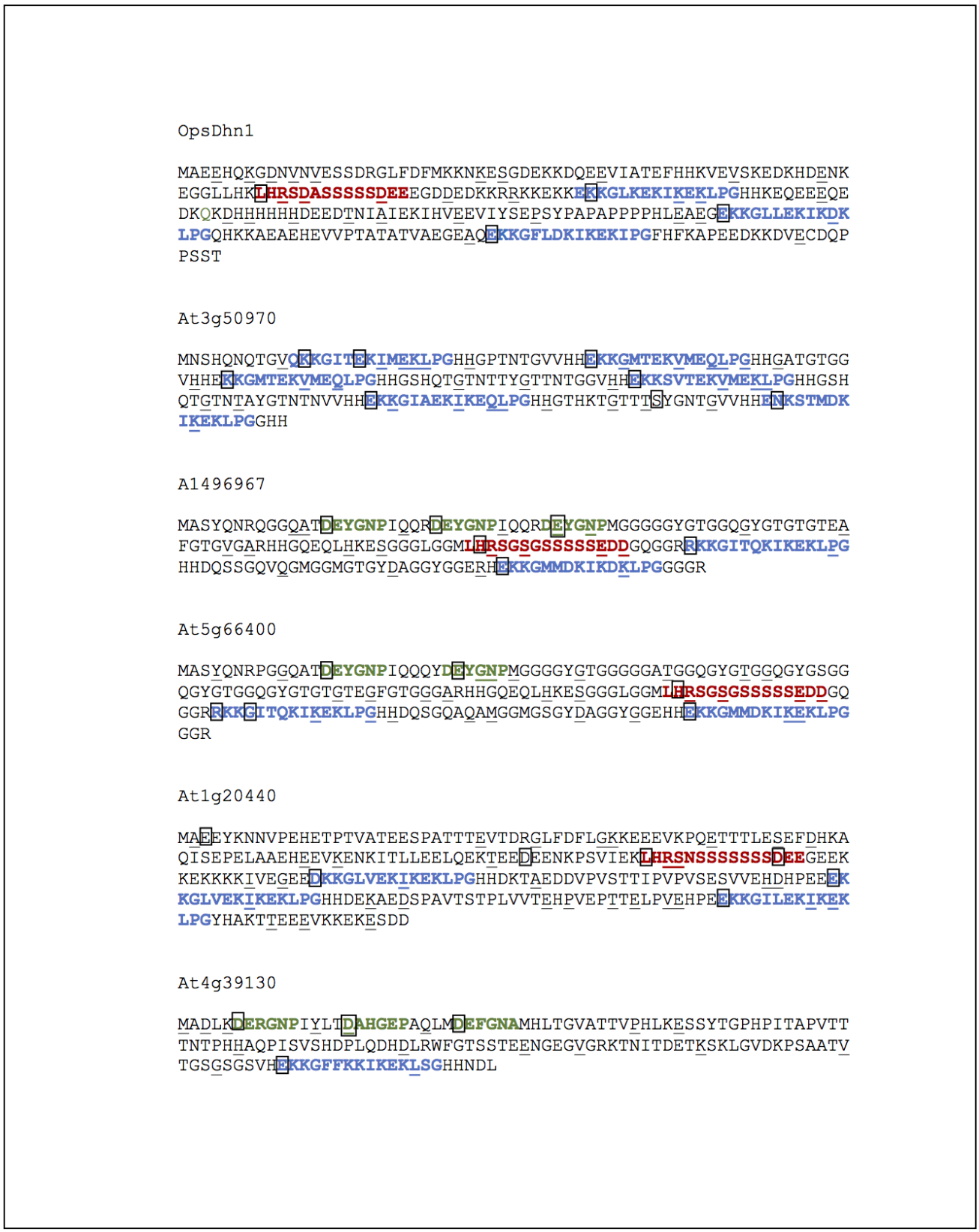

Figure 1. Results obtained for the six protein sequences. True instances of the pattern strings are represented using different colorings: S-segments are shown in red, K-segments in blue, and Y-segments in green. The reported matches (only the first position) are shown as underlines for the basic consensus method, and rectangular boxes for the refined algorithm proposed here.