Empirical Transition Probability Indexing Sparse-Coding Belief Propagation (ETPI-SCoBeP) Genome Sequence Alignment

Aminmohammad Roozgard*, Nafise Barzigar*, Shuang Wang°, Xiaoqian Jiang° and Samuel Cheng*

Abstract

The advance in human genome sequencing technology has significantly reduced the cost of data generation and overwhelms the computing capability of sequence analysis. Efficiency, efficacy and scalability remain challenging in sequence alignment, which is an important and foundational operation for genome data analysis. In this paper, we propose a two stage approach to tackle this problem. In the preprocessing step, we match blocks of reference and target sequences based on the similarities between their empirical transition probability distributions using belief propagation. We then conduct a refined match using our recently published SCoBeP technique. Our experimental results demonstrated robustness in nucleotide sequence alignment and our results are competitive to those of the SOAP aligner and the BWA algorithm. Moreover, compared to SCoBeP alignment, the proposed technique can handle sequences of much longer lengths.

I. Introduction

In bioinformatics, sequence alignment is an important way to identify similar regions that might be associated with similar functional and structural relationship between sequences. With the quick growth of genomic data, it is important to develop effective sequence alignment

*School of Electrical and Computer Engineering, University of Oklahoma, Tulsa, OK, USA.

°Division of Biomedical Informatics, University of California, San Diego, CA, USA.

Shuang Wang and Xiaoqian Jiang were funded in part by the NIH grants R00LM011392, U54HL108460 and the NHGRI (K99HG008175)

techniques that are scalable. The past decade has witnessed the development of many sequence alignment technologies. Cancers are caused by the collection of genomic sequence changes [1]. Therefore, alignment and analyses of cancer genome sequences provide basics to understand cancer biology, diagnosis and therapy.

In general, pairwise sequence alignment methods can be classified into local and global approaches. The global alignment attempts to find the best match between two strings with similar lengths through global optimization. In contrast, the local alignment is usually used to identify regions of similarity between a short query and a longer sequence. Global alignments [2]–[5] are less prone to demonstrating false homology as each letter of one sequence is constrained to being aligned to only one letter of the other. Local alignments [6]–[9], on the other hand, can cope with rearrangements between non-syntenic, orthologous sequences by identifying similar regions in sequences; this, however, comes at the expense of a higher false positive rate due to the inability of local aligners to take into account overall conservation maps [10].

A lot of efforts have been made to improve the efficiency and efficacy of sequence alignments. The ClustalW program proposed by Thompson and Larkin [11], [12] uses a multi-stage mechanism to weight and to align sub-sequences based on sequence divergences. In addition, sequence annealing technique incrementally builds sequence alignment one at a time by checking whether a single match is consistent with a partial multiple alignments [13]. Darling *et al.* proposed a hidden Markov model that uses a sum-of-pairs breakpoint score to facilitate the detection of rearrangement breakpoints, when genomes have unequal gene content [14]. Mummer is a highly efficient suffix tree based matching tool for whole genome alignment, as well as incomplete genomes [15].

Researchers also proposed heuristics to accelerate sequence alignment. For example, the bounded sparse dynamic programming (BSDP) is used to support rapid approximation of exhaustive alignment in [16]. Another heuristic-driven approach, namely FastTree, is a tree-based method that stores profiles of internal nodes in a tree, such that candidate joins can be quickly identified. FastTree is also scalable for handling alignments over 10,000 sequences [17], [18].

Maximum-likelihood based approaches like PhyML and RAxML-VI-HPC have been developed as well. PhyML [19] used a hill-climbing algorithm that adjusts tree topology and branch length at each tree modification iteration. RAxML-VI-HPC [20], which stands for randomized accelerated maximum likelihood for high performance computing, takes advantages of a parallel program to support large-scale genome alignment.

In this paper, we propose a novel alignment method that uses sparse coding [21] and empirical transition probability to tackle the scalability challenge. Thanks to the sparse representation, our mechanism can handle long sequences with reduced memory footprint. We also leverage belief propagation to combine local and neighboring information of candidate nucleotides into consideration and generate matching scores to determine the best match. The rest of this paper is structured as follows. Section II introduces our proposed method. Section III presents our results, including the comparison against SOAP aligner [22] and BWA [23]. Finally, we draw our conclusions in Section IV.

II. Proposed Method

In this section, we present our genome indexing and alignment framework in detail, where the proposed method includes three steps: indexing, index matching, and sequence matching. In this paper, we refer to "*reference sequence*" as the base-line sequence and try to align a "*read sequence*" against the base-line sequence.

A. Indexing

The current genome indexing methods generate huge indices before performing the actual alignment to decrease the alignment time [24], [25]. The indexing process can be very time-consuming. In contrast, our proposed indexing technique provides a faster and light-weight alternative for index generation, which is similar to the big data retrieval systems that were proposed [26]–[28]. These indices can reduce the search space and provide an estimation of the read sequence locations in the reference sequence. The proposed genome indexing technique models a nucleotide sequence as a graph by counting the transitions between each pair of nucleotides. To be more specific, as shown in Fig. 1, we consider a graph with four states according to the different types of nucleotides and sixteen vertices according to all possible transitions between nucleotides. We read the first nucleotide of the sequence and treat it as the initial state. Then, we move from one state to the other state by scanning the next nucleotide repeatedly till the end of the sequence. Afterwards, we calculate the number of nucleotide transitions where we count how many times we pass one vertex in the graph and store them in a 4×4 matrix. Finally, we normalize the resulting matrix as follows:

$$I = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \!\!\! \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} k_{aa} & k_{ac} & k_{ag} & k_{at} \\ k_{ca} & k_{cc} & k_{cg} & k_{ct} \\ k_{ga} & k_{gc} & k_{gg} & k_{gt} \\ k_{ta} & k_{tc} & k_{tg} & k_{tt} \end{pmatrix} \end{array} \times \frac{1}{\sum_{s,w \in \{a,c,g,t\}} k_{sw}} \qquad (1)$$

where k_{sw} is the number that has the S-type nucleotide immediately before the W-type nucleotide.

If the length of a sequence is larger than a given threshold i.e., h, we divide it into subsequences with maximum length of h, where each subsequence will have o nucleotides overlap with their neighbors. We set $o \geq \frac{h}{2}$ so that each pair of nucleotides can be counted at least twice. For each subsequence i, we count the transition of the nucleotides from the start of the subsequence till its end to reveal the number of different nucleotides that reside beside each other. In Fig. 2, an input sequence with $h = 250$ is used to demonstrate the proposed indexing process, where I_i is the calculated index for the input sequence based on the transition graph shown on the left hand side. Finally, we normalize the transition matrix, which will be used to find the approximate location of each subsequence in the next step.

B. Index matching

The index matching step is designed to find similar indices based on global information of the sequence. We define a symmetric distance function between two index matrices I and J as follows: $D_{MSE}(I, J) = \|I - J\|_f$, where $\|\cdot\|_f$ is the Frobenius norm of the matrix.

After generating the indices of the reference sequence and the read sequence, the D_{MSE} distances to all reference sequence indices are calculated, where the top t most similar indices in terms of D_{MSE} are chosen as candidate indices. To find the best matched index, we resort to belief propagation (BP) on a factor graph. In this paper, we provide a concise review about the BP algorithm on factor graph for the proposed algorithm. Interested readers can check our earlier publications in [29]–[31] for more details about the factor graph design and the BP algorithm.

We apply BP to the factor graph of the test sequence with n candidate nucleotides as the prior knowledge. BP updates the probability of candidate nucleotides based on the probabilities of their neighbors.

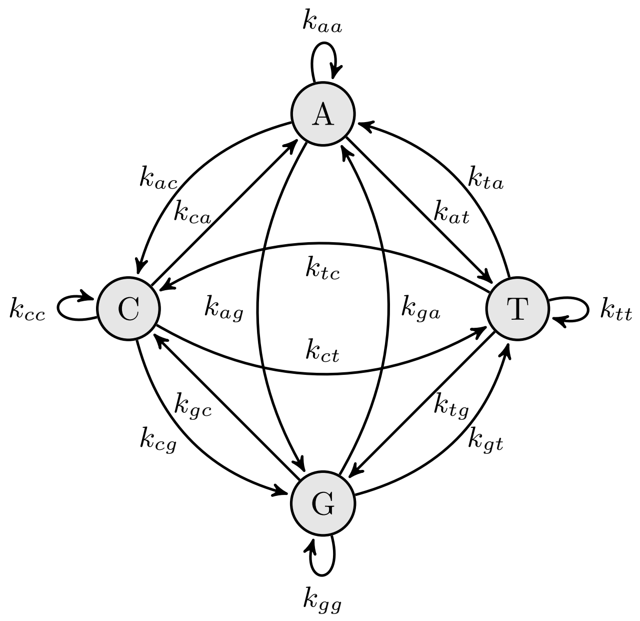
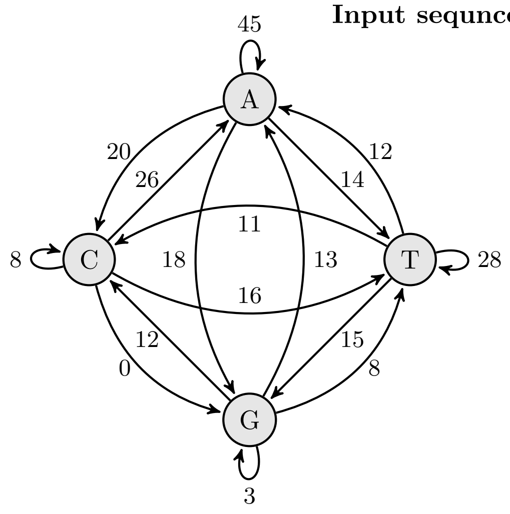


Fig. 1: The transition diagram between nucleotides. k_{sw} is the number of appearance of the W-type nucleotide immediately after the S-type nucleotide where $s, w \in \{a, c, g, t\}$.

Then, the candidate index numbers are fed to a factor graph and the corresponding D_{MSE} of each of candidates is employed to calculate the initial probability (prior probability) of each candidate. Then, message passing (i.e., forward and backward) algorithm is applied to calculate the best match indices. The correspond subsequences of these indices is used in the next step.

C. Sequence matching

The sequence matching step is based on sparse coding and BP algorithm. In this step, we use the subsequences that were selected in the previous step to generate an over-complete dictionary. Then, for each nucleotide in the read sequence, we pick n candidate nucleotides using sparse coding. By applying belief propagation to a factor graph, we can obtain the best match for each nucleotide in the read sequence. A detailed description about the sequence matching can be found in our recent publications [31], [32]. A summary of the main procedure for our proposed alignment method is shown in Algorithm 1.



Input sequnce={aaaccaagaggcagaggttgcagtgagccaagatcatgccattgcactccagcc
ttagcaacagagtgagactccatctcagaacaacaacaacaacaaaaaaaaaaa
acacaaaaaaacaaaaaattctgcaactaattaatttgttgtaactcttaaagcag
gaaccttatatagaaaatgttgtgatcctattaatttttttttctttctatgtaagca
acttcacttttgactttgcagcactgac}

$$I_i = \begin{pmatrix} 45 & 20 & 18 & 14 \\ 26 & 8 & 0 & 16 \\ 13 & 12 & 3 & 8 \\ 12 & 11 & 15 & 28 \end{pmatrix} \times \frac{1}{249}$$

$$= \begin{pmatrix} 0.1807 & 0.0803 & 0.0723 & 0.0562 \\ 0.1044 & 0.0321 & 0.0 & 0.0643 \\ 0.0522 & 0.0482 & 0.0120 & 0.0321 \\ 0.0482 & 0.0442 & 0.0602 & 0.1124 \end{pmatrix}$$

Fig. 2: An example of the indexing procedure for a small sample subsequence.

D. Implementation details:

- $I_i = MakeIndex(x_i)$ fills the state matrix I_i using the relationship of nucleotides in the subsequence x_i. The subsequence x_i is scanned through all its nucleotides and the corresponding counts will be stored into the state matrix I_i. For example, k_{cg} in I_i in (2) shows how many times the nucleotide C will be identified, which is next to the nucleotide G in the subsequence x_i. Note that each subsequence x_i has a separate state matrix I_i, where i is the subsequence index.

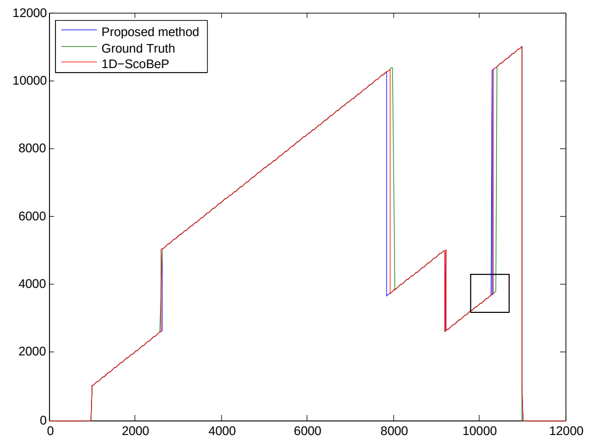
$$I_i = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} k_{aa} & k_{ac} & k_{ag} & k_{at} \\ k_{ca} & k_{cc} & k_{cg} & k_{ct} \\ k_{ga} & k_{gc} & k_{gg} & k_{gt} \\ k_{ta} & k_{tc} & k_{tg} & k_{tt} \end{pmatrix} \end{array} \tag{2}$$

- $[c_j, \rho_j] = FindCandidates(J_j, I, k)$ identifies k candidate state matrices that are highly similar to the test state matrix J_j in I and stores their indices in vector c_j and their probabilities in vector ρ_j. Note that the approach will compute the Mean Square Error (MSE) of the test state matrix J_j with each possible I_i of the reference state matrices and select I_i that has the smallest MSEs.

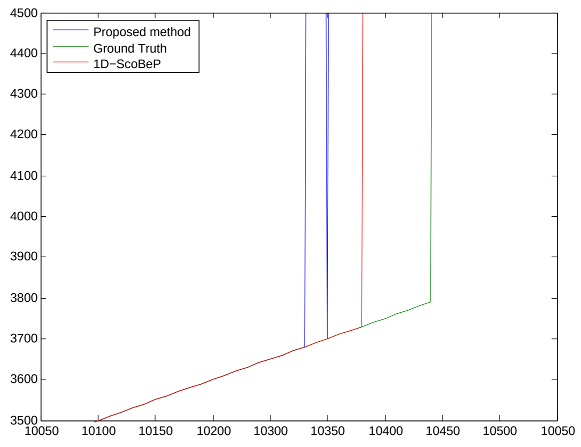
Algorithm 1 Proposed nucleotide sequence alignment algorithm for estimating the location of the input sequence

```
Inputs: a reference sequence $\mathcal{X} \in \mathbb{R}^M$, a test sequence $\mathcal{Y} \in \mathbb{R}^N$, number of the candidate state
    matrix $k$, number of the candidate points $n$
Initialize: a $4 \times 4$ state matrix $I$ storing the numbers of nucleotide states (2), nucleotide overlap
    $v$
Fill the reference state matrix $I$: For each subsequence $x_i \in \mathcal{X}$ with $v$ nucleotide overlap in
    each direction perform:
    • $I_i = MakeIndex(x_i)$
Fill the test state matrix $J$: For each subsequence $y_j \in \mathcal{Y}$ with $v$ nucleotide overlap in each
    direction perform:
    • $J_j = MakeIndex(y_j)$
    • $[c_j, \rho_j] = FindCandidates(J_j, I, k)$
Refine the candidate state matrix:
    • $\hat{\rho} = BP(c, \rho)$
Find the correspond nucleotide in the reference sequence $\mathcal{X}$: For each subsequence $y_j \in \mathcal{Y}$
    with $v$ nucleotide overlap in each direction perform:
    • $z_j = FindBestSubsequence(\mathcal{X}, y_j, \theta, n)$ (see [31], [32] for more details)
Output: the estimated version of aligned sequence $\mathcal{Z}$
```

- $\hat{\rho} = BP(c, \rho)$ models the problem by a factor graph and applies belief propagation [33] to update probability ρ. The updated probability $\hat{\rho}$ can be used to align the reference state matrix index onto the test state matrix index. In our case, we assign a variable node for each test state matrix index and connect each pair of neighboring state matrix indices with a factor node. Also, we introduce one extra factor node to take care of the prior knowledge obtained in the MSE step for each test state matrix index (for more details, see [32]).
- $z_j = FindBestSubsequence(\mathcal{X}, y_j, \theta, n)$ finds the corresponding location for a nucleotide $y_j \in \mathcal{Y}$. In this step, the reference nucleotide sequence $\mathcal{X}$ and the test nucleotide sequence $\mathcal{Y}$ are converted into two integer sequences. Then, an over-complete dictionary is built with all



(a)



(b)

Fig. 3: The results of proposed method for non-collinear nucleotide sequence alignment. a) comparison among alignment results of the ground truth, 1D-SCoBeP [31] and the proposed method. b) zoomed of the black square in figure 3–a to show the gap between the proposed method, ground truth and 1D-SCoBeP [31] on the jump point. The x–axis and y–axis are the index numbers of the original genome sequences and the shuffled genome sequences, respectively.

subsequences in the $\mathcal{X}$. We then apply sparse coding followed by using Belief Propagation (BP) to identify the best matches. (see [31], [32] for more details) Note that we used non-overlapped subsequences to build the dictionary. This change decreases the memory usage and the accuracy of the proposed algorithm in compare to 1D-SCoBeP [31], but it increases the speed of our alignment algorithm.

III. EXPERIMENTAL RESULTS

We designed our experiments based on work in [14] to evaluate the proposed method for aligning the nucleotide sequences and to compare it with SOAP aligner [22], BWA [23] and 1D-SCoBeP [31]. We considered the problem of aligning a sequence of human nucleotides from the National Center for Biotechnology Information (NCBI) [34] and Cancer Genomics Hub (CGHub) [35].

To evaluate the performance of our approach, we conducted two sets of tests on the nucleotide sequences. In the first set, we selected fifty short sub-sequences of human genomes and then

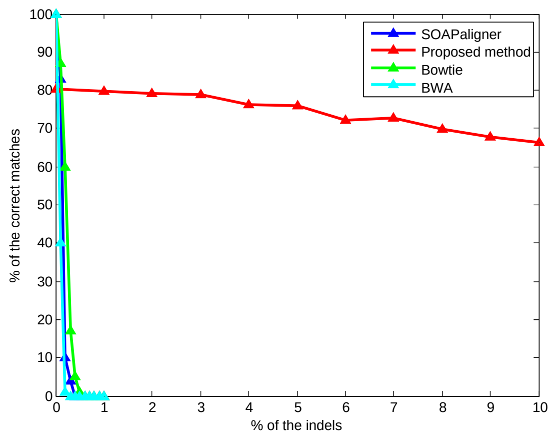


Fig. 4: Accuracy of BWA [23], SOAP aligner [22] and the proposed method in present of different Indel rate, where the testing genome sequences were obtained from [34].

used SOAP aligner, BWA, 1D-SCoBeP and the proposed method to find the location of selected sub-sequence nucleotide in the human chromosome. All of four algorithms successfully passed this test. We created twenty shuffled sub-sequences of the reference sequence as follows: for each read sequence R, we cut it into five pieces p_1, p_2, p_3, p_4 and p_5. Then we switched p_2 with p_4. Therefore, we converted a read sequence $R = [p_1, p_2, p_3, p_4, p_5]$ into a new read sequence $\hat{R} = [p_1, p_4, p_3, p_2, p_5]$.

Fig. 3 shows the result of the 1D-SCoBeP and the proposed method show a better performance with a gap of 100 to 120 nucleotides away from the ground truth. Since we were using non-overlapped subsequences for the dictionary generation, the gap between the proposed method and the ground truth was larger than these reported in 1D-SCoBeP [31]. In our experiments, the following parameters were used: the number of candidate points n is set to be 3, the sparsity factor $k = 3$ and the dictionary column size $a = 200$.

To evaluate the robustness of the proposed method, we generate indices for long human genome sequences (i.e. 5×10^8 nucleotides) where $h = 10000$ and $o = 5000$. Moreover, we synthesized insertion, deletion and mutation (i.e., indel) in these sequences. For indel rate, we picked 10^5 number of subsequences with size of 10^4 nucleotides. Then, we randomly modified a

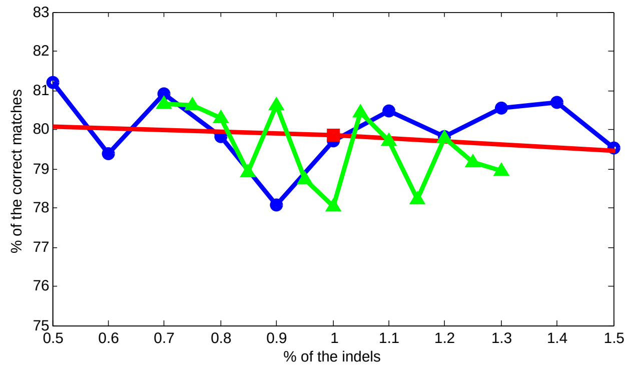


Fig. 5: The percentage of successful alignments in present of 0.5% to 1.5% indels. The **green line** is the percentage successful alignments where the rate of the indels are changing with step equal 0.05% between 0.7% and 1.3%. The **blue line** is the percentage successful alignments where the rate of the indels are changing with step equal 0.1% between 0.5% and 1.5% and the **red line** is the same as the Fig. 4. Each point represents 10^5 random site selection with same indels rate. Note that the genome sequences used in this studies were obtained from [35] and [34].

certain number of nucleotides (based on the indel rate) and aligned them with the references. We counted how many times the alignment location and real subsequence location (i.e., ground truth) are matched, where the accuracy is defined as the count of the successfully aligned sequences over total number of the subsequences. Fig. 4 shows the accuracy of alignment of the proposed method, BWA and SOAP aligner in the presence of the different indel rates. The proposed method showed similar accuracies even when we increased the indel rate to 3%. Moreover, the proposed algorithm still showed more than 75% accuracy even after we modified 5% of the nucleotides in our selected subsequences. In contrast, the accuracy of the BWA and SOAP aligner decreased sharply as the indel rates increase.

We investigate the impact of small indel rate in the range from 0.5% to 1.5% in Fig. 5. In this figure, we showed accuracy of 1% indels in red for the data set used in 4 as reference. To verify our result, we repeat the experiments with different indel steps and different read locations and present the results in green and blue, respectively. Note that each point in this figure was obtained from the evaluation over 10^5 read sequences. There are slight variation among the curves due to statistical deviation. The summary of the indel rate accuracy was shown in Table I.

The computational complexity of proposed is mainly determined by the following three steps: 1) indexing 2) index matching 3) extracting sub-sequence nucleotides as features and constructing the dictionary, 4) finding candidate nucleotides via sparse coding, and 5) applying BP. Assume the size of the read and reference sequences are N and M nucleotides, respectively. The required time for create indexes is $O\big(M + N\big)$, because we have to scan whole read and reference sequences. The number of reference sequence indexes is $I_M = \frac{M}{h} + \frac{2oM}{h^2} = O\big(\frac{M}{h}\big)$ and similarly, $I_N = O\big(\frac{N}{h}\big)$. Therefore, the time for the index match matching is $O\big(\frac{M}{h}\big) \times O\big(\frac{N}{h}\big) = O\big(\frac{MN}{h^2}\big)$. After index matching step, the size of search space reduces from M to $\bar{M} = I_s \times h$ where I_s is the number selected indexes I_s and h is the size of each index. The required time of feature extraction will be $O\big(a(\bar{M}+N)\big)$, where a is the size of the vector of extracted features for each nucleotide. The dictionary construction step involves the normalization of each column, which requires $O(a\bar{M})$ amount of time. Thus the total time complexity of the first step is $O\big(a(\bar{M}+N)\big)$. In the next step, the time complexity of Subspace Pursuit (SP) is $O\big(\log(f)a\bar{M}\big)$ [36], where f is the number of iterations for searching the sparse vector. Since we have to repeat the process to find candidate points for all N feature vectors, the time complexity of finding candidate points by SP is $O\big(\log(f)a\bar{M}N\big)$. Then, the time complexity of Belief Propagation in our factor graph is $O(vn^2\bar{M})$, where v is the number iterations before converging and n is the number of candidates in each variable node. Finally, the time complexity of proposed method will be $O\Big(MN + \log(f)a\bar{M}N + vn^2\bar{M}\Big)$.

IV. Conclusion

In this paper, we proposed a sparse coding and BP based method for indexing and alignment genome sequences. The proposed method builds a transition matrix based on the neighboring nucleotides of an input sequence and then reduces the search space by selecting the top K most similar subsequences based on their distances. The proposed algorithm selects candidate nucleotides by using sparse coding with an over-completed dictionary, which was constructed from the nucleotides of reference sequence in the indexing step. BP algorithm is then applied to select the best matches. Through experimental results, we showed that the proposed algorithm are comparable to SOAP aligner [22], BWA [23] and 1D-SCoBeP [31] in terms of the alignment accuracy. In addition, the proposed method is robust to insertions, deletions, and mutations in the genome sequences when comparing with SOAP aligner and BWA. Finally, the proposed method is able to process much longer sequences then our previous 1D-SCoBeP approach.

REFERENCES

[1] M. Meyerson, S. Gabriel, and G. Getz, "Advances in understanding cancer genomes through second-generation sequencing," *Nature Reviews Genetics*, vol. 11, no. 10, pp. 685–696, 2010.

[2] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[3] B. Morgenstern, "Dialign 2: improvement of the segment-to-segment approach to multiple sequence alignment." *Bioinformatics*, vol. 15, no. 3, pp. 211–218, 1999.

[4] N. Bray, I. Dubchak, and L. Pachter, "Avid: A global alignment program," *Genome research*, vol. 13, no. 1, pp. 97–102, 2003.

[5] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov, E. D. Green, A. Sidow, S. Batzoglou, *et al.*, "Lagan and multi-lagan: efficient tools for large-scale multiple alignment of genomic dna," *Genome research*, vol. 13, no. 4, pp. 721–731, 2003.

[6] T. F. Smith and M. S. Waterman, "Comparison of biosequences," *Advances in Applied Mathematics*, vol. 2, no. 4, pp. 482–489, 1981.

[7] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[8] M. Brudno and B. Morgenstern, "Fast and sensitive alignment of large genomic sequences," in *Bioinformatics Conference, 2002. Proceedings. IEEE Computer Society*. IEEE, 2002, pp. 138–147.

[9] S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler, and W. Miller, "Human–mouse alignments with blastz," *Genome research*, vol. 13, no. 1, pp. 103–107, 2003.

[10] M. Brudno, S. Malde, A. Poliakov, C. B. Do, O. Couronne, I. Dubchak, and S. Batzoglou, "Glocal alignment: finding rearrangements during alignment," *Bioinformatics*, vol. 19, no. suppl 1, pp. i54–i62, 2003.

[11] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic acids research*, vol. 22, no. 22, pp. 4673–4680, 1994.

[12] M. Larkin, G. Blackshields, N. Brown, R. Chenna, P. McGettigan, H. McWilliam, F. Valentin, I. Wallace, A. Wilm, R. Lopez, *et al.*, "Clustal w and clustal x version 2.0," *Bioinformatics*, vol. 23, no. 21, pp. 2947–2948, 2007.

[13] A. S. Schwartz and L. Pachter, "Multiple alignment by sequence annealing," *Bioinformatics*, vol. 23, no. 2, pp. e24–e29, 2007.

[14] A. E. Darling, B. Mau, and N. T. Perna, "progressivemauve: multiple genome alignment with gene gain, loss and rearrangement," *PloS one*, vol. 5, no. 6, p. e11147, 2010.

[15] S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, M. Shumway, C. Antonescu, S. L. Salzberg, *et al.*, "Versatile and open software for comparing large genomes," *Genome Biol*, vol. 5, no. 2, p. R12, 2004.

[16] G. S. Slater and E. Birney, "Automated generation of heuristics for biological sequence comparison," *BMC bioinformatics*, vol. 6, no. 1, p. 31, 2005.

[17] M. N. Price, P. S. Dehal, and A. P. Arkin, "Fasttree: computing large minimum evolution trees with profiles instead of a distance matrix," *Molecular biology and evolution*, vol. 26, no. 7, pp. 1641–1650, 2009.

[18] ——, "Fasttree 2–approximately maximum-likelihood trees for large alignments," *Plos one*, vol. 5, no. 3, p. e9490, 2010.

[19] S. Guindon and O. Gascuel, "A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood," *Systematic biology*, vol. 52, no. 5, pp. 696–704, 2003.

[20] A. Stamatakis, “Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models,” *Bioinformatics*, vol. 22, no. 21, pp. 2688–2690, 2006.

[21] B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: A strategy employed by v1?” *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[22] R. Li, C. Yu, Y. Li, T. Lam, S. Yiu, K. Kristiansen, and J. Wang, “Soap2: an improved ultrafast tool for short read alignment,” *Bioinformatics*, vol. 25, no. 15, pp. 1966–1967, 2009.

[23] H. Li and R. Durbin, “Fast and accurate short read alignment with burrows–wheeler transform,” *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[24] C.-M. Liu, T. Wong, E. Wu, R. Luo, S.-M. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, *et al.*, “Soap3: ultra-fast gpu-based parallel alignment tool for short reads,” *Bioinformatics*, vol. 28, no. 6, pp. 878–879, 2012.

[25] P. D. Vouzis and N. V. Sahinidis, “Gpu-blast: using graphics processors to accelerate protein sequence alignment,” *Bioinformatics*, vol. 27, no. 2, pp. 182–188, 2011.

[26] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik, “Blobworld: A system for region-based image indexing and retrieval,” in *Visual Information and Information Systems*. Springer, 1999, pp. 509–517.

[27] D. Doermann, “The indexing and retrieval of document images: A survey,” *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 287–298, 1998.

[28] C. Liu and H. Wechsler, “Robust coding schemes for indexing and retrieval from large face databases,” *Image Processing, IEEE Transactions on*, vol. 9, no. 1, pp. 132–137, 2000.

[29] A. Roozgard, N. Barzigar, S. Cheng, and P. Verma, “Dense image registration using sparse coding and belief propagation,” in *Signal Processing and Communication Systems (ICSPCS), 2011 5th International Conference on*, Dec 2011, pp. 1–5.

[30] ——, “Medical image registration using sparse coding and belief propagation,” in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, Aug 2012, pp. 1141–1144.

[31] A. Roozgard, N. Barzigar, S. Wang, X. Jiang, L. Ohno-Machado, and S. Cheng, “Nucleotide sequence alignment using sparse coding and belief propagation,” in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*. IEEE, 2013, pp. 588–591.

[32] N. Barzigar, A. Roozgard, S. Cheng, and P. Verma, “Scobep: Dense image registration using sparse coding and belief propagation,” *Journal of Visual Communication and Image Representation*, 2012.

[33] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 498–519, 2001.

[34] N. C. Institute, “The national center for biotechnology information,” http://www.ncbi.nlm.nih.gov/genome?db=genome, January 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/genome?db=genome

[35] S. C. University of California, “Cancer genomics hub,” https://cghub.ucsc.edu/datasets/benchmark-download.html, January 2013. [Online]. Available: https://cghub.ucsc.edu/datasets/benchmark-download.html

[36] W. Dai and O. Milenkovic, “Subspace pursuit for compressive sensing: Closing the gap between performance and complexity,” DTIC Document, Tech. Rep., 2008.

TABLE I: Percentage of successfull alignments

% of the Indels	Accuracy of **red** line	Accuracy of **blue** line	Accuracy of **green** line
0.00	80.33	–	–
0.50	–	81.19	–
0.60	–	79.38	–
0.70	–	80.90	80.64
0.75	–	–	80.63
0.80	–	79.82	80.29
0.85	–	–	78.90
0.90	–	78.09	80.63
0.95	–	–	78.74
1.00	79.85	79.71	78.04
1.05	–	–	80.43
1.10	–	80.49	79.70
1.15	–	–	78.22
1.20	–	79.81	79.78
1.25	–	–	79.16
1.30	–	80.54	78.94
1.40	–	80.70	–
1.50	–	79.52	–
2.00	79.09	–	–
3.00	78.90	–	–
4.00	76.33	–	–
5.00	75.90	–	–
6.00	72.09	–	–
7.00	72.86	–	–
8.00	69.79	–	–
9.00	67.87	–	–
10.00	66.42	–	–