HANDLING UPDATES OF A PAIRWISE SEQUENCE ALIGNMENT

Changjin Hong and Ahmed H. Tewfik

Department of Electrical and Computer Engineering, University of Minnesota
200 Union Street. SE, Minneapolis, MN 55455, USA
{hong0094, tewfik}@umn.edu

ABSTRACT

Sequence alignment or decoding in molecular biology is mostly done via computationally expensive dynamic programming(DP) based approaches. Unfortunately, as sequencing errors are discovered frequently, researchers must repeat all previous similarity analysis for the erroneous sequence. This can take hours or days. In this work, we derive relative tolerance bounds on node distances from a root node that guarantee that partial shortest path distances remain optimal. We then propose an algorithm that uses these bounds to skip all unperturbed parts of a sequence when recomputing an alignment. We also discuss techniques to reduce the memory requirements of the algorithm by focusing on the highly conserved segments of the sequence. Experimental results establish that our proposed alignment procedure can update alignment decisions of modified sequence with 4.6% to 18% of the number of computations required by the normal Needleman-Wunsch algorithm, depending on sequence length. Higher computational savings are achieved with longer sequences.

1. INTRODUCTION

Dynamic programming(DP) has been the fundamental tool for sequence alignment including both pairwise and profile alignment in molecular biology[1, 8]. Recently, it has become more important to detect remote homologies by querying a newly discovered sequence against a tremendously large number of sequences in public domain biological sequence databases[1]. However, it is clear that a large number of erroneous sequences appears in a data bank since nearly every time a listed gene is sequenced a second time, errors are reported. In general, a sequencing error is due to the accumulation of mistakes compounded through an experiment, including the misincorporation of bases in polymerase-chain-reaction amplification of templates, compression in sequencing ladders, misreading of autoradiographs, mistyping of results, and miscommunication of the sequences to the data base[9]. In today's collaborative research environment, any conclusion derived from this updated sequence should be reevaluated. The major challenge is that sequences are quite lengthy and so, repeating the similarity evaluation for all sequences that was done previously takes hours or days.

None of the previous algorithms, including the K-best score, pruning, stack decoding or the heuristic based approach[7] tries to use DP computational outcomes to speed up rematching. In this work, we recast the updated alignment problem in a weighted acyclic graph(WAG) setting. This allows us to analyze the effect of a change in an arc length[4] due to an error correction on a previously computed shortest path. Our proposed algorithm updates alignment decisions with no more than 4.6% to 18% of the calculations performed by the normal Needleman-Wunch(NW) algorithm on the perturbed sequence for a pair of sequences having greater than 20% similarity.

In section 2, we review the prior work on a single arc tolerance analysis and we observe the difficulty of reusing the DP results from a previous alignment. The proposed algorithm is described in section 3. In section 4, our approach is verified with experimental results. We conclude the discussion with potential future work in section 5.

2. THE DIFFICULTY IN MAKING DYNAMIC PROGRAMMING REUSABLE

2.1 Single Arc Tolerance

In the network routing problem under the dynamic behavior, a sensitivity analysis on a single arc in WAG has been explored to determine by how much each arc length can be individually perturbed without changing the previous optimal path in the network[4]. The result shows that any perturbation between two nodes that stays within its tolerance bound defined below, does not affect the optimality of the tree.

Definition 1. *Given a spanning tree of an weighted acyclic graph, the* ***arc tolerance*** *is the maximum increment or decrement in the length of a single arc that does not affect the optimality of the tree.*

In Fig.1, for example, thick solid lines and dashed lines represent an edge(e) on an optimal tree(T) and a non-tree edge($\grave{e}$) respectively. Now, we are curious about e_3^1's tolerance bound. If e_3^1 itself is cut off, two node sets($N^-(e_3^1)$ and $N^+(e_3^1)$) can be easily identified. Suppose that $\grave{e}_9^6$ and $\grave{e}_5^2$ has shortest length in each direction among all $\grave{e}$'s crossing $N^-(e_3^1)$ and $N^+(e_3^1)$. Then, the e_3^1's bound is calculated as:

$$\begin{aligned} |e_2^1|+|\grave{e}_5^2|-|e_5^3| \le |e_3'^1| \le |e_2^1|+|e_4^2|+|e_9^4|-|e_6^3|+|\grave{e}_9^6|, \\ |e_3^1|-\Delta e_5^2 \le |e_3'^1| \le |e_3^1|+\Delta e_9^6, \\ -\Delta_5^2 \le \delta(e_3^1) \le \Delta e_9^6 \end{aligned} \quad (1)$$

Thus, both $\grave{e}_9^6$ and $\grave{e}_5^2$ hold the bound of e_3^1. From the inequality, it is vital to view $\Delta e_5^2$and Δe_9^6 as the *cost* to pay when, by mistake, we expand T with a non-tree edge $\grave{e}_5^2$ and $\grave{e}_9^6$ respectively. If the change in the arc length does not satisfy this bound, a new optimal tree T' should be constructed from scratch.

[1]Last August, DNA/RNA sequence data was reached to 100 billions bases of the genetic codes over 165,000 organisms

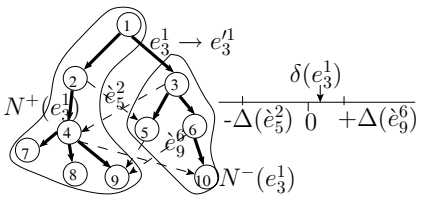

Figure 1: single arc tolerance evaluation

2.2 Alignment Graph and Problem Definition

Given two sequences $s_1[1\cdots M]$and $s_2[1\cdots N]$, a sequence alignment is to transform one into the other to maximize the total alignment value with a pairwise scoring matrix defining a similarity score between two symbols. Let $d_{i,j}$ be the optimal score of $s_1[1\cdots i]$ and $s_2[1\cdots j]$. A DP procedure on the two sequences yields an alignment as following.

$$d_{i,0} = i\cdot\Sigma(-,s_2[1]), d_{0,j} = j\cdot\Sigma(s_1[1],-)$$
$$d_{i,j} = \max\begin{cases} d_{i-1,j}+\Sigma(s_1[i],-), \\ d_{i-1,j-1}+\Sigma(s_1[i],s_2[j]), \\ d_{i,j-1}+\Sigma(-,s_2[j]) \end{cases} \quad (2)$$

As illustrated in Fig. 2, the usual DP algorithm on linear sequences can be generalized to weighted acyclic graphs(G) where the edge length is represented by one of scores of (mis)match and indels(i.e., insertion and deletion). Then the alignment corresponds to source-to-sink optimal paths(O) in the sequence graph[7].

Let us now formulate our problem. In Fig 2, we already obtained an alignment, O from s_1 and s_2. Now, suppose a symbol, $s_2[2]$ has bee updated. How do we obtain a new alignment O' and its new score without running another DP procedure on the new sequence? In general, it is difficult to reuse previous results of DP since it is based on the divide-and-conquer method to solve problems by combining the solutions of subproblems recursively as in Eq2. Let $d_{i,j}$ be a path distance from a root to a node $n_{i,j}$ in T. The perturbation changes all edge lengths between column 1 and 2 *simultaneously* and then each max operations at column 2 may yield a new distance path $d'_{i,2}$ with a new back-trace pointer. Furthermore, updates in multiple columns make reusability of DP even more difficult. Below, we discuss about the previous DP intermediary result can be useful to boost up an update matching when we encounter a perturbed sequence.

3. DYNAMIC SENSITIVITY ANALYSIS

Our proposed algorithm is bi-fold. First, we describe how the single arc tolerance analysis can be extended to solve our updating alignment problem. Furthermore, we discuss which information during the DP procedure should be stored in the off-line computation. Secondly, when a perturbation occurs, we describe how the information can be used in the on-line computation.

3.1 The Relative Node Tolerance Bound(RNTB)

We first consider how the updated distance from the root node to each node in a column can be decomposed so that the prior calculations can be reused. In Fig.2, let $d_{i,2}$ be a distance from $n_{1,1}$ to $n_{i,2}$ and $\delta(d_{i,2})$ be a perturbation in $d_{i,2}$, i.e., the new distance in column 2 is updated to $d'_{i,2} = d_{i,2}+\delta(d_{i,2})$. If the node distance is recomputed, we can perform the sensitivity analysis on the node as follows.

Definition 2. *Given an optimal spanning tree $T(s_1,s_2)$ corresponding to $G(s_1,s_2)$ that is a directed acyclic alignment graph of two sequences s_1 and s_2, the **node tolerance** is the maximum increment $\delta^+(n_{i,j})$ or decrement $\delta^-(n_{i,j})$ in the distance of a single node $n_{i,j}$ from a root that does not affect the optimality of $T(s_1,s_2)$.*

Now note that the result from Definition1 is not applicable to our problem *directly* since a perturbation affects the distance of all nodes in a column to a root node simultaneously. We therefore consider all pairs of nodes that sit on two branches connected by a non-tree edge downstream from the perturbation and calculate a bound on the relative perturbation between the distances from a root node to each pair of such nodes that will not affect optimality. We call this bound a *relative* node tolerance(RNTB).

Definition 3. *The **relative node tolerance bound** $\Phi_{(u:v,j)}$ is the maximum relative increment $\delta^+(d_{u:v,j})$ or decrement $\delta^-(d_{u:v,j})$ between the distances $d_{u,j}$ and $d_{v,j}$ of the two nodes from a root node that does not affect the optimality of $T(s_1,s_2)$.*

Each non-tree edge $\grave{e}$ not on T has an extra cost $\Delta(\grave{e})$ that must be incurred if $\grave{e}$ becomes part of T[10]. We collect all $\grave{e}$ and sort the $|\Delta(\grave{e})|$ in an increasing order. Next, starting from the smallest value, we assign the bounds to each pair of nodes. Back to our example, suppose that $\grave{e}_{4,5}^{4,4}$ has the smallest cost[2]. While tracing back from both the tail and the head of the $\grave{e}_{4,5}^{4,4}$, we propagate $\Delta(\grave{e}_{4,5}^{4,4})$ along the two branches, b_1 and b_2 shown in Fig.2. In general, $\Delta(\grave{e})$ is propagated as $\Phi_{(u:v,j)}$ for a pair of nodes on each branch all the way back to the root node until we come across the first node whose RNTB has been assigned in a previous propagation step or a shared node between the two branches. Refer to [3] for more detailed discussion.

3.2 Delta Propagation

Once the calculation of all RNTBs is completed, a reusable DP-based rematching procedure is available through the on-line computation. Note that in Fig.2, two perturbations occur in column 2 and 5. In column 2, whenever $\delta(d_{u:v,2})$ of a pair of nodes $n_{i_u,2}$ and $n_{i_v,2}$ stays within their $\Phi_{(u:v,2)}$'s, each $\delta(d_{i,2})$ is added into a node that is on its corresponding branch at column 5, right before the next perturbation, $\delta(d_{i,5})$. This means that T' reuses the previous DP's outcome for this skipped part of the alignment without breaking a partial tree starting from column 2 to 5 $T_{2\mapsto 5}$. We call this procedure *delta propagation*.

3.2.1 Reduced Resource Selection

Unfortunately, a complete set of $\Phi_{(u:v,j)}$ constraining all pairs of nodes in each column j takes $O(M^2N)$ space. In general, it is infeasible to store all RNTBs. In building DP resource of RNTBs from the previous section, therefore, we use two

[2]i.e.,$-(d_{3,4}+e_{4,5}^{3,4})+(d_{4,4}+\grave{e}_{4,5}^{4,4})$

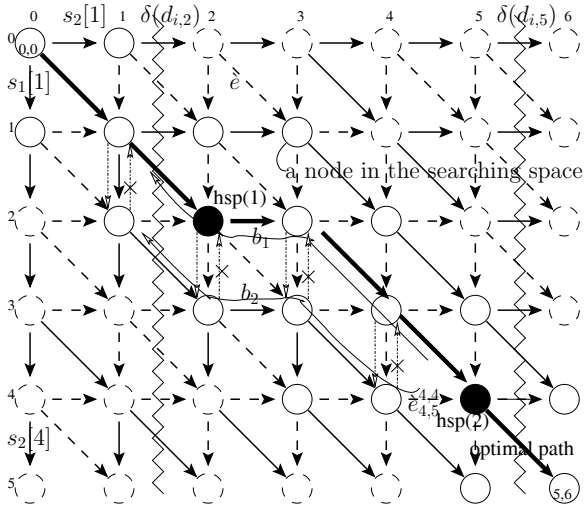

Figure 2: alignment WAG

procedures to select a subset of RNTBs to store in the off-line phase of the algorithm. This yields a reduced search band around T. The first procedure is a novel **static searching space** constrained to a partial tree T. Let T^B, $d^B_{i,j}$, and $d^F_{i,j}$(i.e., $d_{i,j}$) be a backward tree from $n_{M,N}$ to $n_{0,0}$ and the backward and forward distances respectively. By combining both forward node distance and backward distances with a weighting factor, we only include in our evaluation of RNTB nodes for which the sum distance exceeds a certain threshold. Let ξ be a threshold for sum distances and $\hat{i}$ be a row index satisfying the maximum distance. The reduced searching space is obtained by:

$$\frac{1}{2}(d^F_{i,j}+d^B_{i,j}) - (\omega_1 \cdot d^F_{i,j} + \omega_2 \cdot d^B_{i,j})) \leq \xi \qquad (3)$$
$$w_1 + w_2 = 1.0$$

Remark 1. *Assuming that the change in backward distance is negligible(i.e.,$d'^F_{i,j}+d'^B_{i,j} \approx d'^F_{i,j}+d^B_{i,j}$) in column j, let a new maximum sum of node distance be $\hat{d}'^{FB}_{i,j}$. If $d'^F_{i_1,j}+d^B_{i_1,j} \geq d'^F_{i_2,j}+d^B_{i_2,j}$, then $\hat{d}'^{FB}_{i,j} - (d'^F_{i_2,j}+d^B_{i_2,j}) \leq \hat{d}'^{FB}_{i,j} - (d'^F_{i_1,j}+d^B_{i_1,j})$.*

The parameters in Eq.3 are obtained empirically while a trade-off between performance and correctness is considered. In contrast to the classical heuristic approach which is based on the evaluation of a distance at the present node and estimated distance from the node to the goal node in the sense, the backward distance information is used in our off-line analysis. For instance, suppose that there is a consecutive matches from $n_{3,3}$ to $n_{4,4}$ in Fig.2. Then a decision of the searching space based on 3 increases the probability that a T' contains $n_{3,2}$ which is followed by the matching spots.

The second procedure keeps segments that constitute highly matching substrings. It is obvious that the **high score pairs**(HSPs) must be tolerant of perturbation. The tolerance evaluation test is performed on these segments only.

Our reduced RNTBs selection approach has two benefits. First, we can alleviate the expensive tolerance evaluation while maximizing the chance that the relative perturbations will be in the set of the RNTBs that we kept since many branches likely share the upstream HSP node. Second, the total number of tolerance evaluations can be reduced significantly. This can be also validated by the following remark.

Remark 2. *If a new alignment O' exists within a subtree $\dot{T}$ that is confined to the searching band in T, the delta propagation does not deform the correct O'.*

Proof. When a non-tree edge $\grave{e}^+ \in \dot{T}$ replaces a RNTB assignment at a column j of an $\grave{e}^- \notin \dot{T}$ having a shorter cost, $\Delta(\grave{e}^+)$ is still the smallest cost in $\dot{T}$ since a new $\dot{T}'$ generated by $\Phi_{(u:v,j)}$ of $\Delta(\grave{e}^+)$ and the delta propagation is equivalent to that of $\Delta(\grave{e}^-)$ at least within the searching band, $\dot{T}$. □

3.2.2 Tolerance Evaluation For The Last Perturbation

Once the perturbation of each node satisfies its relative tolerance bound in a column, the alignment process can skip over all unperturbed columns before the next perturbation. Otherwise, the evaluation will be accomplished in the next selected column assuming that there was a perturbation in that column. Thus, multiple perturbations can be handled generally by this fashion. More interestingly, there is a reason that we associate to the nodes of a selected column their backward distance $B^F_{i,j}$. If there is no more perturbation remaining during this rematching procedure, we can use Hirschberg's algorithm[5] for the final tolerance evaluation with lower complexity $O(M)$ rather than $O(M^2)$.

Remark 3. *If there is no more perturbation right after a column j that a DP reaches while updating T' and if a node $n_{r,j}$ that satisfies* $\arg\max_r \{d'^F_{r,j}+d^B_{r,j}\}$ *is equal to the previous optimal node, $n_{i,j}$ at j, the rest of new alignment is same as the previous one.*

This observation follows from the fact that there is no perturbation in the backward direction. Since $d'^B_{r,j}$ is equal to $d^B_{r,j}$, only $\delta(d_{i,j})$ is added to the destination node distance $d_{N,M}$ in $O(N)$ time. When either single or a burst type of perturbation occur, this evaluation is valuable if we seek high performance since we do not take care of the rest of T' any more.

The pseudo code of algorithm 1 summarizes our off-line computation of RNTB assignment for a selected resource. Once the perturbations is identified in an updated sequence, the on-line algorithm 2 performs a matching process initiated from a HSP column closest to the first updated column.

Algorithm 1 off-line dynamic sensitivity analysis

```
perform NW DP of Eq.(2)
the HSP column selection on the optimal alignment
obtain the searching band by Eq.(3); Ṫ
Q← sort(Δ(è⁺), ASCEND), where è⁺ ∈ Ṫ
repeat
    è ← pop(Q)
    (n⁻, n⁺) ← (p(head(è)), tail(è))
    repeat
        if Φ_(n⁻:n⁺,c)==NULL then
            Φ_(n⁻:n⁺,c) ← Δ(è)
        end if
        (n⁻, n⁺) ← (p(n⁻), p(n⁺))
    until n⁻ != n⁺ OR Φ_(n⁻:n⁺,c) !=NULL
until Q != EMPTY
```

Algorithm 2 on-line reusable alignment

```
1: perform resource mapping procedure
2: while c ≤ length(s'_2) do
3:    perform NW DP of Eq.(2)
4:    if c==HSP column then
5:       if perturbation==0 then
6:          perform arg max comparison of Remark 3.
7:       else
8:          δ_r ← d'_{r,c} − d_{r,c} for srchBandRow(r,c) ∀r
9:          tolerant ← true, u ← srchBandRow(1,c)
10:         while tolerant AND u ≤srchBandRow(end,c) do
11:            if δ_{u,v:j} is NOT in Φ_{(u:v,j)} ∃ν then
12:               tolerant ← false
13:            end if
14:            u ← u+1
15:         end while
16:         if tolerant then
17:            delta propagation
18:         else
19:            c ← c+1
20:         end if
21:      end if
22:   else
23:      c ← c+1
24:   end if
25: end while
```

4. EXPERIMENTAL RESULTS

There are two main computation phases in the proposed reusable alignment procedure. First, a recursion of the DP module(3:3) would be performed for columns of perturbations source or columns that exceed the node tolerances. The computational time on this module is called "DP time". Second, the routines(5:21) include a a module for detecting HSP and the tolerance evaluation and delta propagation. Similarly, the computational time on this module is called "control time".

In our experiment, we collect a pair of amino acid sequences with similarity that is greater than 20%. Those samples include a protein such as *leghemoglobin*, *beta globin*, *serine protease*, *native elastase*, *hemoglobin beta embroynic* etc. No more than 5% HSP samples are stored in the off-line computation when we select window size and the number of hits as 5 and 3 respectively. Also, we confine our searching band to yield 95% correctness of realignment under the condition that a maximum of 4 perturbations are allowed to occur.

We study the effect of three variables on the performance of the algorithm in table 4:(1)the number of perturbations in a sequence, (2)the sequence length, and (3)the burstiness of the perturbations. It is obvious that more perturbations require more time to rematch since more columns must be analyzed in the updated sequence to detect the valid previous optimality. When the proposed algorithm handles a large number of perturbation, performance deteriorates since frequent visits to the control module consume enormous computation. Next, the proposed scheme outperforms NW more with longer sequences since it performs big jumps over the unperturbed parts of the sequences (e.g., it consumes only 4.6% of the calculations used by NW in the third experiment corresponding to dependency II). Finally, we observe that a new sequence with a burst type of perturbation uses less computational time than the case where perturbations are scattered out. This observation is consistent with the result from the tolerance evaluation for the last perturbation.

dep.	pert#	length	local	NW†	Proposed†	LOCUS
	1	210	1.0	72.92	8.00	1SGC : 3EST
	2	210	1.0	73.72	10.04	
I	4	210	1.0	72.12	14.84	
	2	146	1.0	35.64	6.40	P27199:NP_000509
	2	318	1.0	162.24	14.04	CAA31435:NP_680093
II	2	544	1.0	591.64	27.32	YP_273777:YP_244613
	3	120	0.2	34.84	3.60	NP_001004376
	3	120	0.4	36.44	4.80	: NP_001015058
III	3	120	0.8	36.04	5.60	

Table 1: experimental result for 3 dependencies(†time(ms))

5. DISCUSSION

Our innovative sensitivity analysis has been extended to a general DP including any type of perturbation in practice. We capture a minimum information to perform an efficient realignment for either variant or slightly updated sequences. In this work, we observe that the efficient data structure and a number of calls of the control module holds the key to a higher performance. In particular, we notice that the higher similarity alignment increases a probability that perturbations is within a set of their RNTB[3]. In our future work, we plan to apply a contour derived by suboptimal path into our static searching space such that the probability is maximized. It would be interesting to compare the proposed approach with any standard pruning algorithm[4]. The preliminary implementation of our approach is available at *www.ece.umn.edu/users/hongcj92/.*

REFERENCES

[1] S. B. Needleman and C. D. Wuncsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, 48:443-53

[2] http://www.ncbi.nlm.nih.gov/

[3] C. Hong, A. H. Tewfik, and D. H. Du, "Handling Updates of A Biological Sequence Based on Hidden Markov Models, " *EUSIPCO,* 2005

[4] D. Shier, "Arc tolerances in shortest path and network flow problems," *Networks* 10:277-91, 1980.

[5] D. S. Hirschberg, "Algorithms for the longest commonsubsequence protblem," *J. ACM,* 24:664-75, 1977

[6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on SSC4,* 2:100-7, 1968

[7] D. Gusfield "Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology," *Cambridge University Press,* 1997

[8] R. Durbin, S. Eddy, A. Krogh, G. Mitchison, *Biological Sequence Analysis: probabilistic models of protein and nucleic acids*, *Cambridge University Press,* 1998.

[9] A. G. Clark and T. S. Whittam, "Sequencing Errors and Molecular Evolutionary Analysis," *Mol. Biol. Evol.* 9(4):744-52, 1992.

[10] D. Eppstein, "Finding the K Smallest Spanning Trees," *Lecture Notes in Comp. Sci.*, 447:38-47, 1990

[3] An average probability is 70% in our experiment

[4] The proposed algorithm's DP iteration consumes only 16% of our searching space