HANDLING UPDATES OF A BIOLOGICAL SEQUENCE BASED ON HIDDEN MARKOV MODELS

Changjin Hong, Ahmed H. Tewfik, and David H.C. Du

Digital Technology Center, Department of Electrical and Computer Engineering, University of Minnesota
200 Union Street. SE, 55455, Minneapolis, USA
phone: +1.612.624.5285, fax: +1.612.625.4583, email: {hongcj92, tewfik}@ece.umn.edu, {du}@cs.umn.edu

ABSTRACT

The computational complexity of evaluating homologies between a gene sequence and profile Hidden Markov Models (HMMs) is relatively high. Unfortunately, researchers must re-evaluate matches every time they discover an error in a sequence or encounter a mutation of the sequence. Since these occurrences are frequent, it is desirable to have a low complexity procedure for updating the matching result when a small perturbation in a given input gene sequence is observed. In this paper, we describe such a procedure based on a sensitivity analysis of the Viterbi algorithm used to evaluate the similarity of an unknown gene sequence and a profile HMMs. By extending single arc tolerance bounds to the evaluation of the relative change in all nodes' distances from a root node, our algorithm skips all unperturbed parts of a sequence. As a result, our proposed algorithm can update the matching decision in only 20% of the time required by the current approach that computes a new match with the perturbed sequence and base HMM model.

1. INTRODUCTION AND PREVIOUS WORK

HMMs are the underpinning of the sophisticated analysis techniques used in molecular computational biology, including Gene finding, phylogenetic analysis, and protein structural modeling[1]. The Viterbi decoding algorithm measures a similarity of an unknown sequence with profile HMMs[2] that model families of sequences. Recently, it has become more important to detect remote homologies by querying a newly discovered sequence against large libraries of HMMs[3]. However, a major challenge in real wet labs is that most gene sequences are quite lengthy and contain errors with high probability. Therefore, even a single correction of a symbol of a sequence will trigger all similarity recomputations, a process that takes hours or days. Furthermore, if a mutation has multiple sequences, the similarity evaluation must be repeated for all these sequences.

In order to cope with these update problems, it is imperative to have an innovative framework for the sensitivity analysis of the Viterbi algorithm. Several efficient algorithms have been studied to alleviate this problem. They balance affordable computation with accuracy. For example, *the pruning approach* only keeps the K highest scoring paths with a sufficient likelihood of success[4]. This approach has a weakness in that it fails to detect 'dark horses'. *The token passing* approach explicitly keeps track of all useful paths history. *The stack decoding* alleviates the weakness of the pruning approach by estimating the best possible score of several paths[5]. However, it is very complex and highly sensitive to the selected heuristics.

Our work is based on the observation that dynamic programming is equivalent to finding the shortest path in directed acyclic graphs(DAG). A sensitivity analysis of dynamic programming has been accomplished in the sense that the *tolerance* to the perturbation to any individual edge can be evaluated [6]. Our work extends prior work on the sensitivity analysis of dynamic programs to examine the effect of perturbations in multiple edges on the previously identified optimal match. Our approach leads to a higher performance that is five times faster than the normal application of the Viterbi algorithm to the perturbed gene sequence. In section 2, we revisit the basic sensitivity analysis of dynamic programs. Our novel work is described to address the limitation of this prior analysis. In section 3, we describe our new procedure. In particular, we detail the construction and use of a "sensitivity repository" that helps cut computational complexity at run time when the matching decision needs to be updated. We also provide a performance evaluation. In section 4, we discuss future work.

2. THE LIMITATION ON THE STATIC SENSITIVITY ANALYSIS

2.1 Model with Assumptions

Given a model, $\lambda \equiv (A, B, \pi)$ containing the transition probability, the observation probability, and the initial state distribution respectively in a HMM, the Viterbi algorithm provides the optimal solution indicating which path $\hat{\mathbf{q}}$ most likely explains the version v of a sequence k, $\mathbf{O}^{k(v)}$ that is $o_1, \cdots, \tilde{o}_i, \cdots, \tilde{o}_j, \cdots, o_T$. Let the path $\mathbf{q}$ be $q_1, q_2, \cdots, q_T$, then

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}} P(\mathbf{O}^{k(v)}, \mathbf{q}|\lambda) \tag{1}$$

This alignment process has a complexity of $O(TS^2)$ in time and $O(TS)$ in space, where T and S specify the length and the number of states[7]. Let $G(\mathbf{O}^{k(v)}|\lambda)$ be a trellis graph that is obtained by using the negated logarithm of the conditional probabilities as arc lengths. A shortest path tree $T_B(\mathbf{O}^{k(v)}|\lambda)$ can be readily constructed by applying Dijkstra's algorithm (Fig.1). While assuming this λ is unchanged, $\mathbf{O}^{k(v)}$ can be a new version of an initial sequence $\mathbf{O}^{k(0)}$ that is $o_1, \cdots, o_i, \cdots, o_j, \cdots, o_T$. It is interesting to know that the $\mathbf{O}^{k(v)}$ generates a $G(\mathbf{O}^{k(v)}|\lambda)$ with only different arc length on column i and j from $G(\mathbf{O}^{k(0)}|\lambda)$. Let $n_{c,s}$ be the node in a column c and a state s. Assuming we have a distance $d(n_{c,s})$ of the path $P(B; n_{c,s})$ from B to node $n_{c,s}$, then it is necessary to define a *cost* for each non-tree edge $e \notin E(T_B(\mathbf{O}^{k(0)}|\lambda))$ where E is an edge set. The cost of

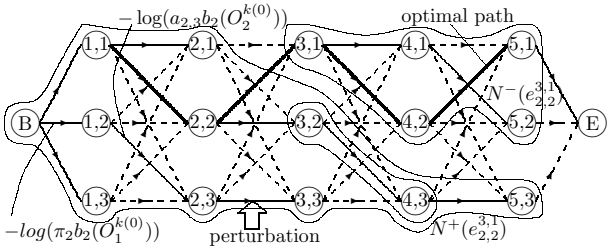


Figure 1: Viterbi Trellis Graph

$e_{c,s1}^{c+1,s2}$, i.e., the edge from $n_{c,s1}$ to $n_{c+1,s2}$ is

$$\Delta(e_{c,s1}^{c+1,s2}) = d(t(e_{c,s1}^{c+1,s2})) + w(e_{c,s1}^{c+1,s2}) - d(h(e_{c,s1}^{c+1,s2})) \quad (2)$$

,where h, t, and w signify head, tail, and the length of $e_{c,s1}^{c+1,s2}$ respectively. It is vital to view $\Delta(e_{c,s1}^{c+1,s2})$ as the cost to pay for selecting a non-tree edge. Any $\Delta(e)$ in $T_B(\mathbf{O}^{k(0)}|\lambda)$ is obviously 0. In addition, when an e is cut out of $T_B(\mathbf{O}^{k(0)}|\lambda)$, two subtrees T_B' and $T_{h(e)}'$ are derived. Let $N^+(e)$ be the set of nodes in T_B' and $N^-(e)$ those in $T_{h(e)}'$. Thus, a set of non-tree edges crossing between these node sets are defined as *cutset* of $G(\mathbf{O}^{k(0)}|\lambda)$. If $e \in T_B$, then

$$C^+(e) = \{\forall \acute{e}\ from\ N^+(e)\ to\ N^-(e), \acute{e} \notin T_B\} \quad (3a)$$
$$C^-(e) = \{\forall \acute{e}\ from\ N^-(e)\ to\ N^+(e), \acute{e} \notin T_B\} \quad (3b)$$

2.2 Problem Definition

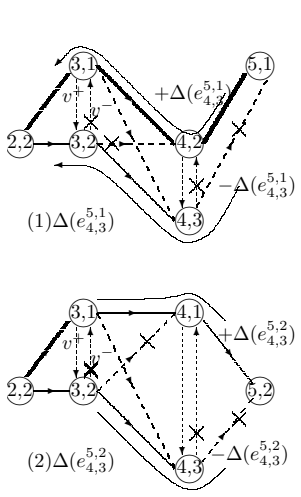
The arc tolerance specifies an allowable range of perturbation $\delta(e)$ in the arc length such that the previous optimal T_B is still effective in a new graph, as explained the following result.

Definition 1. *Given a spanning tree T of a DAG, the **arc tolerance** is the maximum increment or decrement in the length of a single arc without affecting the optimality of T.*

For a non-tree edge e, the lower arc tolerance bound $\delta^-(e)$ is $-\Delta(e)$. If $w(e)$ decreases below this bound, e itself becomes a tree edge. However, there is no upper arc tolerance bound $\delta^+(e)$ since e does not stay on T_B. For a tree edge e, on the other hand, its tolerance bounds are determined by non-tree edges of cutsets induced by removing the e. The following theorem is proven in [6].

Theorem 1. *If $e \in T_B$ then,*
$\delta^-(e) = \max\{-\Delta(\acute{e}) | \acute{e} \in C^-(e)\}$,
$\delta^+(e) = \min\{+\Delta(\acute{e}) | \acute{e} \in C^+(e), \acute{e} \neq e\}$

Assuming that $e_{2,1}^{3,1}$ has the smallest cost in $C^+(e_{2,2}^{3,1})$ of Fig1, for example, the maximum increase in $e_{2,2}^{3,1}$ is at most $\Delta(e_{2,1}^{3,1})$, its tolerance. Intuitively, this is obvious since any other e with larger cost in $C^+(e_{2,2}^{3,1})$ cannot break the optimality, as long as the $e_{2,1}^{3,1}$ *closest* to $e_{2,2}^{3,1}$ keeps away from $n_{3,1}$ of $N^-(e_{2,2}^{3,1})$ by limiting the length of $e_{2,2}^{3,1}$. Similarly, if $\delta(e_{3,1}^{4,3})$ is the smallest cost in $C^-(e_{2,2}^{3,1})$, a lower tolerance on $e_{2,2}^{3,1}$ is $-\Delta(e_{3,1}^{4,3})$. If the length of $e_{2,2}^{3,1}$ becomes shorter, it helps $e_{3,1}^{4,3}$ take away nodes from $N^+(e_{2,2}^{3,1})$ since the first *recipient* $e_{3,1}^{4,3}$ has the smallest cost. This sensitivity analysis is 'static' in the sense that the tolerance calculation for an edge is performed while all other edges stay constant. In our scenario, however, $\mathbf{O}^{k(v)}$ changes all arcs lengths that correspond to the modified columns in $G(\mathbf{O}^{k(0)}|\lambda)$ simultaneously. Therefore, the previous analysis is not applicable to $T(\mathbf{O}^{k(v)}|\lambda)$ directly. In reality, it turns out that the optimal tree depends on *relative* perturbation differences to the each state of the updated column. We refer to this concept as 'dynamic' perturbation.



(a) tolerance propagation

col		3		4	
state		δ^-	δ^+	δ^-	δ^+
1	2	?	$+\Delta(e_{4,3}^{5,1})$	?	?
	3	?	?	?	?
2	1	$-\Delta(e_{4,3}^{5,1})$	?	?	?
	3	?	?	?	$+\Delta(e_{4,3}^{5,1})$
3	1	?	?	?	?
	2	?	?	$-\Delta(e_{4,3}^{5,1})$	?

col		3		4	
state		δ^-	δ^+	δ^-	δ^+
1	2	?	$+\Delta(e_{4,3}^{5,1})$	?	?
	3	?	?	?	$+\Delta(e_{4,3}^{5,2})$
2	1	$-\Delta(e_{4,3}^{5,1})$	?	?	?
	3	?	?	?	$+\Delta(e_{4,3}^{5,1})$
3	1	?	?	$-\Delta(e_{4,3}^{5,2})$	?
	2	?	?	$-\Delta(e_{4,3}^{5,1})$	?

(b) repository tables

Figure 2: partial node tolerance assignments

3. DYNAMIC SENSITIVITY ANALYSIS OF THE VITERBI ALGORITHM

We first consider how updated distance of nodes in a column can be decomposed when a perturbation occurs. For example, when an updated sequence $\mathbf{O}^{k(v)}$ is given (i.e., there is "perturbation" in column 3 as in Fig.1), the new distance can be updated by

$$d'(n_{3,s2}) = d(n_{2,s1}) + \delta(o_3^{k(v)}),\ 1 \leq s1, s2 \leq S \quad (4)$$

where, $\delta(o_3^{k(v)})$ reflects a perturbation between column 2 and 3. There is no change in $T(\mathbf{O}^{k(0)}|\lambda)$ before column 3. Once the perturbation "$d'(n_{3,s2}) - d(n_{3,s2})$" is evaluated, we can figure out whether it propagates into the next column in terms of a distance to each node. If the node distance is carried out, we relate our sensitivity analysis to the nodes.

Definition 2. *Given an optimal $T(\mathbf{O}^{k(0)}|\lambda)$ corresponding to $G(\mathbf{O}^{k(0)}|\lambda)$, the **node tolerance** is the maximum increment $\delta^+(n)$ or decrement $\delta^-(n)$ in the distance of a single node n from a start node that does not affect the optimality of $T(\mathbf{O}^{k(0)}|\lambda)$.*

We switch over to the analysis of a node tolerance as in definition 2. Suppose that $e_{4,3}^{5,1}$ has the smallest cost in $G(\mathbf{O}^{k(0)}|\lambda)$. When tracing back from both tail and head of the $e_{4,3}^{5,1}$ toward B, we identify a subtree $T_{n_{2,2}}$ that has

two branches with these terminals. $T_{n_{2,2}}$ has root $n_{2,2}$ as a greatest lower bound (*GLB*) for both paths $P(n_{2,2};n_{5,1})$ and $P(n_{2,2};n_{4,3})$. Other edges would not participate in deciding node tolerance in this tree. When we evaluate the relative change in a node's distance, an upper bound of a state in any column coexists with the other state's lower bound in that column and vice versa. This is a consequence of the fact that $G(\mathbf{O}^{k(0)}|\lambda)$ has a specific structure of trellis where all nodes between only two consecutive columns are connected. There are $S-1$ bounds for each node with respect to other states. Each node of the subtree can be constrained with its tolerance by the following theorem.

Theorem 2. *If* $\Delta(e_0) = \min\{+\Delta(e)|e \in T(\boldsymbol{O}^{k(0)}|\lambda)$*, then* $\pm\Delta(e_0)$ *is propagated as* $\delta^+(n)$ *for all n's (except* $GLB(h(e_0),t(e_0))$ *and* $h(e_0)$*) in* $p(GLB(h(e_0),t(e_0));h(e_0))$ *and as* $\delta^-(n)$ *for* $p(GLB(h(e_0),t(e_0));t(e_0))$.

Proof. Since $p(B;GLB(h(e_0),t(e_0)))$ is shared between two paths, $\Delta(e_0)$ cannot affect any node's tolerance from the starting node to $GLB(h(e_0),t(e_0))$. Considering two virtual edges, v^+ and v^- between two distinct nodes in any column of $T_{GLB(h(e_0),t(e_0))}$, $h(v^+)$ corresponds to $t(v^-)$ and vice versa. $v^+$is directed toward $p(GLB(h(e_0),t(e_0));t(e_0))$ with a length corresponding to $d(t(v^+))$. Thus $\delta^+(v^+)$ is $+\Delta(e_0)$ while $\delta^-(v^-)$ is $-\Delta(e_0)$ according to Theorem 1. □

Corollary 1. *While the* $\delta(e_0)$ *propagates the tolerance for nodes of* $T_{GLB(h(e_0),t(e_0))}$ *according to Theorem 2, all e's from* $p(GLB(h(e_0),t(e_0));t(e_0))$ *to* $p(GLB(h(e_0),t(e_0));h(e_0))$ *are annihilated.*

By following the proof in Theorem 2, $+\Delta(e_0) = \delta^+(v^-) = \min\{+\Delta(\acute{e})|\acute{e} \in C^+(v^-)\}$ can be deduced. It is also clear that all $\acute{e}$'s are confined to $T_{GLB(h(e_0),t(e_0))}$. Therefore, $C^+(v^-) - \{e_0\}$ is unable to contribute to the tolerance assignment for any other nodes in $T(\mathbf{O}^{k(0)}|\lambda)$. For example, $+\Delta(e^{5,1}_{4,3})$ is stored as an upper bound for the pair path's state 3 in $n_{4,2}$(i.e., $\delta^+_3(n_{4,2}) \leftarrow +\Delta(e^{5,1}_{4,3})$). On the other hand, at $n_{4,3}$, $-\Delta(e^{5,1}_{4,3})$ is recorded as a lower bound for its pair state 2(i.e., $\delta^-_2(n_{4,3}) \leftarrow \Delta(e^{5,1}_{4,3})$). Similarly, in column 2, $n_{3,1}$ has $+\Delta(e^{5,1}_{4,3})$ as an upper bound for state 2 and $-\Delta(e^{5,1}_{4,3})$ as lower one for state 1 at $n_{3,2}$. Then $e^{4,2}_{3,2}$ is crossed out since it is not evaluated any more. This procedure is accomplished all the way down right before $GLB(h(e_0),t(e_0))$ as illustrated in Fig.2. Up to this point, only how the edge of smallest cost gets involved in deciding tolerance of nodes on its connected paths is considered. We consider the selection of the edge on which tolerance propagation should be performed and also resolve the case where more than two costs have conflicting bounds. Our algorithm is completed with the following theorem.

Theorem 3. *A node tolerance analysis resulting in any* $\Delta(e)$ *larger than that of the edge used in the previous analysis step according to Theorem 2 is not allowed to replace the node bounds which were previously assigned.*

Proof. Suppose we identify a pair of nodes, n^- and n^+ of which bounds has already been assigned in previous step. Considering a virtual edge v^+ from n^+ to n^-, $\Delta(e_0)$ is still smallest in the non-tree edges from $N(p(n^+;h(e_0))) \cup N(p(n^+;h(e)))$ to $N(p(n^+;t(e_0))) \cup N(p(n^+;t(e)))$. Thus, $\Delta(e)$ is discarded. □

Tolerance propagation by the cost of an edge e would be discontinued at either, coming across the first node that has been assigned previously or $GLB(h(e),t(e))$. Therefore, it is essential to sort the $\Delta(e)$ value in an increasing order before we directly evaluate the tolerance for each node. For example, let $\Delta(e^{5,2}_{4,3})$ be the next to smallest. After the cost is assigned to $n_{4,1}$ and $n_{4,3}$, it stops propagating at the pair of node, $n_{3,1}$ and $n_{3,2}$ of the column 3 (Fig.2(a)). In this case, $\Delta(e^{5,1}_{4,3})$ remains the smallest in a cutset from $N(p(n_{3,2};n_{4,3}))$ to $N(p(n_{3,1};n_{5,1})) \cup N(p(n_{3,1};n_{5,2}))$. Our dynamic sensitivity analysis of Vitebi algorithm is based on the off-line computation of a repository of node distance and tolerance bounds. A computational complexity of building a repository is $O(TS^2)$. However, the algorithm boosts up the overall computation's performance by removing unnecessary edges to be evaluated or avoiding duplicated tolerance propagation. The pseudo codes in Algorithm 1 summarizes our proposed solution. Once the preliminary repository is obtained, a matching process is initiated from the first different column in $\mathbf{O}^{k(v)}$ decoding procedure for updated sequence or slight varies of multiple observation as Algorithm 2. Once the perturbation of each node satisfies its tolerance bound in the column, the decoding process can hop on the next updated column which conceives that there was a perturbation in its own column. Therefore, decoding by multiple updates can be accomplished generally.

Algorithm 1 off-line dynamic sensitivity analysis

```
1: convert O^{v(0)}|λ(A,B,π) to DAG with "−log"
2: {p[n],d[n]} ← Dijkstra(DAG)
3: repeat (2) for ∀e^{t+1,j}_{t,i} ∉ T_B
4: Q← sort(E,cost,ASCEND)
5: repeat
6:    e ← pop(Q)
7:    (n_in, n_out) ← (tail(e), p*(head(e))) {*return a direct ascendant}
8:    repeat
9:       s_in ← state(n_in); s_out ← state(n_out);
10:      if δ^-_{s_out}(n_in) or δ^+_{s_in}(n_out)==NULL then
11:         δ^-_{s_out}(n_in) ← −Δ(e); δ^+_{s_in}(n_out) ← +Δ(e)
12:      else
13:         break
14:      end if
15:      S^old_out ← S_out; (n_in, n_out) ← (p(n_in), p(n_out))
16:      delete(Q, e^{c+1,S^old_out}_{n_in})
17:   until n_in != n_out
18: until Q != EMPTY
```

4. EXPERIMENTAL RESULTS

There are two main computation phases in our proposed decoding procedure. First, a recursion of the Viterbi algorithm would be performed for updated columns that exceed the node tolerances(3:4 of Algorithm 2). Secondly, it requires the computation to evaluate the perturbation with its tolerance(5:12) and propagating the updated distance to the next updated column(13:19), which we define as "control time". The overall computational complexity is $O(TS)$.

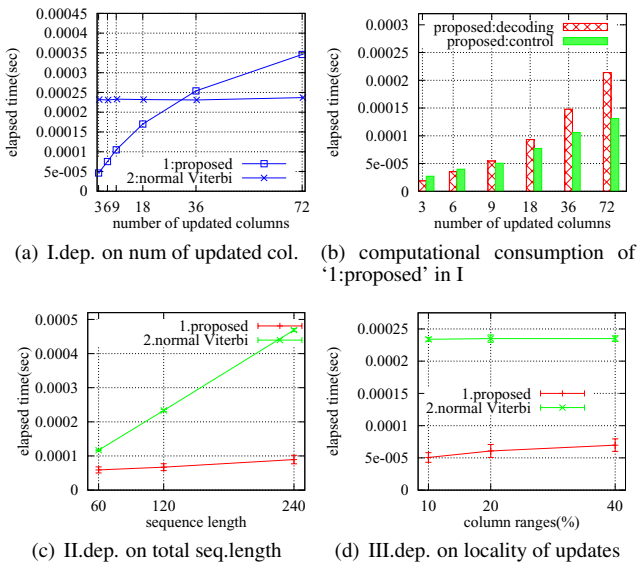
Algorithm 2 on-line decoding process for updated sequence

$u \leftarrow 1$, $c \leftarrow \delta^*(\mathbf{O}^{k(0)}, \mathbf{O}^{k(v)})[u]$ {*:it generates the vector of updated columns index}
while $c \leq T$ **do**
  update $d'(n_{c,i})$, $\delta_i \leftarrow d'(n_{c,i}) - d(n_{c,i})$ for each state
  $tolerant \leftarrow$ true, $j \leftarrow 1$
  while *tolerant* AND $j \leq S$ **do**
    for $(i = j+1$ to $S)$ **do**
      if $!(\delta_i^-(n_{c,j}) \leq (\delta_j - \delta_i) \leq \delta_i^+(n_{c,j}))$ **then**
        $tolerant \leftarrow$ false
      end if
    end for
    $j \leftarrow j+1$
  end while
  if *tolerant* **then**
    $next \leftarrow \delta(\mathbf{O}^{k(0)}, \mathbf{O}^{k(v)})[++u]$
    $d'(n_{next,i}) \leftarrow d(n_{next,i}) + \delta_i$ for each state* {*necessary to identify the corresponding branch}
    $c \leftarrow$ next+1
  else
    $c \leftarrow c+1$
  end if
end while

	$\lvert\vec{\delta}(\mathbf{O}^{k(0)}, \mathbf{O}^{k(v)})\rvert$	T	locality(%)
I	3	120	100
	6	120	100
	9	120	100
	18	120	100
	36	120	100
	72	120	100
II	5	60	100
	5	120	100
	5	240	100
III	6	120	10
	6	120	20
	6	120	40

Table 1: experiment configuration to evaluate three dependencies

In our experimental configuration, we synthesize a simple model, λ having 4 states and 10 symbols and we take into account three dependencies such as a number of updated columns, sequence length, and the locality in the sense of updated column ranges. Our algorithm for sensitivity repository and decoding procedure is implemented with ANSI C language and we perform the experiments under 1.5GHz Intel Pentium M processor®. In Fig.3(a), the computational time of the proposed algorithm highly depends on the number of updated columns since it leads a higher chance that it needs more columns for updated sequence to come back to the "right track" of the previous optimality. When we handle 3 updated columns, proposed scheme consumes .461μsec, which is only 20% computational time of the normal Viterbi(2.320μsec). We assess how much time each computation consumes in Fig.3(b). In 9 column updates, average 18.4 columns are computed for .545μsec and .506μsec is consumed for either evaluation or propagation. On the other hand, as updated symbols becomes dominant(i.e., 36 out of 120≃30%), the selected decoding time becomes enormously large, deteriorating the performance. In Fig.3(c), the computation time of the proposed scheme stays almost constant since only the propagation portion gets increased while elapsed time becomes lengthy in the normal scheme. In Fig.3(d), more interestingly, the proposed scheme takes longer to complete as the range of column updates in the



(a) I.dep. on num of updated col. (b) computational consumption of '1:proposed' in I

(c) II.dep. on total seq.length (d) III.dep. on locality of updates

Figure 3: elapsed decoding time

sequence increases since a new sequence having burst updated columns increases the probability that tolerances are exceeded.

5. DISCUSSION

In this work, the sensitivity analysis of the Viterbi algorithm is extended to capture a minimum information to perform an efficient decoding computation for either variant or slightly updated sequences. The major shortcoming of our approach is the space complexity of $O(TS^2)$ in the process of building a sensitivity repository. It is possible to reduce the space when we discard a relatively large bounds exceeding difference of any combination of symbol change with a compression scheme and also it would be valuable to appreciate a partnership with the K-best optimal decoding paths. In our practical implementation, we plan to integrate our seamless decoding procedure with sensitivity repository into object-based storage disk supporting the fast sequencing updates of cross-species genome.

REFERENCES

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

[2] P. Baldi, Y. Chauvin, "Smooth on-line learning algorithm for hidden markov models," *Neural Computation* 6:307-318, 1994.

[3] http://ccgb.umn.edu/

[4] R. Schwartz, Y.-L. Chow, "The N-best algorithm: An efficient and exact procedure for finding the N most likely sentence hypotheses," in *Proc. ICASSP*, pp.81-84, 1990.

[5] N. J. Nilsson, *Problem Solving Methods of Artificial Intelligence*. New York: McGraw-Hill, 1971

[6] D. Shier, "Arc tolerances in shortest path and network flow problems," *Networks* 10:277–291, 1980.

[7] G. D. Forney, Jr., "The Viterbi algorithm," in *Proc. of the IEEE*, 61:268-278, 1973.