A Fast Computation of Pairwise Sequence Alignment Scores Between a Protein and a Set of Single-Locus Variants of Another Protein

Yongwook Choi
J. Craig Venter Institute
9704 Medical Center Drive
Rockville, MD 20850
U.S.A.
ychoi@jcvi.org

ABSTRACT

Recently we have developed a new algorithm, PROVEAN (Protein Variation Effect Analyzer), for predicting the functional effect of protein sequence variations, including single amino acid substitutions and small insertions and deletions [2]. The prediction is based on the change, caused by a given variation, in the similarity of the query sequence to a set of its related protein sequences. For this prediction, the algorithm is required to compute a semi-global pairwise sequence alignment score between the query sequence and each of the related sequences. Using dynamic programming, it takes $O(n \cdot m)$ time to compute alignment score between the query sequence Q of length n and a related sequence S of length m. Thus given ℓ different variations in Q, in a naive way it would take $O(\ell \cdot n \cdot m)$ time to compute the alignment scores between each of the variant query sequences and S. In this paper, we present a new approach to efficiently compute the pairwise alignment scores for ℓ variations, which takes $O((n+\ell) \cdot m)$ time when the length of variations is bounded by a constant. In this approach, we further utilize the solutions of overlapping subproblems, which are already used by dynamic programming approach. Our algorithm has been used to build a new database for precomputed prediction scores for all possible single amino acid substitutions, single amino acid insertions, and up to 10 amino acids deletions in about 91K human proteins (including isoforms), where ℓ becomes very large, that is, $\ell = O(n)$. The PROVEAN source code and web server are available at `http://provean.jcvi.org`.

Categories and Subject Descriptors

I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search—*Dynamic programminag*; J.3 [**LIFE AND MEDICAL SCIENCES**]: Biology and genetics

General Terms

Algorithms, Theory

1. INTRODUCTION

As many large-scale genome sequencing projects generate a massive amount of sequencing data, a large number of sequence variation data also become available. This necessitates computational tools to predict the functional effect of amino acid variations and narrow down the list of causal variants for disease phenotypes. More than a dozen of algorithms have been developed to solve this problem [3, 7]. However, to the best of our knowledge, existing approaches are limited to work only for single amino acid substitutions. Thus, recently we have developed a prediction algorithm, PROVEAN (Protein Variation Effect Analyzer), which works not only for single amino acid substitutions but also for any other types of protein sequence variations including amino acid indels (insertions and deletions) and multiple amino acid substitutions [2].

In this approach, we introduced an alignment-based score as a new metric to measure the damaging effect of variations, which can be naturally applied to any type of protein sequence variations. More specifically, given a query protein sequence Q, PROVEAN first collects a supporting sequence set from related protein sequences, which are initially collected by the homology search tool BLAST [1]. Then, for a given sequence variation v, delta alignment scores (defined below) are computed with respect to each sequence S in the supporting sequence set. Finally, the PROVEAN score is computed from these delta alignment scores by averaging them with weights based on their sequence similarity.

The delta alignment score of v in Q with respect to S is defined as

$$A(Q_v, S) - A(Q, S),$$

where Q_v is a variant protein sequence of Q caused by v and $A(P_1, P_2)$ is the semi-global pairwise alignment score between two protein sequences P_1 and P_2. The delta alignment score can be obtained by computing two alignment scores, which can be solved by classical dynamic programming approach.

However, the problem arises when there are a large number of variations in Q, and for each of the variations the delta alignment score has to be computed with respect to a protein sequence S. Let us assume that there are ℓ variations, $v_1, v_2, \cdots, v_\ell$ in Q, whose lengths are bounded by a constant. To analyze the time complexity, let n and m be the lengths of Q and S, respectively. Then a naive approach would take $O(\ell \cdot n \cdot m)$ time since it needs to compute $\ell+1$ alignment scores, $A(Q_{v_1}, S)$, $A(Q_{v_2}, S)$, $\cdots$, $A(Q_{v_\ell}, S)$, and $A(Q, S)$, each of which takes $O(n \cdot m)$ time by dynamic programming approach.

In this paper, we present a fast algorithm, which takes $O((n + \ell) \cdot m)$ time to solve the same problem. The main idea of our approach is to further utilize the solutions of overlapping subproblems, which are already utilized by the dynamic programming approach. Specifically in our problem we make use of the fact that the two sequences Q and Q_{v_i} differ only in a small contiguous region. Thus many subproblems for computing $A(Q, S)$ are overlapped with ones for computing $A(Q_{v_i}, S)$.

In Section 2, we provide the preliminaries on pairwise alignments and formally define the problem. The main algorithmic and experimental results will be presented in Section 3. In Section 4, we will discuss how this idea can be generally used in other problems that exhibit overlapping subproblems and can be solved by dynamic programming.

2. BACKGROUND

In this section, we define some notations, review basic approaches on computing pairwise sequence alignment scores, and provide a formal definition of our problem.

Let X be a sequence (or protein sequence) of length n. Then X_i denotes the i-th letter (or amino acid) of X, and X_i^j denotes the substring of X, $X_i X_{i+1} \cdots X_j$ (if $i > j$, then X_i^j represents the empty string).

In aligning two protein sequences, a score is assigned to a matched or mismatched pair of amino acids, and a penalty is given to a gap, caused by an insertion or deletion in one of the sequences. The scores are given based on a scoring matrix such as the BLOSUM and PAM matrices [4, 5]. The alignment score is computed by summing all scores and gap penalties, and an optimal (highest scoring) alignment can be obtained efficiently using dynamic programming.

Sequence alignments generally fall into two categories, global alignment and local alignment. Global alignment finds an optimal alignment between the entire length of the two sequences [6]. By contrast, local alignment finds a region of strong similarity [8]. That is, it finds a highest scoring alignment between substrings of the two sequences.

2.1 Semi-global Pairwise Sequence Alignment

A semi-global alignment is a variant of global alignment where gaps at the beginning and end of the two sequences are not penalized. This is used by PROVEAN since it best suits the idea of the delta alignment score. In this paper, for simplicity we assume linear gap penalty, where each gap gets the same penalty, but the algorithm can be easily extended to affine gap penalty, where gap opening and gap extension penalties are different, without increasing the time and space complexities.

Algorithm 1 shows how an optimal (highest) semi-global pairwise alignment score between two protein sequences X of length n and Y of length m can be computed efficiently in $O(n \cdot m)$ time by dynamic programming. Here $D(i, j)$ stores the highest alignment score between X_1^i and Y_1^j with no penalties on the gaps at the beginning of the sequences. Note that we do penalize the gaps at the end of the sequences for scores in $D(i, j)$. $S(a, b)$ denotes the similarity score between amino acids a and b on the scoring matrix used, and $g < 0$ is the penalty for a gap. The optimal score is found by taking the maximum among all elements in the last row and the last column of D. This is because the penalties on the gaps at the end of the sequences were imposed for the scores in $D(i, j)$'s and thus for the semi-global alignment score, we need to find a maximum score between any prefix of X and the whole sequence Y or between the whole sequence X and any prefix of Y.

Algorithm 1 Computing semi-global pairwise alignment score between X of length n and Y of length m

```
// initialization
for $i = 0$ to $n$ do
  $D(i, 0) \leftarrow 0$
end for
for $j = 1$ to $m$ do
  $D(0, j) \leftarrow 0$
end for
for $i = 1$ to $n$ do
  for $j = 1$ to $m$ do
    // recursion steps
    $Match \leftarrow D(i-1, j-1) + S(X_i, Y_j)$
    $Delete \leftarrow D(i-1, j) + g$
    $Insert \leftarrow D(i, j-1) + g$
    $D(i, j) \leftarrow \max\{Match, Delete, Insert\}$
  end for
end for
// find an optimal score
$Max_1 \leftarrow \max_{j=0,\cdots,m} D(n, j)$
$Max_2 \leftarrow \max_{i=0,\cdots,n-1} D(i, m)$
return $\max\{Max_1, Max_2\}$
```

2.2 Delta Alignment Score and Problem

Given a query protein sequence Q of length n, its supporting protein sequence S of length m, and ℓ variations v_1, v_2, $\cdots$, v_ℓ in Q, we want to compute the delta alignment scores for each of ℓ variations with respect to S, that is, we want to compute

$$A(Q_{v_i}, S) - A(Q, S) \quad \text{for } i = 1, 2, \cdots, \ell.$$

Here a variation v can be a single or multiple amino acid substitution, insertion, or deletion whose length is bounded by a constant c (e.g. $c = 10$). Thus, generally any variation v is described by "k_1 amino acids starting at position p in Q are replaced by $X_1 X_2 \cdots X_{k_2}$" ($k_1 = 1$ and $k_2 = 1$ for single amino acid substitutions, $k_1 = 0$ for insertions, and $k_2 = 0$ for deletions.) In Figure 1, for example, the variation v is a replacement of amino acids $MALR$ starting at position 83 with amino acids IS, and thus $p = 83$, $k_1 = 4$, and $k_2 = 2$.

To compute the delta alignment scores for ℓ variations, we need to compute $\ell+1$ pairwise alignment scores, $A(Q_{v_1}, S)$, $A(Q_{v_2}, S)$, $\cdots$, $A(Q_{v_\ell}, S)$, and $A(Q, S)$. This would take $O(\ell \cdot m \cdot n)$ time in a naive way by running Algorithm 1 $\ell+1$ times.

3. RESULTS

In this section, we present our algorithm and its application, and we provide our experimental results.

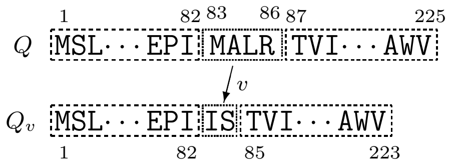

Figure 1: Example of variation v that replaces amino acids $MALR$ with IS.

3.1 Algorithm

Given a variation v in Q, our algorithm uses the fact that Q_v and Q differ only in a small contiguous region, as seen in Figure 1. Thus when we compute the alignment score between Q_v and S, we will reuse the results stored in the array D for Q and S as many time as we can, that is, for all the regions that are shared by Q_v and Q. Given the variation, "$Q_pQ_{p+1}\cdots Q_{p+k_1-1}$ is replaced by $X_1X_2\cdots X_{k_2}$," as seen in Figure 1 we can easily observe that there are two long substrings that are shared by Q and Q_v, that is,

$$Q_1^{p-1} = {Q_v}_1^{p-1} \tag{1}$$

and

$$Q_{p+k_1}^{n} = {Q_v}_{p+k_2}^{n+k_2-k_1}. \tag{2}$$

To utilize the second equality above, we need another two-dimensional array D_B whose values for the region do not depend on the amino acids $Q_p^{p+k_1-1}$ in the variant region. Thus, D_B is defined slightly differently from D as follows. Given two sequences X and Y, $D_B(i,j)$ is defined as the highest alignment score between X_i^n and Y_j^m with no penalties on the gaps at the *end* of the sequences. By definition, this array needs to be filled backwardly as described in Algorithm 2. From now on, the array D described in Section 2 will be called D_F since it is filled in a forward direction.

Algorithm 2 Computing *backwardly* semi-global pairwise alignment score between X of length n and Y of length m

```
for $i = 1$ to $n+1$ do
  $D_B(i, m+1) \leftarrow 0$
end for
for $j = 1$ to $m$ do
  $D_B(n+1, j) \leftarrow 0$
end for
for $i = n$ to $1$ do
  for $j = m$ to $1$ do
    $Match \leftarrow D_B(i+1, j+1) + S(X_i, Y_j)$
    $Delete \leftarrow D_B(i+1, j) + g$
    $Insert \leftarrow D_B(i, j+1) + g$
    $D_B(i,j) \leftarrow \max\{Match, Delete, Insert\}$
  end for
end for
$Max_1 \leftarrow \max_{j=1,\cdots,m+1} D_B(1,j)$
$Max_2 \leftarrow \max_{i=2,\cdots,n+1} D_B(i,1)$
return $\max\{Max_1, Max_2\}$
```

In our algorithm, given two sequences Q and S, we first build the two arrays, D_F and D_B. Then we keep both arrays to look up the values whenever we can reuse them to compute the alignment scores $A(Q_{v_i}, S)$ for $i = 1, 2, \cdots, \ell$.

Now assume that we have a variation v in Q, which is a replacement of $Q_pQ_{p+1}\cdots Q_{p+k_1-1}$ with $X_1X_2\cdots X_{k_2}$, and we want to compute $A(Q_v, S)$.

By (1), the values $D_F(p-1, i)$ $(i = 0, 1, \cdots, m)$ for Q and S would be the same as ones for Q_v and S. That is, $D_F(p-1, i)$ stores the highest alignment score between Q_1^{p-1} and S_1^i with no penalties on the gaps at the beginning, which is the same as the score between ${Q_v}_1^{p-1}$ and S_1^i. Similarly, by (2) the values $D_B(p+k_1, i)$ $(i = 1, \cdots, m+1)$ for Q and S would store the highest alignment score between ${Q_v}_{p+k_2}^{n+k_2-k_1}$ and S_i^m with no penalties on the gaps at the end.

Thus, starting from $D_F(p-1, i)$ values we build a temporary two-dimensional array D_T. $D_T(i,j)$ is defined as the optimal score between $Q_1^{p-1}X_1^i$ and S_1^j with no penalties on the gaps at the beginning of the sequences. After filling D_T, one can find the optimal score by looking at the k_2-th row in D_T and $(p+k_1)$-th row in D_B for Q and S since the two rows store the scores between ${Q_v}_1^{p-1} \cdot X_1^{k_2}$ and any prefix of S and the scores between ${Q_v}_{p+k_2}^{n+k_2-k_1}$ and any suffix of S, respectively.

To find the maximum score more efficiently, we also keep two one-dimensional arrays M_F and M_B of size $n+1$ as defined below. For $i = 0, 1, \cdots, n$,

$$M_F(i) = \max_{k=0,1,\cdots,i} D_F(k, m),$$

and similarly, for $j = 1, 2, \cdots, n+1$,

$$M_B(j) = \max_{k=j,j+1,\cdots,n+1} D_B(k, 1).$$

All elements in M_F and M_B can be easily computed while filling the arrays D_F and D_B.

The overall procedure to compute the alignment score between Q_v and S, given D_F and D_B for computing $A(Q, S)$, is described in Algorithm 3.

Algorithm 3 Computing semi-global pairwise alignment score between Q_v and S, given D_F and D_B for Q and S

```
// $v$: $Q_p^{p+k_1-1}$ is replaced by $X_1X_2\cdots X_{k_2}$
for $i = 1$ to $k_2$ do
  $D_T(i, 0) \leftarrow 0$
end for
for $i = 1$ to $k_2$ do
  for $j = 1$ to $m$ do
    if $i == 1$ then
      $Match \leftarrow D_F(p-1, j-1) + S(X_i, S_j)$
      $Delete \leftarrow D_F(p-1, j) + g$
    else
      $Match \leftarrow D_T(i-1, j-1) + S(X_i, S_j)$
      $Delete \leftarrow D_T(i-1, j) + g$
    end if
    $Insert \leftarrow D_T(i, j-1) + g$
    $D_T(i,j) \leftarrow \max\{Match, Delete, Insert\}$
  end for
end for
$Max_1 \leftarrow \max_{j=0,\cdots,m} D_T(k_2, j) + D_B(p+k_1, j+1)$
$Max_2 \leftarrow \max_{k=1,\cdots,k_2} D_T(k, m)$
return $\max\{Max_1, Max_2, M_F(p-1), M_B(p+k_1)\}$
```

Now let us analyze the time complexity to compute alignment scores for ℓ variant sequences. It takes $O(n \cdot m)$ time to fill the arrays D_F and D_B. Then, for each variation v_i, it takes $O(m)$ time to compute the delta alignment score

Table 1: Running time for computing delta alignment scores for ℓ variations (in seconds)

protein	algorithm	ℓ=100	ℓ=1K	ℓ=10K
Human TP53	naive	219	2141	21177
	ours	6	13	92
Ecoli LacI	naive	87	875	8574
	ours	2	6	40

$A(Q_{v_i}, S)$ using Algorithm 3 since k_2 is bounded by a constant. Thus the overall time complexity becomes $O((n+\ell)\cdot m)$.

3.2 Application and Experiment

Our algorithm made it feasible to build a large database of precomputed prediction scores. For academic and clinical researchers, we have built a database that stores precomputed PROVEAN prediction scores for all possible single amino acid substitutions, single amino acid insertions, and up to 10 amino acid deletions for each of about 91 thousand human protein sequences (including isoforms). In this case, the number of variations is $O(n)$. Thus our algorithm is faster than the naive approach by a factor of n. It would have been impossible to build such a large database if we had taken the naive approach.

For our test, we selected two well-studied proteins, Human TP53 and Ecoli LacI. For each protein, among the all possible simple variations described above, we randomly selected three subsets of size 100, 1K, and 10K, respectively. We measured the running time for computing delta alignment scores for each set. For TP53, 503 supporting sequences, whose length ranges from 21 to 788, were collected and used for the delta alignment score computation. For LacI, 218 supporting sequences whose length ranges from 56 to 741 were used. As shown in Table 1, our approach was much faster than the naive approach (more than 200 times faster for ℓ=10K). We used a linux machine with 64-bit Intel Xeon 2.27GHz CPU and 4GB memory.

4. DISCUSSION

In our algorithm, we used the idea of further utilizing precomputed results of overlapping subproblems. The same idea can be applied to similar problems that have the property of overlapping subproblems and are solvable by dynamic programming.

For example, a longest common subsequence (LCS) between two DNA sequences can be used to measure the similarity between the two sequences. The LCS problem can be solved by dynamic programming. Thus exactly the same idea can be applied in cases where longest common subsequences are needed for a set of variant sequences with respect to a fixed sequence.

The idea can also be applied to the classical 0-1 knapsack problem, which can be solved using dynamic programming. For example, let us consider two 0-1 knapsack problems where two item sets have many common items. In this case, by ordering the items so that the common items come first, we can reuse a part of the two-dimensional array used for the first problem when we solve the second problem.

5. CONCLUSION

We presented a fast algorithm to compute semi-global pairwise alignment scores between a protein sequence and each sequence in a set of variant sequences of another protein. Our approach proved to be fast and useful for building a database of precomputed prediction scores for a large number of human protein variations. We believe that a similar idea can be extended and applied to many other problems that are solvable by dynamic programming approach.

6. ACKNOWLEDGMENTS

The author would like to thank Agnes Chan and Granger Sutton for providing helpful discussion and comments. This work was supported by the National Human Genome Research Institute grant 5R01HG004701.

7. REFERENCES

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[2] Y. Choi, G. E. Sims, S. Murphy, J. R. Miller, and A. P. Chan. Predicting the functional effect of amino acid substitutions and indels. submitted, 2012.

[3] G. M. Cooper and J. Shendure. Needles in stacks of needles: finding disease-causal variants in a wealth of genomic data. *Nature Reviews Genetics*, 12(9):628–640, 2011.

[4] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of protein sequence and structure*, 5(suppl 3):345–352, 1978.

[5] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–10919, 1992.

[6] 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*, 48(3):443–453, 1970.

[7] P. C. Ng and S. Henikoff. Predicting the effects of amino acid substitutions on protein function. *Annual Review of Genomics and Human Genetics*, 7:61–80, 2006.

[8] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.