PERGAMON

Applied Mathematics Letters 16 (2003) 785–790

Applied Mathematics Letters

www.elsevier.com/locate/aml

An Optimization Approach to Multiple Sequence Alignment

F. Y. HUNT AND A. J. KEARSLEY
Mathematical and Computational Sciences Division
National Institute of Standards and Technology
Gaithersburg, MD 20899, U.S.A.

HONGHUI WAN
National Center for Genome Research
National Institutes of Health
Bethesda, MD 20894, U.S.A.

(Received February 2002; accepted March 2002)

Communicated by P. T. Boggs

Abstract—The problem of multiple sequence alignment is recast as an optimization problem using Markov decision theory. One seeks to minimize the expected or average cost of alignment subject to data-derived constraints. In this setting, the problem is equivalent to a linear program which can be solved efficiently using modern interior-point methods. We present numerical results from an implementation of the algorithm for protein sequence alignment.

Keywords—Linear programming, Sequencing, Alignment.

1. INTRODUCTION

The advent of large scale sequencing of DNA has triggered a massive accumulation of DNA sequence data and data about proteins—the products of the genes encoded by DNA. Methods for applying this information to fields such as law enforcement, biotechnology, and medicine or to gain crucial understanding of the biological significance and functionality of genes and proteins depend on a technique known as sequence alignment. For example, in database searching, a match between a query DNA sequence and selected members of the database is accomplished this way. This is also true in the case of proteins. Here, the purpose of a search is to gain information about the biological function or structure of the protein by identifying members of a database of proteins of known functionality and structure that are similar to the query protein.

Alignment methods are based on the underlying assumption that present day protein sequences from different species descended from common ancestor protein sequences. In the course of evolution, differences between descendants due to mutation and other genome changes often occur

0893-9659/03/$ - see front matter Typeset by AMS-TEX
PII: S0893-9659(03)00083-1

in such a way that biological function is preserved or only slightly modified. Given an underlying model of protein sequence evolution, sequences in the database that are related to the query in this way have high scores and are consequently identified as database "hits". Thus, aligning protein sequences is a way to infer information about protein function and structure. Given a set of sequences of letters, each letter representing an amino acid residue (the building block for proteins), an alignment is an array that displays the degree of relatedness of the sequences through matching and equivalent substitutions of corresponding amino acid residues and insertions of gap characters used to maximize the degree of similarity. The quality of the alignment is evaluated by assigning a total "score" or "cost" and the goal is to attain the "best" alignment. We can formalize this as follows.

DEFINITION. *(See [1].) Given a set of k sequences, $S = \{s_1, \cdots s_k\}$ where each s_i, $i = 1 \cdots k$, is a string of symbols from a finite alphabet $\mathfrak{G}$, a multiple sequence alignment is an array* $\mathsf{A} = \langle \sigma_1, \cdots \sigma_k \rangle^\top$ *with σ_i a string from the finite alphabet $\mathfrak{G}' = \mathfrak{G} \cup \{-\}$, and where $L = |\sigma_i|$, called here the length of the alignment, is the length of the string $\sigma_i, \forall i$. The sequence obtained from σ_i by removing all "–" gap characters is equal to s_i. Algorithms for multiple alignment are based on the minimization of the total cost over the set of all alignments* A,

$$\min_{\mathsf{A}} \sum_j c(\mathsf{a}_j), \tag{1}$$

where a_j *is the j^{th} column of the array, and* $c(\mathsf{a})$, *the cost of the aligned column* a, *is specified in advance [2].*

When each symbol is a letter of the alphabet and stands for one of the 20 amino acids or is a gap, a column of an alignment is assigned a cost based on the probability the acids represented by the letters in the column are descended from a common ancestor amino acid. The scoring function includes the possibility of gaps which correspond to the insertion or deletion of amino acids. Several standard scoring tables based on widely accepted models of protein evolution are used—the most common ones being the PAM or BLOSSOM matrices. Clearly, the choice of a score or cost function is an essential and perhaps the most important factor in achieving an effective and biologically meaningful solution. Nevertheless, in this work, we concentrate on the algorithm used to solve (1). Dynamic programming has been the most widely employed method for solving this problem [3] particularly for aligning small numbers of sequences. Given the increasing need for methods that handle larger numbers of sequences and given the immense computer power available in parallel and cluster networks, it is natural to investigate parallel algorithms for the solution of (1). Using a stochastic approach similar to hidden Markov models [4,5], aligned sequences are modeled as sample paths of a statistical process in an enhanced state space and alignment is treated as a stochastic control problem. Rather than calculate the minimum cost of a fixed alignment, we seek to minimize the expected or average cost of a family of alignments that are related to each other by a model. This can be done within the framework of Markov decision theory. One advantage of this is that the problem becomes equivalent to a linear programming problem that can be solved with fast numerical methods. Our purpose here is to introduce the concepts in Markov decision theory that define the problem and to report on the results of a numerical experiment in protein sequence alignment. Markov decision theory has been used to solve a variety of optimization problems in fields such as economics, biology, and network control. However, its application to multiple sequence alignment problems appears to be new.

2. MARKOV DECISION PROCESS

A Markov decision process or controlled Markov chain is a stochastic process (X_t, a_t), $t = 0, 1, \ldots$, where $X_t \in \mathbf{X}$, a finite set of m elements and $a_t \in \mathfrak{A}$, with $\mathfrak{A}$, is called the set of *actions*. An element of the set $\mathbf{X}$ is called a *state* of the process. For convenience, we will identify $\mathbf{X}$ with the set $\{1, 2, \ldots m\}$. Define the history of the process as the sequence, $h_t =$

$(x_1, a_1, \ldots, x_{t-1}, a_{t-1}, x_t)$. A *policy* is a sequence $\pi = (\pi_1, \pi_2, \ldots)$ of probability measures. If history h_t is observed at time t, then the controller chooses action a with probability $\pi_t(a \mid h_t)$. In full generality, a policy is a sequence of probability measures indexed by time, but in many applications of interest simplifying assumptions are made. This leads to the following set of special cases in order of increasing specialization.

- Markov policies: π_t is a function of x_t only.
- Stationary policies $\subset$ Markov policies: π does not depend on t, $\pi_t(a \mid h_t) = \mu(a \mid x_t)$.
- Stationary deterministic policies $\subset$ Markov policies: each $\pi_t(\mid x_t) = f(x_t)$ where f is a deterministic function defined on $\mathbf{X}$. Thus, $a = f(x_t)$.

Once an action at time t is chosen, the next state is chosen at random with probability $P_{ij}(a)$ where $X_t = i$, $X_{t+1} = j$, and $a_t = a$ where $i, j = 1, \ldots, m$. It can be seen therefore that we assume that the probability of transition from state X_t to state X_{t+1} depends only on the states and not on time, i.e., transitions are assumed to be stationary.

An alignment can be interpreted to be the sample path of a random process where the state of the process at time t is the t^{th} column of the alignment. Thus, X_t is a k-tuple of symbols from the alphabet $\mathfrak{G}'$. At the next time step the process transitions to a new state X_{t+1} that is determined by X_t and the action a_t taken by a controller at time t. From the point of view of stochastic control, an action is a decision at each time t about which positions in the $\mathrm{t}+1$ column will contain the gap symbol "$-$". Thus, a_t can be encoded as a k-tuple with elements 0 or 1 depending on whether or not a gap symbol is used in a given row. To fix ideas suppose our task is to find an optimal global alignment for three protein sequences. The MDP (Markov decision process) model for our application is based on a set $\mathfrak{G}$ of 20 letters representing the amino acids. The alphabet for the states is based on $\mathfrak{G} \cup \{-\}$, and thus, $\mathbf{X} = \{\mathfrak{G} \cup -\}^3$. The set of actions is $\mathfrak{A} = \{0, 1\}^3$. The protein alignment problem, therefore, has a state space with 21^3 elements and eight actions. Each pair (X_t, a_t) has a cost $C(X_t, a_t)$ associated with state X_t and action a_t. We will assume these costs are known. To find a global optimal alignment one formulates a (stochastic) control problem. The goal is to find a sequence of actions that results in a minimum alignment cost (or maximum alignment score). A variety of expressions for the total cost exists but one frequently discussed in the literature is the (finite horizon) expected average cost, given an initial state $X_0 = i$

$$\sum_{t=1}^{L} \mathbf{E}_\pi \frac{[C(X_t, a_t) \mid X_0 = i]}{L},$$

where L is the length of the alignment, i.e., the length of an aligned sequence. An expected discounted cost for the path $(X_0, a_0, X_1, a_1, \ldots)$ can also be defined (infinite horizon). For $0 < \alpha < 1$, we define

$$V_\pi^\alpha(i) = \mathbf{E}_\pi \left[\sum_{t=0}^{\infty} \alpha^t C(X_t, a_t) \mid X_0 = i \right]. \tag{2}$$

We note that although the index t goes to infinity, the range of the sum will be finite if the cost function vanishes after the last symbol of a sequence is reached. In the case where there are an infinite number of positive costs, the sum in equation (2) could diverge as $\alpha \to 1$, but it can be shown that, as α approaches 1, the quantity $(1 - \alpha)V_\pi^\alpha$ is nearly the expected average cost for large L when π is a stationary policy [6]. Let $V_\alpha(i) = \inf_\pi V_\pi^\alpha(i)$. Then $V_\alpha(i)$ is the least mean discounted cost for every starting state i. The goal is to find an optimal policy π^* such that $V_{\pi^*}^\alpha(i) = V_\alpha(i)$ for every state i. It turns out that such an optimal π^* exists and is often a stationary and/or deterministic policy. The derivation of the linear programming problem we sketch here is based on [7] (see also [6]). The transformation of the optimization problem stated

above begins with the fact that V_α satisfies the Bellman equation

$$V_\alpha(i) = \min_a \left[C(i,a) + \alpha \sum_{j=1}^{m} P_{ij}(a) V_\alpha(j) \right]. \tag{3}$$

Let $B(\mathbf{X}) = \{u \mid u : \mathbf{X} \to \mathbb{R}\}$, the set of real valued functions defined on $\mathbf{X}$ with norm $\|u\| = \sup_{i \in X} |u(i)|$. Define the operator $T_\alpha : B(\mathbf{X}) \to B(\mathbf{X})$, by

$$T_\alpha u(i) = \min_a \left\{ C(i,a) + \alpha \sum_{j=1}^{m} P_{ij}(a) u(j) \right\}.$$

When $0 < \alpha < 1$, T_α is a contraction mapping with unique fixed point V_α. Moreover, for any $u \in B(\mathbf{X})$, $T_\alpha^{(n)} u \to V_\alpha$ as $n \to \infty$. Suppose $T_\alpha u \geq u$. Then by repeated application of T_α, we have $V_\alpha \geq u$. Thus, V_α is the largest function satisfying the condition $T_\alpha u \geq u$. It is natural then to use the following method to calculate $V_\alpha(i)$:

$$\begin{aligned} &\text{maximize} && u(i), \\ &\text{subject to} && T_\alpha u(i) \geq u(i). \end{aligned}$$

Since the maximization occurs for each i one can replace u by $\sum_{i=1}^{m} u(i)$ where m is the number of states. The optimization problem for the expected discounted cost can then be written as

$$\begin{aligned} &\text{maximize} && \sum_{i=1}^{m} u(i), \\ &\text{subject to} && C(i,a) + \alpha \sum_{j=1}^{m} P_{ij}(a) u(j) \geq u(i), \end{aligned}$$

where one maximizes over all bounded nonnegative functions defined on $\mathbf{X}$ and where the constraint inequality holds for all a. Once such an optimal u is found, it can be shown to be V_α (see Section 3.2 in [6]). Note that an important issue in successful implementation of this method is the estimation of $P_{ij}(a)$ from alignment data. If the states defined by the sequences to be aligned is a subset $S \subset \mathbf{X}$, then the problem becomes

$$\text{maximize} \quad \sum_{i \in S} u(i), \tag{4}$$

$$\text{subject to} \quad C(i,a) + \alpha \sum_{j \in S} P_{ij}(a) u(j) \geq u(i), \tag{5}$$

for all a and $i \in S$.

Now let a_i be such that

$$C(i,a_i) + \alpha \sum_{j=1}^{m} P_{ij}(a_i) V_\alpha(j) = \min_a \left[C(i,a) + \alpha \sum_{j=1}^{m} P_{ij}(a) V_\alpha(j) \right]. \tag{6}$$

Then setting $f_\alpha(i) = a_i$ one defines an optimizing stationary deterministic policy. This policy gives us the sequence of actions that leads to the best alignment of sequences as defined by the expected discounted cost for a fixed α.

3. NUMERICAL RESULTS

Problems (4),(5) can be rewritten as a linear programming problem (LP) by splitting our u variable into two bound-constrained variables,

$$u(i) = \hat{u}(i) + \bar{u}(i), \text{ with } \quad \hat{u}(i) \geq 0, \ \bar{u}(i) \leq 0.$$

The LP can then be written

$$\max \sum_{i \in S} (\hat{u}(i) + \bar{u}(i)), \tag{7}$$

$$\text{s.t.} \ \sum_{j \in S} (I - \alpha P_{ij}(a)) \hat{u}(j) \leq C(i,a) \tag{8}$$

$$-\sum_{j \in S} (I - \alpha P_{ij}(a)) \bar{u}(j) \leq C(i,a) \tag{9}$$

$$\hat{u}(i) \geq 0, \qquad \bar{u}(i) \leq 0, \tag{10}$$

for all $i \in S$. This problem is potentially large scale with a sparse constraint matrix. The development of fast and efficient algorithms for the solution of linear programming problems is an active and productive research area (see, for example, [8,9]). The family of problems derived here presents an interesting challenge for software and computers. The objective function, as stated here, is a linear function however, one can foresee the addition of more complicated nonlinear penalty terms regularization terms or even more complicated definitions of optimality that give rise to truly nonlinear functions.

Below, we report on a sample test problem of aligning a set of sequences arising from the protein family cytochrome-p450. The constraint (or transition) matrices were computed from a set of 98 triples; each sequence of the triple having length 775. This specific problem contains 26,872 primal length 775. This specific problem contains 26,872 primal variables and 107,488 linear inequality constraints. The constraint matrix contains 132,984 nonzero values and every primal variable is bounded above and below. In this sample problem, as with most of our numerical experimentation to date, the constraints were degenerate at the solutions. That is, the collection of inequality constraints that are binding at solutions form a linearly dependent set. The primary intention of presenting this numerical example is to demonstrate the compatibility of our mathematical formulation with existing numerical algorithms.

In order to solve (7)–(10), we used an implementation of a *primal-dual* interior-point algorithm for linear programming problems [10] and an implementation of a *simplex* method [11]. In Table 1, we report the number of iterations and elapsed CPU time required to solve the test problem described above.

Table 1. Iteration count for linear program solvers (CPLEX 3.0 and PCx).

Method	Variables	Rows	Iterations
PCx	26,872	107,488	69
CPLEX	26,872	107,488	716

It is worth noting that both methods appear to be effective at solving the problem. Future work includes a more detailed numerical survey of the performance of optimization algorithms for the solution of more sophisticated nonlinear programming problem formulations based on (7)–(9). Currently, an implementation of computer codes that generate constraint matrices is underway [12].

4. CONCLUSIONS

A novel optimization approach to multiple sequence alignment has been presented along with promising preliminary numerical results. Using modern optimization techniques this approach provides a new way of addressing a large class of important bioinformatics problems. The algorithm presented in this paper successfully locates reasonable solutions to a typical instance of the multiple sequence alignment problem in a reasonable amount of (computational) time. The algorithm is quite flexible allowing a variety of cost functions to be employed and the addition (or relaxing) of constraints.

REFERENCES

1. C. Korostensky and G. Gonnet, *Gap Heuristics and Tree Construction Using Gaps*, (Preprint 1998).
2. A. Apostolico and R. Giancarlo, Sequence alignment in molecular biology, *J. Comput. Biol.* **5** (2), 173–196, (1998).
3. S. Needleman and C. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *J. Mol. Bio.* **48**, 443–453, (1970).
4. R. Durbin, S. Eddy, A. Krogh and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, (1998).
5. M. Waterman, *Introduction to Computational Biology*, Chapman and Hall, London, (1995).
6. E. Altman, *Constrained Markov Decision Processes*, Chapman and Hall/CRC, Boca Raton, (1999).
7. S. Ross, *Applied Probability Models with Optimization Applications*, Holden-Day, San Francisco, CA, (1970).
8. S. Wright, *Primal Dual Interior Point Methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA, (1997).
9. V. Chvátal, *Linear Programming*, W.H. Freeman and Company, New York, NY, (1983).
10. J. Czyzyk, S. Mehrotra and S. Wright, PCx User Guide, Technical Report OTC 96/01, Optimization Technology Center, (May 1996).
11. R.E. Bixby, Implementing the simplex method: The initial basis, *ORSA Journal on Computing* **4**, 267–284, (1992).
12. F. Hunt, A.J. Kearsley and A. O'Gallagher, Software for optimization-based multiple sequence alignment, Technical Report, Mathematical and Computational Science Division, National Institute of Standards and Technology, Gaithersburg, MD.