Computational Experiments with Multicriteria Sequence Alignment

Luís Paquete and João P.O. Almeida
CISUC, Department of Informatics Engineering,
University of Coimbra, Coimbra, Portugal
paquete@dei.uc.pt,jpoa@student.dei.uc.pt

Abstract

In this article we investigate the performance of multicriteria dynamic programming algorithm for pairwise sequence alignment that maximizes the number of matches and minimizes the number of indels or/and gaps. We give explicit recurrence relations for a number of multicriteria score functions. In addition, we provide estimates on the number of optimal alignments for pairs of random sequences, as well as computational results in two benchmark datasets.

1 Introduction

In the last few years, there has been a growing interest on the multicriteria formulation of many problems that arise in Bioinformatics [8]. Multicriteria optimization is concerned with simultaneously optimization of two or more conflicting criteria subject to constraints (see [4, 13, 18]). On one hand, it has the advantage of offering a trade-off set of alternative solutions to the decision maker. However, on the other hand, multiobjective optimization problems are not only more complex to solve than their single-objective counterpart as they also need more computational cost [3]. In this article, we focus on the multicriteria formulation of a classical bioinformatics optimization problem, the pairwise sequence alignment problem, and examine its tractability from a theoretical and practical point of view.

Sequence alignment is in the core of many Bioinformatics applications. It aims to identify regions of similarity in sequences of biological data. It is mainly used for the comparison of different sequences of the same gene, the search for a given pattern as subsequence in a database of strings, and as subprocedures for solving other more complex problems like the computation of approximate overlaps in DNA sequencing. They are also needed for the comparison of DNA or protein sequences of different organisms and are used as a measure of their relationship.

The procedure consists of inserting gaps between the residues so that similar symbols from several sequences become aligned. Usually, practitioners use a score function that is a convex combination of the number of matches and gaps. Efficient algorithms, such as the Needleman-Wunsch dynamic programming algorithm [14], are able to find optimal alignments for two sequences in an amount of time that is bounded by a polynomial function on the size of the sequences. However, the current software packages depend on the definition of coefficients in the score function. Therefore, the resulting alignment is heavily influenced by those coefficients. This would be acceptable if the best choice of those coefficients was clear *a priori*. Yet, this choice is an open problem, which depends on the functional and structural context of the sequences under analysis. As such, a procedure that returns a single final alignment eliminates many possible candidates that may be of interest for the practitioner.

Our work aims to present a number of possible alternative alignments, such that no two alignments are clearly better than each other with respect to the components of the score function. Those alignments are called *efficient*. Clearly, this different formulation brings several advantages to the practitioner: i) an optimal alignment for the scalar score function with positive coefficients is also an efficient alignment; ii) there may be more efficient alignments that are not optimal for any combination of coefficient values into convex score functions; iii) the number of efficient alignments with distinct score value is linear with respect to the size of the sequences. Therefore, multicriteria sequence alignment allows the possibility of exploring a tractable set of alignments that are not reachable by any other method and may give further biological insights. Moreover, it allows the practitioner to get rid of coefficients in the score function.

Interestingly, this problem has only been tackled by Royberg et al. [16]. These authors provide the first multicriteria formulation as well as an extension of Needleman-Wunsch algorithm to solve this problem. In this article, we extend the work of these authors by providing explicit recurrence equations to compute the set of optimal scores for three score functions and providing numerical results on randomly generated instances as well as on two well-known benchmark sets of instances. This article is organized as follows. Section 2 introduces the problem and some relevant theoretical properties. Section 3 gives the recurrence equations for three score functions. Sections 4 and 5 describe the experimental analysis. Finally, we conclude with Section 6.

2 Multicriteria sequence alignment

The problem of computing optimal alignments is a problem of sequences comparison. Sequences are aligned with each other by inserting gaps such that the aligned sequences coincide with each other. Usually, the way used for scoring an alignment consists of counting the number of matches and gaps. The optimal alignment is the one that maximizes a given convex combination of these two quantities. Other more complex score functions penalize the opening of gaps and/or rewards long gaps [6]. Other approaches use a scoring matrix that assigns different scores for different symbols at the same position in the alignment [9].

Most of these approaches to the sequence alignment problem rely on the *a priori* definition of coefficients that are assigned to the components of the score function. However, there is a considerable disagreement about how to weight each coefficient [6]. In order to minimize this drawback, parametric sequence alignment has been proposed [7]. The goal is to partition the coefficient space into convex regions such that in each region any alignment that is optimal for some choice of coefficients inside the region is optimal in that entire region and nowhere else. This is performed by the software package XPARAL that uses polygonal decomposition to find such optimal alignments. Several authors showed that the number of such regions is sublinear with respect to the sizes of the sequences [5].

A multicriteria formulation of the pairwise sequence alignment problem has been proposed by Roytberg et al. [16]. The authors consider a vector score function in which each component is associated to the occurrence of matches and gaps in a given alignment. Then, an alignment is efficient if it is maximal with respect to the component-wise ordering of the scoring of all alignments (by taking the opposite number of gaps).

More formaly, an *alignment* between sequences A and B is a pair of equal lenght sequences $\varphi = (A', B')$, where A' (respectively, B') is obtained by inserting a space into A (B) under the constraint that there can be no position in which both A' and B' have spaces. A *match* is a position in which A' and B' have the same symbol. An *indel* is a position in which either A' or B' has a space, while a *gap* is a sequence of one or more consecutive spaces in A' or B'. For the purpose of this article, we shall denote the number of matches, indels and gaps in φ by $\mathrm{m}(\varphi)$, $\mathrm{d}(\varphi)$, and $\mathrm{g}(\varphi)$, respectively.

We explore the multicriteria formulation of this problem as given by Roytberg et al. [16]. For the purpose of this article we shall consider the following three score functions

$$\begin{aligned}\mathrm{Score}_{\mathrm{md}}(\varphi) &= (\mathrm{m}(\varphi), -\mathrm{d}(\varphi))\\ \mathrm{Score}_{\mathrm{mg}}(\varphi) &= (\mathrm{m}(\varphi), -\mathrm{g}(\varphi))\\ \mathrm{Score}_{\mathrm{mdg}}(\varphi) &= (\mathrm{m}(\varphi), -\mathrm{d}(\varphi), -\mathrm{g}(\varphi))\,.\end{aligned}$$

We say that an alignment φ *dominates* an alignment φ' if each score component of the score function of φ is greater or equal than the corresponding score component of the score function of φ' and at least one of these inequalities is strict. Then, an alignment φ is *efficient* if there is no other alignment that dominates it. The *efficient set* contains only and all efficient alignments. We say that the image of the efficient set in the corresponding score function space is called the *nondominated set of scores.*

A negative result is that the problem of finding the efficient set of alignments is intractable, that is, the size of the efficient is exponentially large for any score function. We provide an artificial example of an intractable instance of this problem. Let A and B be two sequences where $A = \mathtt{A}^n$ and $B = \mathtt{G}(\mathtt{AG})^{2n}$. Then, $|A| = n$ and $|B| = 4n + 1$. Note that sequence B has $2n$ As. Therefore, there can be $\binom{2n}{n}$ possible choices of matching n As between the two sequences, and there cannot be other alignment with larger number of matches and lesser number of indels ($3n + 1$ indels). The same applies to the number of gaps (three gaps). Therefore, any of these $\binom{2n}{n}$ alignments is efficient.

Therefore, we should expect to incur in exponential time to find the efficient set in general. However, there exists a tractable number of nondominated scores, as shown in [16]. Given two sequences of size m and n, the number of nondominated scores is $O(n+m)$ for score functions $\mathrm{Score}_{\mathrm{md}}$ and $\mathrm{Score}_{\mathrm{mg}}$ and $O(n+m)^2$ for score function $\mathrm{Score}_{\mathrm{mdg}}$. Note that the number of matches is at most the size of the longest common subsequence of the two sequences and the number of indels and gaps is at most $m + n$, if there is no possible match between the two sequences[16]. Moreover, we should expect that the bound for score function $\mathrm{Score}_{\mathrm{mdg}}$ is not strict for most of the cases, since the increase on the number of indels should very often correspond to an increase on the number of gaps.

Another property of this problem results from the classical result in multicriteria optimization concerned with scalarized optimality [4]: an optimal alignment for the parametric sequence alignment (with positive coefficients) is also an efficient alignment. However, not

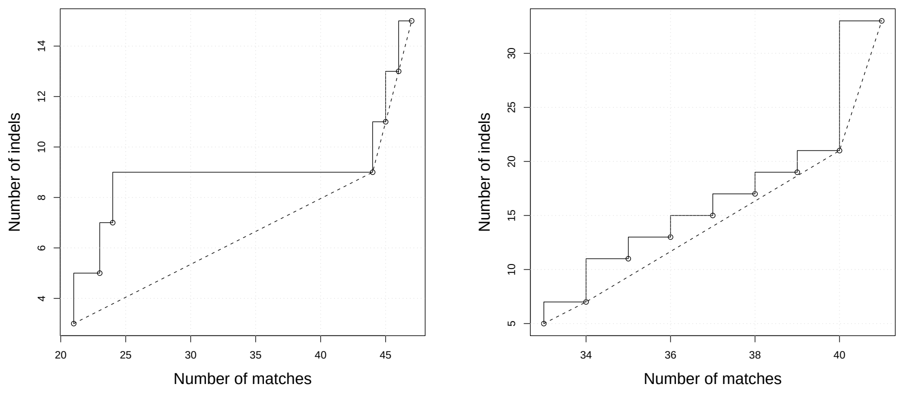

Figure 1: Nondominated scores of alignments in Figures 6 (left plot) and 7 (right plot).

all efficient alignments are optimal for parametric sequence alignment. Using a real-life case as an example, Figures 6 and 7 present the efficient alignments and corresponding nondominated scores that were found for two pairs of sequences of human S100 calcium-binding proteins, sequences P23297 and P60903, and sequences Q96FQ6 and P29034, respectively, taken from the database UniProtKB. Figure 1 plots the corresponding nondominated scores; the dashed line connects the scores that are located in the convex hull. Clearly, any parametric approach would miss the scores that do not lie in the convex hull envelope (see other example in [16]).

These results indicate that multicriteria sequence alignment provides further information for the pratictioner and that he should expect a small (tractable) number of nondominated scores to analyse.

3 A multicriteria dynamic programming algorithm

Roytberg et al. [16] presented the pseudo-code of a multicriteria dynamic programming algorithm that extends the classical Needleman-Wunsch algorithm to obtain the nondominated set of scores for the score function Score_{md}. This approach maintains nondominated partial alignments at each iteration, similar to the Nemhauser-Ullman algorithm to solve the knapsack problem [15] and further explored in the literature to solve multicriteria knapsack problems [1, 11].

In the following sections, we give a more explicit recurrence equations for computing the nondominated scores for Score_{md}, Score_{mg} and $\text{Score}_{\text{mdg}}$. We consider two sequences $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_m)$. The set $S_{(i,j)}$ will denote the states that corresponds to the image of the efficient alignments for subsequences $A' = (a_1, a_2, \ldots, a_i)$ and $B' = (b_1, b_2, \ldots, b_j)$, for $i \leq n$ and $j \leq m$; we will say that $S_{(i,j)}$ contains nondominated states. As in the Needleman-Wunsch algorithm, this algorithm starts by filling the first row and column of an $(n+1)$-by-$(m+1)$ matrix with the basis cases and it proceeds by filling it row-wise according to the recurrence equations. The nondominated states are given in the $(n+1, m+1)$ cell.

3.1 Recurrence equations for Score_{md}

For score function Score_{md}, each state s in $S_{(i,j)}$ is represented as a pair of values as follows

$$s = (m = \text{m}(\varphi), d = -\text{d}(\varphi))$$

for the corresponding alignment φ. The recurrence equations to compute $S_{(i,j)}$ for score function Score_{md} are as follows

$$S_{(i,j)} = \max_{(m,d)} \{T_A \cup T_B \cup T_0\} \tag{1}$$

where

$$\begin{aligned} T_A &= \left\{(m, d-1) \mid \text{ for all } (m,d) \in S_{(i-1,j)}\right\}, \\ T_B &= \left\{(m, d-1) \mid \text{ for all } (m,d) \in S_{(i,j-1)}\right\}, \\ T_0 &= \left\{\left(m + u_{(i,j)}, d\right) \mid \text{ for all } (m,d) \in S_{(i-1,j-1)}\right\} \end{aligned}$$

and

$$u_{(i,j)} = \begin{cases} 1 & \text{if } a_i = b_j \\ 0 & \text{otherwise} \end{cases}$$

with the following bases cases

$$\begin{aligned} S_{(0,0)} &= \{(0,0)\}, \\ S_{(i,0)} &= \{(0,-i)\}, \quad \text{for } 0 < i \leq n, \\ S_{(0,j)} &= \{(0,-j)\}, \quad \text{for } 0 < j \leq m. \end{aligned}$$

Note that sets T_A and T_B contain the states that are obtained by incrementing the number of indels to every nondominated state in $S_{(i-1,j)}$ and $S_{(i,j-1)}$, respectively. This corresponds to add an indel and a symbol in the same position of a partial alignment. Set T_0 contains the states that are obtained by a conditional increment on the number of matches to every nondominated state in $S_{(i-1,j-1)}$. In this case, this corresponds to the operation of adding both symbols from the two sequences in the same position of the partial alignment. Finally, note that the maximum operator in Eq. (1) is the componentwise maximum over the two components. This operation can be performed by the algorithm of Kung et al. [12]. Given that there can be $O(m+n)$ scores at each cell, the algorithm has $O(n \cdot m \cdot (n+m) \log(n+m))$ time complexity [16].

3.2 Recurrence equations for Score_{mg}

For the score function Score_{mg} we need to keep track of the last indel at the predecessor of each state; for a given state s, we will denote its predecessor by $\text{pred}(s)$. Therefore, we need a third component to each state. We define a state s in $S_{(i,j)}$ as follows

$$s = (m = \text{m}(\varphi), g = -\text{g}(\varphi), v_s)$$

where v_s indicates whether pred(s) is in $S_{(i-1,j)}$, $S_{(i,j-1)}$ or $S_{(i-1,j-1)}$. Then, we have the following recursive equation

$$S_{(i,j)} = \max_{(m,g)} \{T_A \cup T_B \cup T_0\} \tag{2}$$

where

$$\begin{aligned}
T_A &= \left\{ s = (m, g - w, v_s) \;\middle|\; \left(m, g, v_{\text{pred}(s)}\right) \in S_{(i-1,j)} \right\}, \\
T_B &= \left\{ s = (m, g - w, v_s) \;\middle|\; \left(m, g, v_{\text{pred}(s)}\right) \in S_{(i,j-1)} \right\}, \\
T_0 &= \left\{ s = \left(m + u_{(i,j)}, g, v_s\right) \;\middle|\; \left(m, g, v_{\text{pred}(s)}\right) \in S_{(i-1,j-1)} \right\},
\end{aligned}$$

where

$$w = \begin{cases} 1 & \text{if } v_s \neq v_{\text{pred}(s)} \\ 0 & \text{otherwise} \end{cases}$$

with the following bases cases

$$\begin{aligned}
S_{(0,0)} &= \{(0, 0, \phi)\}, \\
S_{(i,0)} &= \{(0, -1, \beta)\}, \quad \text{for } 0 < i \leq n, \\
S_{(0,j)} &= \{(0, -1, \alpha)\}, \quad \text{for } 0 < j \leq m,
\end{aligned}$$

where ϕ, α, and β are arbitrary distinct symbols, which indicate that the state is predecessor of a state in $S_{(i-1,j-1)}$, $S_{(i,j-1)}$ and $S_{(i-1,j)}$, respectively. The componentwise maximum in Eq. (2) is only defined over the two first components. We need to keep distinct states with respect to the number of matches, gaps, and v values; two states with the same number of matches and gaps and different v value may be predecessors of two distinct nondominated states. However, in the iteration $(n + 1, m + 1)$ we only need to keep distinct states with respect to the two first components. The algorithm has the same complexity of the algorithm for score function Score_{md}.

3.3 Recurrence equations for $\text{Score}_{\text{mdg}}$

The three-criteria case simply consists of extending the two cases presented above. Now, we define a state s in $S_{(i,j)}$ as follows

$$s = (m = \text{m}(\varphi), d = -\text{d}(\varphi), g = -\text{g}(\varphi), v_s)$$

and we have the following recursive equation

$$S_{(i,j)} = \max_{(m,d,g)} \{T_A \cup T_B \cup T_0\} \tag{3}$$

where

$$\begin{aligned}
T_A &= \left\{ s = (m, d - 1, g - w, v_s) \;\middle|\; \left(m, d, g, v_{\text{pred}(s)}\right) \in S_{(i-1,j)} \right\}, \\
T_B &= \left\{ s = (m, d - 1, g - w, v_s) \;\middle|\; \left(m, d, g, v_{\text{pred}(s)}\right) \in S_{(i,j-1)} \right\}, \\
T_0 &= \left\{ s = \left(m + u_{(i,j)}, d, g, v_s\right) \;\middle|\; \left(m, d, g, v_{\text{pred}(s)}\right) \in S_{(i-1,j-1)} \right\},
\end{aligned}$$

with the following bases cases

$$\begin{aligned}
S_{(0,0)} &= \{(0,0,0,\phi)\}, \\
S_{(i,0)} &= \{(0,-i,-1,\beta)\}, \quad \text{for } 0 < i \leq n, \\
S_{(0,j)} &= \{(0,-j,-1,\alpha)\}, \quad \text{for } 0 < j \leq m,
\end{aligned}$$

Also, the componentwise maximum in Eq. (3) is only defined over the first three components for the same reasons presented for the score function Score_{mg}. This algorithm has $O\left(n \cdot m \cdot (n+m)^2 \log(n+m)\right)$, if the algorithm of Kung et al. [12] is used to find the nondominated scores at each cell.

Table 1 shows an example of the several iterations of the multicriteria dynamic programming algorithm for $\text{Score}_{\text{mdg}}$ for the sequence alignment of two sequences, `AGGA` and `TAA`. The states in boldface correspond to states in set $S_{(i,j)}$ at each iteration (i,j). The superscripts in the states distinguish the two paths in the matrix that generate the final nondominated states in the bottom-rightmost entry. For the example above, the two efficient alignments that correspond to the states $(1,-1,-1)$ and $(2,-3,-2)$ are

$$\begin{pmatrix}\texttt{AGGA}\\ \texttt{T-AA}\end{pmatrix} \quad \text{and} \quad \begin{pmatrix}\texttt{-AGGA}\\ \texttt{TA--A}\end{pmatrix}$$

respectively; the symbol "-" corresponds to an indel.

4 Experiments with random sequences

We investigate the size of the nondominated set of scores for several sequences and alphabet sizes, as well as the CPU-time taken by our implementation of the multicriteria dynamic programming. For the bicriteria case, that is, for the score functions Score_{md} and Score_{mg}, we considered sequences from size 200 to 1000 with interval of 200. For score function $\text{Score}_{\text{mdg}}$, we consider sizes from 100 to 500 with interval of 100; preliminary experiments indicated that sizes above 500 would increase the CPU-time considerably (above 300 secs.). For each size, we generated 30 pairs of randomly generated sequences with alphabet size 2, 4, 10, 20, 30 and 40 which amounts to 900 pairs of sequences for each of the two bicriteria scores functions, plus 1080 for score function $\text{Score}_{\text{mdg}}$. Our implementation was coded in Java and it is minimally optimized. We used the sweep-line algorithm from Kung et al. [12] at each iteration to remove dominated states in loglinear time. The implementation was ran under Ubuntu 8.10 on an computer with 2GB DDR2 and with an 1.86 GHz Intel Core2Duo processor.

4.1 Number of nondominated scores

The top plots of Figure 2 present the average number of nondominated scores in random sequences for score functions Score_{md} (top-left plot) and Score_{mg} (top-right plot) for several alphabet sizes. Two observations are noteworthly: First, the number of nondominated scores is roughly a fixed fraction of the sequence size; secondly, the alphabet size has little effect on the number of nondominated scores, which also holds for parametric sequence alignment [5].

			A	G	G	A
		0	1	2	3	4
	0	**(0, 0, 0, ϕ)**^{1}	**(0, -1, -1, β)**	**(0, -2, -1, β)**	**(0, -3, -1,β)**	**(0, -4, -1, β)**
T	1	**(0, -1, -1, α)**^{2}	(0, -2, -2, α) **(0, 0, 0, ϕ)**^{1} (0, -2, -2, β)	(0, -3, -2, α) **(0, -1, -1, ϕ)** **(0, -1, -1, β)**^{1}	(0, -4, -2, α) **(0, -2, -1, ϕ)** (0, -2, -2, β) **(0, -2, -1, β)**	(0, -5, -2, α) **(0, -3, -1, ϕ)** (0, -3, -2, β) **(0, -3, -1, β)**
A	2	**(0, -2, -1, α)**	(0, -1, -1, α) **(1, -1, -1, ϕ)**^{2} (0, -3, -2, β)	(0, -2, -2, α) (0, -2, -2, α) **(0, 0, 0, ϕ)** **(1, -2, -2, β)**^{2}	(0, -3, -2, α) (0, -3, -2, α) (0, -1, -1, ϕ) **(0, -1, -1, ϕ)**^{1} **(0, -1, -1, β)** **(1, -3, -2, β)**^{2}	(0, -4, -2,α) (0, -4, -2, α) (1, -2, -1, ϕ) **(1, -2, -1, ϕ)** (0, -2, -2, β) (0, -2, -1, β) (1, -4, -2, β)
A	3	**(0, -3, -1, α)**	(1, -2, -2, α) **(1, -2, -1, ϕ)** (0, -4, -2, β)	(0, -1, -1, α) (1, -3, -3, α) **(1, -1, -1, ϕ)** (1, -3, -2, β)	(0, -2, -2, α) (0, -2, -2, α) (1, -4, -3, α) **(0, 0, 0, ϕ)** **(1, -2, -2, ϕ)** **(1, -2, -2, β)**	(1, -3, -2, α) **(1, -1, -1, ϕ)**^{1} (1, -1, -1, ϕ) **(2, -3, -2, ϕ)**^{2} (0, -1, -1, β) (1, -3, -3, β) (1, -3, -2, β)

Table 1: An example of the several iterations of the dynamic programming algorithm for the score function $\text{Score}_{\text{mdg}}$.

Interestingly, the data suggests a peak on the size of the nondominated set for sequences with alphabet size between 10 and 20.

The bottom plot of Figure 2 shows the average number of nondominated scores for score function $\text{Score}_{\text{mdg}}$. Clearly, the size of the nondominated set is larger than for the bicriteria case, and its relationship with the sizes of the sequences is nonlinear. We performed a regression analysis for the number of nondominated scores in relation with the size of the sequences for alphabet of size 20. The best fit was obtained by a square root transformation in the response variable by following a Box-Cox procedure, which matched our expectations with respect to the upper bound on the size of the nondominated set (see Section 2).

4.2 Computation time

The top plots of Figure 3 present the average CPU-time taken by our implementation for both bicriteria score functions. We performed a regression analysis for the results obtained with alphabet of size 20 for each score function. A Box-Cox procedure indicated that the best fit would be obtained by a cubic root transformation in the response variable, which matches the upper bound time complexity of our algorithm (see Section 3.1 and 3.2).

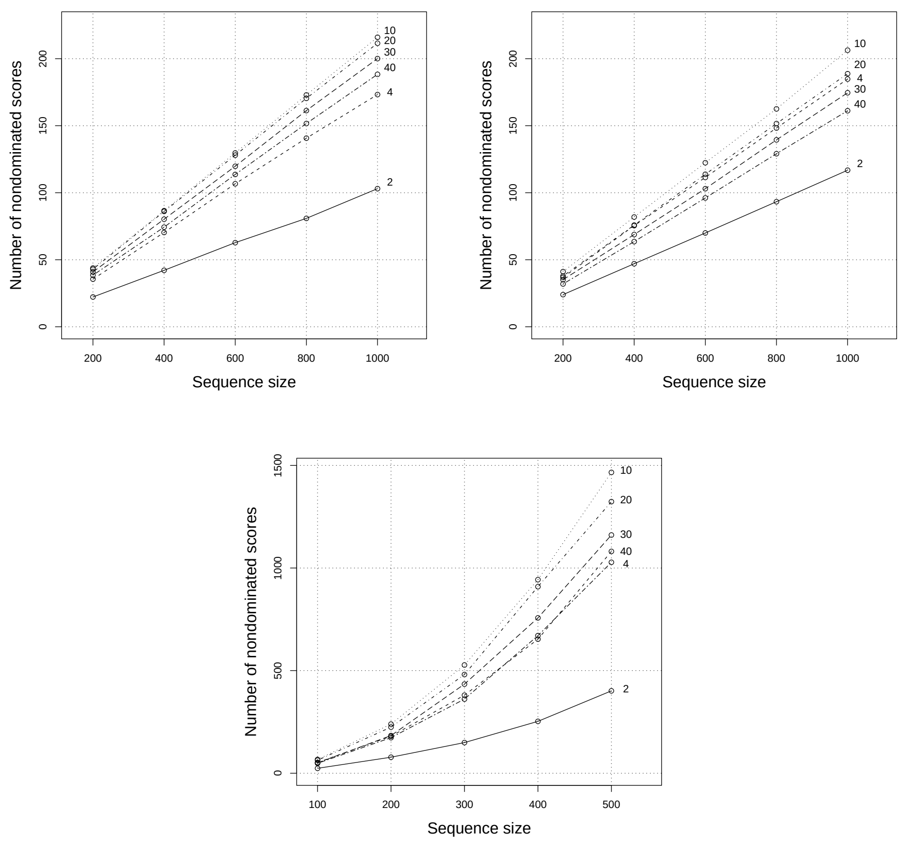

Figure 2: Number of nondominated scores for random instances for Score_{md} (top-left), Score_{mg} (top-right) and $\text{Score}_{\text{mdg}}$ (bottom).

We remark that our implementation for the score function Score_{mg} takes more time than for the score function Score_{md}. In fact, we observed that the number of partial non-dominated alignments at each iteration is larger for score function Score_{mg}. Note that the dynamic programming algorithm has to maintain distinct states with respect to three state components at each iteration for the score function Score_{mg}, whereas for the score function Score_{md} only two state components are considered.

The bottom plot shows the average CPU-time taken for the score function $\text{Score}_{\text{mdg}}$. Given the large values of CPU-time as the instance size grows, we plot the axis for CPU-time in logarithm scale. Clearly, the plot shows that the relationship between CPU-time and sequence size can be expressed by a polynomial function. The Box-Cox procedure suggested that the best fit for an alphabet of size 20 is obtained with a 4-th root transformation on CPU-time, which also matches the time complexity given in Section 3.3.

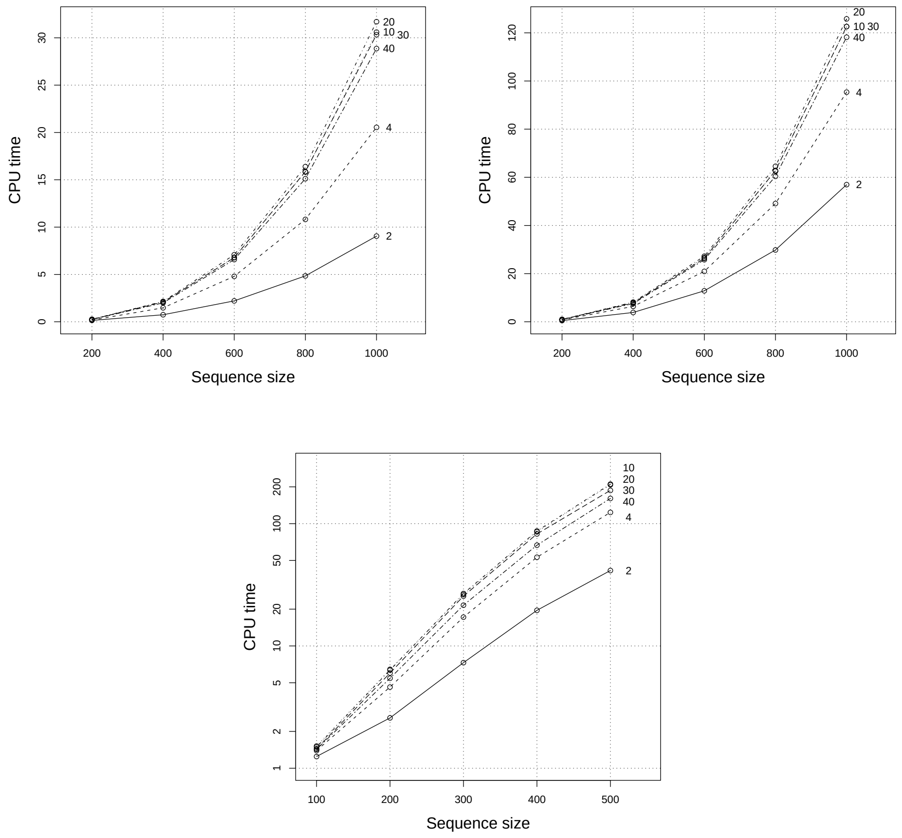

Figure 3: CPU-time of the multicriteria dynamic programming algorithm for random instances for Score$_{\text{md}}$ (top-left), Score$_{\text{mg}}$ (top-right) and Score$_{\text{mdg}}$ (bottom).

5 Experiments with real data

We performed another in-depth experimental analysis with protein sequences available in two benchmark data. The first is from the PREFAB version 4.0 that has been used for testing algorithms for multiple sequence alignment [2]. The second benchmark is from SABmark version 1.65 [10]. We choose 50 pairs of the largest sequences from each dataset with sequence sizes ranging from 108 to 1132 symbols for PREFAB and from 110 to 721 for SABmark benchmark. The sequences chosen from each benchmark were paired randomly.

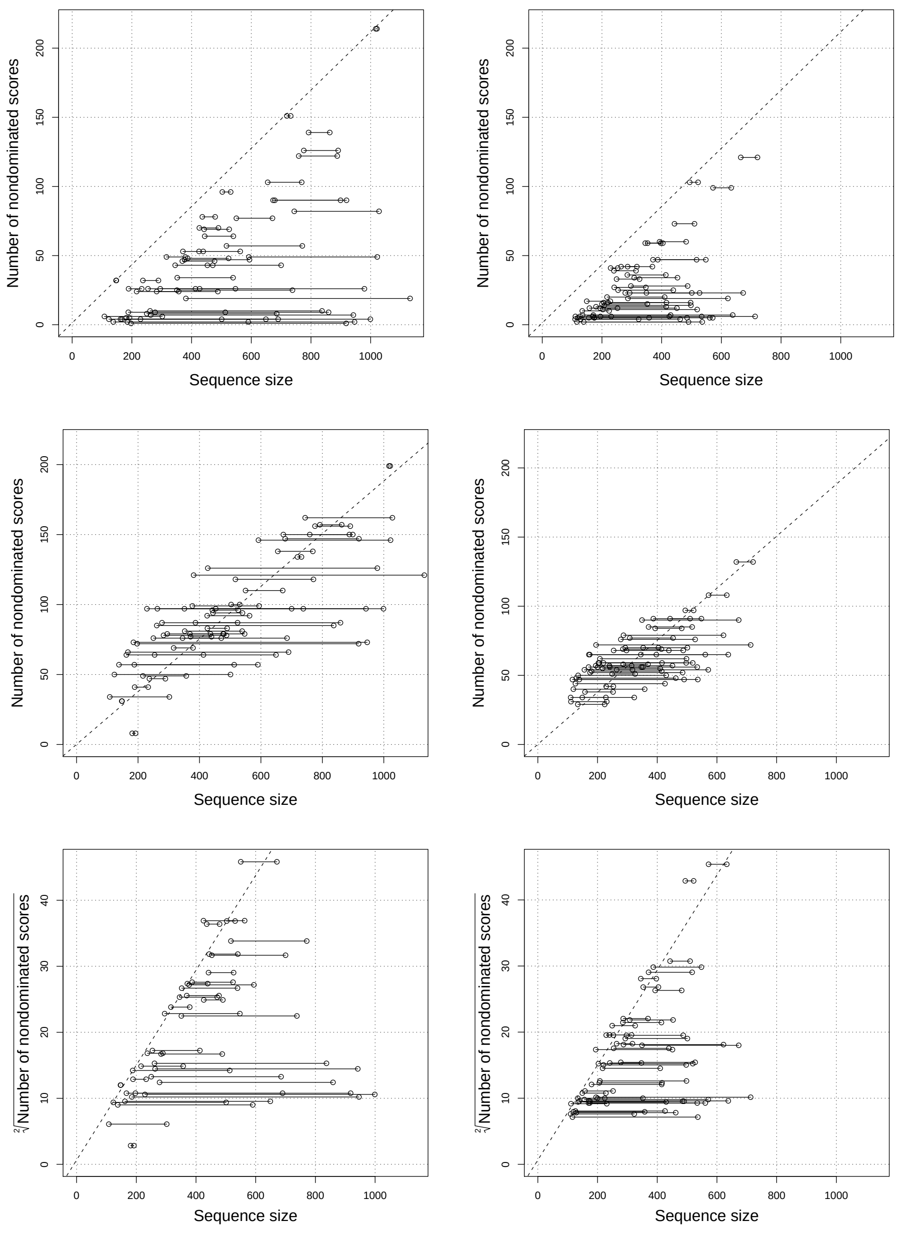

Figure 4: Number of nondominated scores for Score_{md} (top), Score_{mg} (center) and $\text{Score}_{\text{mdg}}$ (bottom) in PREFAB and SABmark (right) benchmarks.

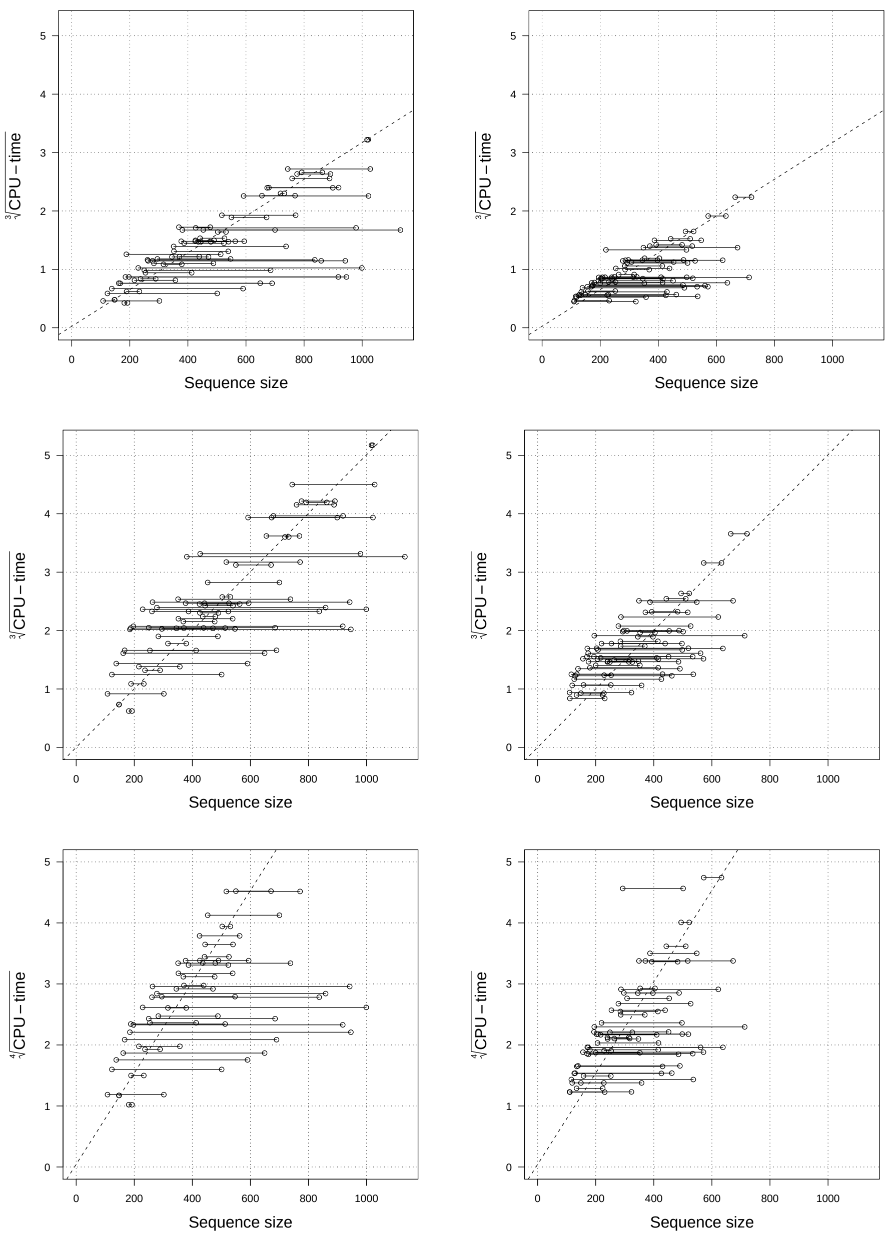

Figure 5: CPU-time of the dynamic programming algorithm for Score_{md} (top), Score_{mg} (center) and $\text{Score}_{\text{mdg}}$ (bottom) in PREFAB (left) and SABmark (right) benchmarks.

5.1 Number of nondominated scores

The plots in the top, center and bottom of Figure 4 show the number of nondominated scores according to score functions Score_{md}, Score_{mg} and $\text{Score}_{\text{mdg}}$, respectively. Each pair of points connected by a line indicates the sizes of the two sequences that were tested pairwise. The dashed line corresponds to the regression line obtained for pairs of random sequences with alphabet size of 20.

The plots for score function Score_{md} in both benchmarks (top plots) indicate that the set of nondominated scores for most pairs of sequences is smaller than for random sequences with similar sequence size. Furthermore, the plots for score function Score_{mg} (center plots) show that the number of nondominated scores follows the same growth observed in random sequences. Also, we computed the convex hull for set of nondominated scores, which would correspond to the output of parametric sequence alignment. We observed that, in average, only 30% of the efficient set is also optimal for the latter.

For the score function $\text{Score}_{\text{mdg}}$, 12 pairs of sequences from PREFAB benchmark and two pairs from SABmark benchmark were not considered since the algorithm took more time than the time limit of 300 seconds. The bottom plots of Figure 4 show the number of nondominated scores for both benchmarks. We present the square root of the number of nondominated scores in order to match the transformation suggested in Section 4.1. The dashed line corresponds to the regression line obtained for pairs of random sequences with alphabet size of 20. In both benchmarks, the results indicates a relationship between the size of the smaller sequence of each pair and the number of nondominated scores.

5.2 Computation time

The top, center and bottom plots of Figure 5 show the CPU-time taken by our implementation on the score functions Score_{md}, Score_{mg} and $\text{Score}_{\text{mdg}}$, respectively. We plot the sequence size against the cubic-root of CPU-time for the bicriteria case and against the 4-th root of CPU time for the three-criteria case, as well as the regression line suggested by the regression model for random sequences (see Section 4.2). The results indicate that the same order of magnitude holds for the sequences of both benchmark datasets.

6 Discussion and conclusions

Multicriteria sequence alignment allows the possibility of exploring a tractable set of optimal alignments that are not reachable by any other method. Moreover, it allows the biologist to get rid of coefficients in the score functions. However, it also brings more computational cost. This article explored whether this computational cost is worth being payed. These preliminary results indicate that this approach is at least feasible for small to medium sequence sizes. We believe that further improvements in computation time can be obtained by exploring more conditions that allow the removal of more states at each iteration.

This work can be extended to other scenarios. For instance, for local pairwise sequence alignment, that is, to find similar regions among two sequences, it is possible to extend Smith-Waterman algorithm [17] for the multicriteria case as done in this article, as also shown in [16]. In this case, the number of mismatches can be defined as a third criteria to minimize.

In case of more than two sequences, the current approach can be easily extended for the *Sum-of-Pairs* score function, where the score component of each pair of sequences is added to form the overall score. However, for an arbitrary number $k > 2$ of sequences, the problem becomes NP-hard and the problem may not be suitable for a multicriteria dynamic programming approach.

Acknowledgements

The authors thank Dr. Francisco Pereira for the helpful suggestions to improve this article. J.P.O. Almeida is supported by a grant from the Fundação de Ciência e Tecnologia.

References

[1] C. Bazgan, H. Hugot, and D. Vanderpooten. Solving efficiently the 0-1 multi-objective knapsack problem. *Computers and Operations Research*, 36(1):260–279, 2009.

[2] R. C. Edgar. MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.

[3] M. Ehrgott. Hard to say it's easy - Four reasons why combinatorial multiobjective programmes are hard. In Y. Y. Haimes and R. E. Steuer, editors, *Research and Practice in Multiple Criteria Decision Making*, volume 487 of *Lecture Notes in Economics and Mathematical Systems*, pages 69–81. Springer, Berlin, Germany, 2000.

[4] M. Ehrgott. *Multicriteria Optimization*, volume 491 of *Lecture Notes in Economics and Mathematical Systems*. Springer, Heidelberg, Germany, 2000.

[5] D. Fernández-Baca, T. Seppäläinen, and G. Slitzki. Bounds for parametric sequence alignment. *Discrete Applied Mathematics*, 118(3):181–198, 2002.

[6] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.

[7] D. Gusfield, K. Balasubramanian, and D. Naor. Parametric optimization of sequence alignment. In *Proceedings of the third annual ACM-SIAM symposium on discrete algorithms*, pages 432–439, 1992.

[8] J. Handl, D. B. Kell, and J. Knowles. Multiobjective optimization in bioinformatics and computational biology. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):279–292, 2007.

[9] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. In *Proceedings of the National Academy of Sciences of the USA*, volume 89, pages 10915–10909, 1992.

[10] L. Wyns I.V. Walle, I. Lasters. Sabmark: A benchmark for sequence alignment that covers the entire fold space. *Bioinformatics*, 21:1267–1267, 2005.

[11] K. Klamroth and M. Wiecek. Dynamic programming approaches to the multiple criteria knapsack problem. *Naval Research Logistics*, 45:57–76, 2000.

[12] H. Kung, F. Luccio, and F. Preparata. On finding the maxima of a set of vectors. *Journal of the ACM*, 22(4):469–476, 1975.

[13] K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12 of *Kluwer's International Series in Operations Research & Management Science.* Kluwer Academic Publishers, 1999.

[14] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[15] G. L. Nemhauser and Z. Ullman. Discrete dynamic programming and capital allocation. *Management Science*, 15(9):494–505, 1969.

[16] M. A. Roytberg, M. N. Semionenkov, and O. Yu. Tabulina. Pareto-optimal alignment of biological sequences. *Biophysics*, 44(4):581–594, 1999.

[17] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[18] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application.* Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1986.

```
 Sequence P23297: MPSQMEHAMETMMFTFHKFAGDKGYLTKEDLRVLMEKEFPGFLENQKDPLAVDKIMKDLDQCRDGKVGFQSFFSLIAGLTIACNDYFVVHMKQKGKK
 Sequence P60903: MGSELETAMETLINVFHAHSGKEGDKYKLSKKELKELLQTELSGFLDAQKDVDAVDKVMKELDENGDGEVDFQEYVVLVAALTVACNNFFWENS

Matches: 21  Indels:3
MPSQMEHAMETMMFTFHKFAGDKGYLTKEDLRVLMEKEFPGFLENQKDPLAVDKIMKDLDQCRDGKVGFQSFFSLIAGLTIACNDYFVVHMKQKGKK
| |  | ||||    ||   |  |   |     | |               ||   |          |   |      |  |
MGSELETAMETLINVFHAHSGKEGDKYKLSKKELKELLQTELSGFLDAQKDVDAVDKVMKELDENGDGEVDFQEYVV-LVAALTVACNNFFWENS--

Matches: 23  Indels:5
MPSQMEHAMETMMFTFHKFAGDKGYLTKEDLRVLMEKEFPGFLENQKDPLA-VDKIMKDLDQCRDGKVGFQSFFSLIAGLTIACNDYFVVHMKQKGKK
| |  | ||||    ||   |  |   |     | |      |    |    ||   |          |   |      |  |
MGSELETAMETLINVFHAHSGKEGDKYKLSKKELKELLQTE-LSGFLDAQKDVDAVDKVMKELDENGDGEVDFQEYVV-LVAALTVACNNFFWENS--

Matches: 24  Indels:7
MPSQMEHAMETMMFTFHKFAGDKGYLTKEDLRVLMEKEFPGFLENQKDPLA-VDKIMKDLDQCRDGKVGFQSFFSLIAGLTIACNDYFVVHMKQKGKK-
| |  | ||||    ||   |  |   |     | |      |    |    ||   |          |   |      |  |     |
MGSELETAMETLINVFHAHSGKEGDKYKLSKKELKELLQTE-LSGFLDAQKDVDAVDKVMKELDENGDGEVDFQEYVV-LVAALT---VACNNFFWENS

Matches: 44  Indels:9
MPSQMEHAMETMMFTFHKFAGDKG--Y-LTKEDLRVLMEKEFPGFLENQKDPLAVDKIMKDLDQCRDGKVGFQSFFSLIAGLTIACNDYFVVHMKQKGKK
| |  | ||||    ||   |  |  | | |  |  |   |  |||  |||  |||| || ||   || | ||    | | || |||  |
MGSELETAMETLINVFHAHSGKEGDKYKLSKKELKELLQTELSGFLDAQKDVDAVDKVMKELDENGDGEVDFQEYVVLVAALTVACNNFFWENS------

Matches: 45  Indels:11
MPSQMEHAMETMMFTFHKFAGDKG--Y-LTK-EDLRVLMEKEFPGFLENQKDPLAVDKIMKDLDQCRDGKVGFQSFFSLIAGLTIACNDYFVVHMKQKGKK
| |  | ||||    ||   |  |  | | | | |  |   |  |||  |||  |||| || ||   || | ||    | | || |||  |
MGSELETAMETLINVFHAHSGKEGDKYKLSKKE-LKELLQTELSGFLDAQKDVDAVDKVMKELDENGDGEVDFQEYVVLVAALTVACNNFFWENS------

Matches: 46  Indels:13
MPSQMEHAMETMMFTFHKFAGDK-G--Y-LTK-EDLRVLMEKEFPGFLENQKDPLAVDKIMKDLDQCRDGKVGFQSFFSLIAGLTIACNDYFVVHMKQKGKK
| |  | ||||    ||   | | |  | | | | |  |   |  |||  |||  |||| || ||   || | ||    | | || |||  |
MGSELETAMETLINVFHAHSG-KEGDKYKLSKKE-LKELLQTELSGFLDAQKDVDAVDKVMKELDENGDGEVDFQEYVVLVAALTVACNNFFWENS------

Matches: 47  Indels:15
MPSQMEHAMETMMFTFH----KFAGDKGY-LTK-EDLRVLMEKEFPGFLENQKDPLAVDKIMKDLDQCRDGKVGFQSFFSLIAGLTIACNDYFVVHMKQKGKK
| |  | ||||    ||    |  ||| | | | | |  |   |  |||  |||  |||| || ||   || | ||    | | || |||  |
MGSELETAMETLINVFHAHSGKE-GDK-YKLSKKE-LKELLQTELSGFLDAQKDVDAVDKVMKELDENGDGEVDFQEYVVLVAALTVACNNFFWENS------
```

Figure 6: Efficient alignments for two sequences of human S100 calcium-binding proteins.

```
 Sequence Q96FQ6: MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHMLSDTGNRKAADKLIQNLDANHDGRISFDEYWTLIGGITGPIAKLIHEQEQQSSS
 Sequence P29034: MMCSSLEQALAVLVTTFHKYSCQEGDKFKLSKGEMKELLHKELPSFVGEKVDEEGLKKLMGSLDENSDQQVDFQEYAVFLALITVMCNDFFQGCPDRP

Matches: 33  Indels:5
MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHMLSDTGNRKAADKLIQNLDANHDGRISFDEYWTLIGGITGPIAKLIHEQEQQSSS
|  |   || |  |||  | || |     | | ||    | | |||              ||   || | |    | ||      ||         |
MM-CSS-LEQALAVLVTTFHKY-SCQEGDKFKLSKGEMKELLHKELPSFVGEKVDEEGLKKLMGSLDENSDQQVDFQEYAVFLALITVMCNDFF--QGCPDRP

Matches: 34  Indels:7
MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHMLSDTGNRKAADKLIQNLDANHDGRISFDEYWT-LIGGITGPIAKLIHEQEQQSSS
|  |   || |  |||  | || |     | | ||    | | |||              ||   || | |    | ||   |   ||         |
MM-CSS-LEQALAVLVTTFHKY-SCQEGDKFKLSKGEMKELLHKELPSFVGEKVDEEGLKKLMGSLDENSDQQVDFQEYAVFLAL-ITVMCNDFF--QGCPDRP

Matches: 35  Indels:11
MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHMLSDTGNRKAAD--K-LIQNLDANHDGRISFDEYWTLIGGITGPIAKLIHEQEQQSSS
|  |   || |  |||  | || |     | | ||    | | |||    |  |        | |   || | |    | ||      ||         |
MM-CSS-LEQALAVLVTTFHKY-SCQEGDKFKLSKGEMKELLHKELP---SFVGEKVDEEGLKKLMGSLDENSDQQVDFQEYAVFLALITVMCNDFF--QGCPDRP

Matches: 36  Indels:13
MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHMLSDTGNRKAAD--K-LIQNLDANHDGRISFDEYWT-LIGGITGPIAKLIHEQEQQSSS
|  |   || |  |||  | || |     | | ||    | | |||    |  |        | |   || | |    | ||   |   ||         |
MM-CSS-LEQALAVLVTTFHKY-SCQEGDKFKLSKGEMKELLHKELP---SFVGEKVDEEGLKKLMGSLDENSDQQVDFQEYAVFLAL-ITVMCNDFF--QGCPDRP

Matches: 37  Indels:15
MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHMLSDTGNRKAAD---K-LIQNLDANHDGRISFDEYWT-LIGGITGPIAKLIHEQEQQSSS
|  |   || |  |||  | || |     | | ||    | | |||    |  |  |      | |   || | |    | ||   |   ||         |
MM-CSS-LEQALAVLVTTFHKY-SCQEGDKFKLSKGEMKELLHKELP---SFVGE-KVDEEGLKKLMGSLDENSDQQVDFQEYAVFLAL-ITVMCNDFF--QGCPDRP

Matches: 38  Indels:17
MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHMLSDTGNRKAAD----K-LIQNLDANHDGRISFDEYWT-LIGGITGPIAKLIHEQEQQSSS
|  |   || |  |||  | || |     | | ||    | | |||    |  |  |  |    | |   || | |    | ||   |   ||         |
MM-CSS-LEQALAVLVTTFHKY-SCQEGDKFKLSKGEMKELLHKELP---SFVGE-KV-DEEGLKKLMGSLDENSDQQVDFQEYAVFLAL-ITVMCNDFF--QGCPDRP

Matches: 39  Indels:19
MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHM-L-SDTGNRKAAD---K-LIQNLDANHDGRISFDEYWT-LIGGITGPIAKLIHEQEQQSSS
|  |   || |  |||  | || |     | | ||    ||  ||| |  | |  |  |      | |   || | |    | ||   |   ||         |
MM-CSS-LEQALAVLVTTFHKY-SCQEGDKFKLSKG---EM--KELLHKELPSFVGE-KVDEEGLKKLMGSLDENSDQQVDFQEYAVFLAL-ITVMCNDFF--QGCPDRP

Matches: 40  Indels:21
MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHM-L-SDTGNRKAAD----K-LIQNLDANHDGRISFDEYWT-LIGGITGPIAKLIHEQEQQSSS
|  |   || |  |||  | || |     | | ||    ||  ||| |  | |  |  |  |    | |   || | |    | ||   |   ||         |
MM-CSS-LEQALAVLVTTFHKY-SCQEGDKFKLSKG---EM--KELLHKELPSFVGE-KV-DEEGLKKLMGSLDENSDQQVDFQEYAVFLAL-ITVMCNDFF--QGCPDRP

Matches: 41  Indels:33
MSDCYTELEKAVIVLVENFYKYVSKYSLVKNKISKSSFREMLQKELNHM-L-SDTGNRKAAD----K-LIQNLDANHDGRISFDEYWT-LIGGITGPIAKLIHEQEQ---QSSS---
|  |   || |  |||  | || |     | | ||    ||  ||| |  | |  |  |  |    | |   || | |    | ||   |        | ||        |
MM-CSS-LEQALAVLVTTFHKY-SCQEGDKFKLSKG---EM--KELLHKELPSFVGE-KV-DEEGLKKLMGSLDENSDQQVDFQEYAVFL--------A-LITVMCNDFFQGCPDRP
```

Figure 7: Efficient alignments for two sequences of human S100 calcium-binding proteins.