BIOINFORMATICS ORIGINAL PAPER

Vol. 29 no. 8 2013, pages 996–1003
doi:10.1093/bioinformatics/btt098

Sequence analysis

Advance Access publication February 23, 2013

Improvements on bicriteria pairwise sequence alignment: algorithms and applications

Maryam Abbasi[1], Luís Paquete[1,*], Arnaud Liefooghe[2], Miguel Pinheiro[3] and Pedro Matias[1]

[1]CISUC, Department of Informatics Engineering, University of Coimbra, Pólo II, 3030-290 Coimbra, Portugal, [2]LIFL, UMR CNRS 8022, Bât. M3, Université Lille 1, 59655 Villeneuve d'Ascq, France and [3]Bioinformatics Unit, Biocant, 3060-197 Cantanhede, Portugal

Associate Editor: Martin Bishop

ABSTRACT

Motivation: In this article, we consider the bicriteria pairwise sequence alignment problem and propose extensions of dynamic programming algorithms for several problem variants with a novel pruning technique that efficiently reduces the number of states to be processed. Moreover, we present a method for the construction of phylogenetic trees based on this bicriteria framework. Two exemplary cases are discussed.

Results: Numerical results on a real dataset show that this approach is very fast in practice. The pruning technique saves up to 90% in memory usage and 80% in CPU time. Based on this method, phylogenetic trees are constructed from real-life data. In addition of providing complementary information, some of these trees match those obtained by the Maximum Likelihood method.

Availability and implementation: Source code is freely available for download at URL http://eden.dei.uc.pt/~paquete/MOSAL, implemented in C and supported on Linux, MAC OS and MS Windows.

Contact: paquete@dei.uc.pt

Received on October 15, 2012; revised on January 31, 2013; accepted on February 20, 2013

1 INTRODUCTION

Recently, there has been a growing interest on the multi-criteria formulation of optimization problems that arise in computational biology (see an extensive review in Handl *et al.*, 2007). However, exact solution approaches for most of these new formulations have not been thoroughly investigated, with the exception of the work of Roytberg *et al.* (1999) for bicriteria pairwise sequence alignment.

For an alignment φ of sequences $A := (a_1, \ldots, a_{n_1})$ and $B := (b_1, \ldots, b_{n_2})$, we denote by $s(\varphi)$ the *substitution score* of alignment φ according to a substitution matrix $\mathbf{M}$, and by $d(\varphi)$ and $g(\varphi)$, the number of indels and gaps (maximal consecutive run of indels) of φ. The two following score vector functions are considered:

$$\mathrm{VSD}(\varphi) := (s(\varphi), -d(\varphi))$$

$$\mathrm{VSG}(\varphi) := (s(\varphi), -g(\varphi))$$

*To whom correspondence should be addressed.

Two bicriteria problems that consist of finding the alignments that are 'maximal' with respect to the score vector functions above are as follows:

$$\arg \mathrm{vmax}\{\varphi : \mathrm{VSD}(\varphi), \varphi \in \Phi\} \qquad \text{(VSDP)}$$

$$\arg \mathrm{vmax}\{\varphi : \mathrm{VSG}(\varphi), \varphi \in \Phi\} \qquad \text{(VSGP)}$$

where Φ denotes the set of all feasible alignments. The image of set Φ in the score function space is called *feasible score set*. To give a proper meaning to the operator vmax, we introduce the following dominance relation between score vectors in Problem (VSDP): Given two alignments φ and φ', $\mathrm{VSD}(\varphi) > \mathrm{VSD}(\varphi')$ (φ *dominates* φ') if and only if it holds that $s(\varphi) \geq s(\varphi')$, $d(\varphi) \leq d(\varphi')$ and $\mathrm{VSD}(\varphi) \neq \mathrm{VSD}(\varphi')$. An alignment φ is *Pareto optimal* if there exists no other alignment φ^* such that $\mathrm{VSD}(\varphi^*) > \mathrm{VSD}(\varphi)$. The set of all Pareto optimal alignments is called *Pareto optimal alignment set*. The image of a Pareto optimal alignment in the score function space is a *non-dominated score*, and the set of all non-dominated scores is called *non-dominated score set*. The dominance relation and the above notation also apply to Problem (VSGP) with the necessary changes.

Computing the Pareto optimal alignment set can be an intractable task: consider the sequences $A := G^n$ and $B := T(GT)^{2n}$ and the substitution matrix $\mathbf{M}[i,i] := 1$ and $\mathbf{M}[i,j] := 0$, $i \neq j$; then, there exist $\binom{2n}{n}$ Pareto optimal alignments that match G^n in both sequences, as there exists no other alignment with larger substitution score and lesser number of indels ($3n+1$ indels). The aforementioned example also applies to the number of gaps ($n+1$ gaps). However, the size of the non-dominated score set is bounded by $\min(\ell, n_1 + n_2 - 2\ell)$, where ℓ is the size of longest common subsequence of A and B (Roytberg *et al.*, 1999).

Interesting properties of this formulation in relation to parametric sequence alignment (see Gusfield *et al.*, 1992) are discussed in Roytberg *et al.*, 1999: an optimal alignment for the parametric score function with positive parameters is a Pareto optimal alignment. However, there may exist Pareto optimal alignments that are not optimal for any parameter setting. These alignments, and corresponding scores, are called *supported* and *non-supported*, respectively (Ehrgott, 2005). From a geometrical point of view, supported scores are those that lie in the convex-hull boundary of the non-dominated score set.

Furthermore, it is common to distinguish between *extreme* and *non-extreme* supported scores (Ehrgott, 2005). A score is extreme supported if it is a vertex of the convex hull boundary of the feasible score set; otherwise it is a non-extreme supported score. Although it is relatively easy to find all extreme supported scores (Gusfield *et al.*, 1992), it is more challenging to find all those that are non-extreme. Finding the whole non-dominated score set would allow identifying both extreme and non-extreme supported non-dominated scores, as well as non-supported non-dominated scores.

Therefore, multicriteria sequence alignment brings advantages to the practitioner, as it allows to get rid of parameters and to explore a tractable set of alignments that are not reachable by any other methods. However, to the best of our knowledge, few work has been done on multi-criteria sequence alignment (Paquete and Almeida, 2009; Roytberg *et al.*, 1999; Taneda, 2010). In this article, we propose extensions of dynamic programming algorithms for several bicriteria problem variants with a pruning technique that is based on the comparison of lower and upper bounds, as performed in branch-and-bound procedures. Moreover, we describe an experimental analysis on a real dataset and discuss the use of bicriteria pairwise sequence alignment in the context of phylogenetic tree construction.

2 ALGORITHMS

In the following sections, we describe dynamic programming techniques for solving several formulations of the bicriteria pairwise sequence alignment as well as a novel pruning technique that reduces the number of states in practice.

2.1 Multicriteria dynamic programming

For the sake of clarity, we introduce the dynamic programming algorithm for Problem (VSDP) as proposed in Roytberg *et al.* (1999) but with a different formulation. For a given alignment φ, we define a state $p := \mathrm{VSD}(\varphi)$. To compute the non-dominated scores, a matrix $\mathbf{P}$ is constructed where each entry $\mathbf{P}[i,j]$, for $(i,j) \in \{1,\ldots,n_1\} \times \{1,\ldots,n_2\}$, will store the set of states corresponding to the Pareto optimal alignments of subsequences $(a_1,\ldots,a_i)$ and $(b_1,\ldots,b_j)$. The recurrence for $\mathbf{P}[i,j]$ is as follows:

$$\mathbf{P}[i,j] := \mathrm{vmax}\begin{cases} \{p + (\mu(i,j),0) : p \in \mathbf{P}[i-1,j-1]\} \\ \{p + (0,-1) : p \in \mathbf{P}[i-1,j]\} \\ \{p + (0,-1) : p \in \mathbf{P}[i,j-1]\} \end{cases}$$

where $\mu(i,j)$ is the substitution score for (a_i,b_j). The bases cases are $\mathbf{P}[0,0] := \{(0,0)\}$, $\mathbf{P}[i,0] := \{(0,-i)\}$, $\mathbf{P}[0,j] := \{(0,-j)\}$, for $1 \le i \le n_1$ and $1 \le j \le n_2$. Operator vmax keeps only the non-dominated states at entry $\mathbf{P}[i,j]$. Roytberg *et al.* (1999) suggested the use of a log-linear algorithm for this operation. However, this can be performed in linear time by extending the MERGE algorithm in Beier and Vöcking (2011) for three sorted lists of non-dominated scores. The overall time and space-complexity of the algorithm above is $O(n_1 \cdot n_2 \cdot (n_1+n_2))$.

The dynamic programming algorithm for Problem (VSGP) is briefly discussed in Roytberg *et al.* (1999). We give a more detailed explanation of this approach, which extends the algorithm in Gusfield (1997, pp. 244). For a given alignment $\varphi := (A', B')$, we define a state $q := \mathrm{VSG}(\varphi)$. For computing the set of non-dominated scores, we keep four dynamic programming matrices: $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{S}$ and $\mathbf{T}$. For a given $(i,j) \in \{1,\ldots,n_1\} \times \{1,\ldots,n_2\}$, entry $\mathbf{R}[i,j]$, $\mathbf{S}[i,j]$ and $\mathbf{T}[i,j]$ will store the set of states corresponding to Pareto optimal alignments of subsequences $(a_1,\ldots,a_i)$ and $(b_1,\ldots,b_j)$ that end with (a_i,b_j), $('-',b_j)$ and $(a_i,'-')$, respectively, where $'-'$ is a gap character. The entry $\mathbf{Q}[i,j]$ will store the states corresponding to Pareto optimal alignments of subsequences $(a_1,\ldots,a_i)$ and $(b_1,\ldots,b_j)$. Then, the recursion is as follows:

$$\begin{aligned} \mathbf{Q}[i,j] &:= \mathrm{vmax}\begin{cases} \mathbf{R}[i,j] \\ \mathbf{S}[i,j] \\ \mathbf{T}[i,j] \end{cases} \\ \mathbf{R}[i,j] &:= \{q + (\mu(i,j),0) : q \in \mathbf{Q}[i-1,j-1]\} \\ \mathbf{S}[i,j] &:= \mathrm{vmax}\begin{cases} \mathbf{S}[i,j-1] \\ \{q + (0,-1) : q \in \mathbf{Q}[i,j-1]\} \end{cases} \\ \mathbf{T}[i,j] &:= \mathrm{vmax}\begin{cases} \mathbf{T}[i-1,j] \\ \{q + (0,-1) : q \in \mathbf{Q}[i-1,j]\} \end{cases} \end{aligned}$$

The base cases of the matrices are as follows: $\mathbf{Q}[0,0] := \{(0,0)\}$, $\mathbf{Q}[i,0] := \mathbf{S}[i,0] := \mathbf{Q}[0,j] := \mathbf{T}[0,j] := \{(0,-1)\}$, for $1 \le i \le n_1$ and $1 \le j \le n_2$. Operation vmax takes also linear amount of time by using the same technique as for Problem (VSDP). As the number of gaps is bounded from above by the number of indels in an alignment, the time and space-complexity is also $O(n_1 \cdot n_2 \cdot (n_1+n_2))$.

2.2 Bounds in problem (VSDP)

We describe a pruning technique for the dynamic programming algorithm for Problem (VSDP) that is able to reduce the number of states by comparing their upper bounds with a pre-computed lower bound set. Upper and lower bounds have also been used within scalarized score functions (Roytberg, 1992).

2.2.1 Lower bound set For the definition of lower bounds on the non-dominated score set for Problem (VSDP), we introduce the notions of lexicographic and scalarized score functions.

We say that a vector $x \in \mathbb{R}^2$ is lexicographically larger than or equal to a vector $y \in \mathbb{R}^2 (x >_{\mathrm{lex}} y)$ if $x_1 > y_1$ or if $x_1 = y_1$ and $x_2 \ge y_2$. In this article, we consider two problems that consist of finding an alignment that is lexicographic maximal (lexmax) according to a given order of priority on the optimization of the two score function components:

$$\arg\mathrm{lexmax}\{\varphi : (\mathrm{s}(\varphi), -\mathrm{d}(\varphi)), \varphi \in \Phi\} \qquad \text{(LexSDP)}$$

$$\arg\mathrm{lexmax}\{\varphi : (-\mathrm{d}(\varphi), \mathrm{s}(\varphi)), \varphi \in \Phi\} \qquad \text{(LexDSP)}$$

The order of the function components indicates the priority that is considered among the criteria.

Let φ_s and φ_d be the lexicographic maximal alignments for Problems (LexSDP) and (LexDSP), respectively. Let $\mathrm{MAX} := \mathrm{VSD}(\varphi_s)$ and $\mathrm{MIN} := \mathrm{VSD}(\varphi_d)$. By the definition of optimality for Problem (VSDP), it holds that MAX and MIN belong to the non-dominated score set (Ehrgott, 2005). Moreover, they indicate that there cannot exist a Pareto optimal

alignment with larger (smaller) substitution score value and more (less) indels than given by the components of MAX (MIN). Hence, the two score vectors give a bound on the possible ranges of the non-dominated score set. In fact, if MAX = MIN, then the non-dominated score set contains only a single element and no further computation is required.

Another lower bound is given by the solution to a *scalarized* version of the bicriteria alignment problem. We consider the weighted sum scalarization approach:

$$\mathrm{WSD}(\varphi) := w_s \cdot \mathrm{s}(\varphi) - w_d \cdot \mathrm{d}(\varphi)$$

where w_s and w_d are positive real weighting coefficients. The goal is to find the alignment that maximizes the scalarized score function as follows:

$$\arg\max\{\varphi : \mathrm{WSD}(\varphi), \varphi \in \Phi\} \qquad \text{(WSDP)}$$

Other scalarized functions are also possible (Ehrgott, 2005). In the particular case of the weighted sum function, the alignment that is optimal to Problem (WSDP) is also Pareto optimal to Problem (VSDP), although the opposite does not hold in general (Ehrgott, 2005). Let φ_w denote the optimal alignment for Problem (WSDP) for a given w_s and w_d and let MID := VSD(φ_w). It is also important to highlight that the scalarized problem can be solved several times for different coefficients to get a tighter lower bound set.

The score vectors MAX, MID and MIN allow to define a lower bound set on the non-dominated score set of Problem (VSDP). Let $\mathcal{R}$ denote the region

$$\mathcal{R} = \{r \in \mathbb{R} \times \mathbb{R}_0^- : b > r, b \in \{\mathrm{MAX}, \mathrm{MID}, \mathrm{MIN}\}\}.$$

Figure 1 illustrates the location of MAX, MID and MIN and definition of $\mathcal{R}$ (shaded area). There may exist further Pareto optimal alignments whose score vectors are located in the complement of $\mathcal{R}$. However, any alignment whose score vector is in the interior of $\mathcal{R}$ cannot be Pareto optimal, as it would be dominated by an alignment with a score vector equals to MIN, MID or MAX.

The computation of the three score vectors can be performed with the algorithm of Needleman and Wunsch (1970) by keeping the components separately in the dynamic programming matrix and choosing the state at each entry that maximizes the scalarized score function WSD, for the case of MID, or according to the lexicographic ordering for the case of MAX and MIN, respectively. Therefore, the three score vectors can be found in $O(n_1 \cdot n_2)$-time.

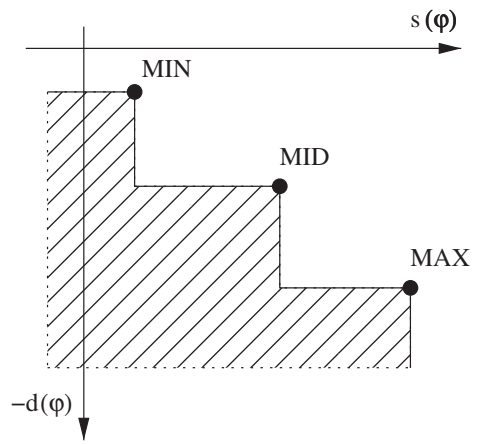

Fig. 1. Illustration of the lower bound set for problem (VSDP)

In the following, we only introduce the recurrence relation required for computing the lexicographic maximal alignment for Problem (LexSDP), which gives the score vector MAX. We consider a dynamic programming matrix **L**, where the entry $\mathbf{L}[i,j]$, for $(i,j) \in \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$, will store the state corresponding to the lexicographic maximal alignment of subsequences $(a_1, \ldots, a_i)$ and $(b_1, \ldots, b_j)$. The elements of matrix **L** are calculated recursively by

$$\mathbf{L}[i,j] \quad := \quad \mathrm{lexmax}\begin{cases} \mathbf{L}[i-1,j-1] + (\mu(i,j), 0) \\ \mathbf{L}[i,j-1] + (0, -1) \\ \mathbf{L}[i-1,j] + (0, -1) \end{cases}$$

with basis cases $\mathbf{L}[0,0] := (0,0)$, $\mathbf{L}[i,0] := (0, -i)$ and $\mathbf{L}[0,j] := (0, -j)$, for $1 \le i \le n_1$ and $1 \le j \le n_2$. The operator lexmax keeps only the lexicographic maximum of the three states in the recursive step.

2.2.2 Upper bound The pruning technique proposed in this article follows a branch-and-bound principle. Let $t := (s, -d)$ be a state at entry $\mathbf{P}[i,j]$. Let u be the maximum substitution score and v the minimum number of indels that can be achieved from entry $\mathbf{P}[i,j]$ to entry $\mathbf{P}[n_1,n_2]$. Then, for either $1 \le i < n_1$ or $1 \le j < n_2$, we consider the following upper bound for t: $\mathrm{ub}(t) := (s+u, -d-v)$. If ub($t$) is located in the interior of $\mathcal{R}$, then state t will not lead to any state that corresponds to a score vector of a Pareto optimal alignment and can be discarded from entry $\mathbf{P}[i,j]$.

The value of u can be computed with the algorithm of Needleman and Wunsch (1970) with a null indel penalty for the sequences $(a_{i+1}, \ldots, a_{n_1})$ and $(b_{j+1}, \ldots, b_{n_2})$. This can be easily obtained for every entry in matrix **P** in a pre-processing step. The minimum number of indels v is computed by the absolute difference between the sizes of two subsequences $(a_i, \ldots, a_{n_1})$ and $(b_j, \ldots, b_{n_2})$, i.e. $v := |(n_2 - j) - (n_1 - i)|$. Therefore, $\mathrm{ub}(s) := (s+u, -d-v)$ is a valid upper bound for state t. This bound may not correspond to a feasible alignment.

Matrix **L** as well as the algorithm of Needleman and Wunsch (1970) can be computed in a pre-processing phase in $O(n_1 \cdot n_2)$-time. Hence, the upper bound at each matrix entry can be computed in a constant amount of time during the main phase of the algorithm.

2.3 Bounds in problem (VSGP)

For Problem (VSGP), the computation of lower and upper bounds follow the same reasoning as for Problem (VSDP). In the following, we will only give a brief explanation and highlight the main differences.

2.3.1 Lower bound set The lexicographic and scalarized problems described in Section 2.2.1 can also be formalized in terms of gaps. In this case, MAX and MIN correspond to the score vectors of the Pareto optimal alignments that maximize the substitution score and minimize the number of gaps, respectively. Also, MID corresponds to the score vector of the Pareto optimal

alignment that maximizes a scalarized score function by taking into account the number of gaps (we use w_g instead of w_d).

Score vectors MAX and MID can be computed in $O(n_1 \cdot n_2)$-time by using the algorithm described in Gusfield (1997) with the necessary changes (see Section 2.2.1). However, score vector MIN can be computed faster. The minimum number of gaps can only be zero or one, the latter case arising when $n_1 \neq n_2$. Assume w.l.o.g. that $n_1 < n_2$. Then, the computation of the maximum substitution score that is possible for one gap can be performed by comparing the substitution score for each of the $n_1 + 1$ possible locations of a gap. This can be performed in $O(n_1)$-time in an incremental manner.

2.3.2 Upper bound In this problem, the computation of the maximum substitution score that can be achieved in entry $\mathbf{Q}[n_1, n_2]$ by a state t at entry $\mathbf{Q}[i, j]$ follows the same procedure as explained in Section 2.2.2.

For the computation of the minimum number of gaps, we consider a partition of matrix $\mathbf{Q}$ into three sections; w.l.o.g., we assume that $n_1 < n_2$. Let $\mathbf{Q}^D := \mathbf{Q}[i - n_2 + n_1, i]$, for $n_2 - n_1 \leq i \leq n_2$, which corresponds to the diagonal in $\mathbf{Q}$ starting at $\mathbf{Q}[0, n_2 - n_1]$ and ending at $\mathbf{Q}[n_1, n_2]$. Let $\mathbf{Q}^A$ and $\mathbf{Q}^B$ denote the entries in matrix $\mathbf{Q}$ that are located above and below $\mathbf{Q}^D$, respectively. From this partitioning of $\mathbf{Q}$, we can derive the following results for v, the minimum number of gaps, that is achieved at $\mathbf{Q}[n_1, n_2]$ by a state t (we relate state t with a partial alignment $\varphi := (A', B')$):

(i) If state $t \in \mathbf{Q}^D$, then $v := 0$;
(ii) If state $t \in \mathbf{Q}^A$ ($\mathbf{Q}^B$) and alignment φ ends with a gap in A' (B'), then $v := 0$;
(iii) If state $t \in \mathbf{Q}^A$ ($\mathbf{Q}^B$) and alignment φ ends with two characters or a gap character in B' (A'), then $v := 1$.

Conditions (ii) and (iii) can be determined by keeping an additional variable that stores whether state t was obtained from matrix $\mathbf{R}$, $\mathbf{S}$, or $\mathbf{T}$ in the recursion. Therefore, the upper bound for the case of gaps can also be computed in a constant amount of time.

2.4 Performance of the pruning technique

We performed an experimental analysis for comparing the performance of the algorithms described in the previous sections, with and without the pruning technique (Prune and NoPrune, respectively) and for both problem variants with substitution matrix PAM250 (Dayhoff *et al.*, 1978). The implementations were coded in C and compiled with gcc version 4.6.1 with the -O3 compiler option, in a computer with 2 processors Intel Xeon 5620, 2.4 GHz, 4 core and 16 GB RAM, with operating system Ubuntu 11.10. Except of the compiler option, no other code optimization technique was used in the experiments.

We considered the sequences available from the benchmark BAliBase version 3.0 (Thompson *et al.*, 2005) reference set 9. The subsets RV911, RV912 and RV913 were chosen, as they are organized into three different groups according to the sequence variability: <20%, 20–40% and 40–80% identity, respectively. From the datasets, we extracted the sequences from the following groups: RV911-BOX096 (12 sequences), RV911-BOX115 (7 sequences), RV911-BOX010 (18 sequences), RV912-BOX075 (13 sequences), RV912-BOX258 (16 sequences), RV912-BOX154 (5 sequences), RV913-BOX158 (55 sequences), RV913-BOX222 (7 sequences) and RV913-BOX063 (8 sequences). Our implementations were run on all pairs of sequences of the same group.

Preliminary experiments indicated that only three bounds (MAX, MID and MIN) were insufficient for obtaining good performance. For this reason, several weighted sum problems were solved for different weight combinations to obtain a tighter lower bound set: 5, 10, 15 and 20. For each bound w, the weights were varying in the following manner: $w_s := i,\ w_d := w - i,\ i \in \{1, \ldots, w - 1\}$. Other experiments that we performed indicated that no improvement in terms of pruning can be obtained for w values larger than 20. In a second set of experiments for Problem (VSGP), we observed that the pruning technique was only being effective for entries $Q[i, j],\ i \geq n_1/2$ and $j \geq n_2/2$. Therefore, to reduce the overall CPU time, we switched off the pruning technique for lower indices.

Tables 1 and 2 give the results obtained for both problem variants, where *size* is the average sequence length, *#nd* corresponds to the average number of non-dominated states, *CPU time* gives the average and standard deviation of CPU times in seconds to terminate and *%prun* gives the percentage of states that were pruned in the Prune(w) version. A bold value indicates the best average CPU time; in case of a tie, the value with the largest pruning percentage was chosen, as it suggests less memory usage.

The results show that the Prune version is able to prune from 31 to 92% of the states that are generated by the NoPrune version for Problem (VSDP). The improvement in terms of CPU time can go up to 80% in the set RV913-BOX063. However, no improvement can be found for RV912-BOX075 and RV912-BOX154, although all versions were extremely fast in those cases (≤ 0.5 s). For Problem (VSGP), the pruning can reach 83% and CPU time improved up to 60% in RV913-BOX222. In both problems, it is possible to observe that the increase of parameter w does not translate directly into faster CPU time; for instance, in the sets RV911 for Problem (VSGP), the best CPU time was obtained with $w = 10$, although better pruning percentage was obtained with $w = 20 (\geq 20\%)$. Clearly, the larger the lower bound set, the higher the required time for comparison. Moreover, the pruning seems to be more effective for large levels of residue identity.

It is also noteworthy to mention that the number of non-dominated states is a small fraction of the average size of the genes. We also observed that both algorithms on Problem (VSGP) take roughly 4–5 times more CPU time than on Problem (VSDP).

3 PHYLOGENETIC TREE CONSTRUCTION

Phylogenetic trees are diagrams that illustrate historical relationships among the species. They have a valuable application in evolutionary and population biology [we refer to Schuh and Brower (2009) for more details]. In the context of our bicriteria framework, it is important to understand whether the non-dominated score set can provide further information than that provided by existing methods. In this section, we describe a method for constructing phylogenetic trees from the non-dominated score set and discuss its application for deriving further

Table 1. Experimental results for the pruning technique for problem (VSDP)

Datasets	Size	#nd	CPU-time (%prun)					
			noPrune	Prune (1)	Prune (5)	Prune (10)	Prune (15)	Prune (20)
RV911-BOX115	750	190	0.5 ± 0.2	0.4 ± 0.2 (45%)	0.4 ± 0.2 (53%)	0.4 ± 0.2 (58%)	0.4 ± 0.2 (59%)	**0.4** ± 0.2 (60%)
RV911-BOX096	795	189	0.6 ± 0.3	0.5 ± 0.2 (38%)	0.4 ± 0.2 (55%)	0.4 ± 0.2 (60%)	0.4 ± 0.2 (62%)	**0.4** ± 0.2 (63%)
RV911-BOX010	1134	269	2.1 ± 1.3	1.5 ± 1.2 (31%)	1.4 ± 1.1 (35%)	1.3 ± 1.1 (62%)	1.4 ± 1.2 (63%)	**1.4** ± 1.2 (65%)
RV912-BOX075	457	114	0.1 ± 0.0	0.1 ± 0.0 (53%)	0.1 ± 0.0 (71%)	0.1 ± 0.0 (75%)	0.1 ± 0.0 (76%)	**0.1** ± 0.0 (77%)
RV912-BOX258	607	106	0.3 ± 0.1	0.1 ± 0.1 (67%)	0.1 ± 0.1 (84%)	0.1 ± 0.1 (86%)	**0.1** ± 0.1 (87%)	**0.1** ± 0.1 (87%)
RV912-BOX154	1076	164	0.8 ± 0.4	0.5 ± 0.3 (59%)	0.4 ± 0.2 (71%)	0.4 ± 0.2 (74%)	0.4 ± 0.2 (75%)	**0.4** ± 0.2 (76%)
RV913-BOX158	664	74	0.2 ± 0.1	0.1 ± 0.0 (83%)	0.0 ± 0.0 (91%)	**0.0** ± 0.0 (92%)	**0.0** ± 0.0 (92%)	**0.0** ± 0.0 (92%)
RV913-BOX222	983	149	0.8 ± 0.2	0.3 ± 0.2 (72%)	0.2 ± 0.1 (86%)	0.2 ± 0.1 (88%)	**0.2** ± 0.1 (89%)	**0.2** ± 0.1 (89%)
RV913-BOX063	1374	206	2.0 ± 0.4	0.7 ± 0.3 (76%)	0.4 ± 0.2 (88%)	0.4 ± 0.2 (90%)	**0.4** ± 0.2 (91%)	**0.4** ± 0.2 (91%)

Table 2. Experimental results for the pruning technique for problem (VSGP)

Datasets	Size	#nd	CPU-time (%prun)					
			noPrune	Prune (1)	Prune (5)	Prune (10)	Prune (15)	Prune (20)
RV911-BOX115	750	239	3.8 ± 0.9	3.5 ± 1.4 (1%)	3.3 ± 1.4 (10%)	**3.3** ± 1.4 (16%)	3.4 ± 1.5 (19%)	3.5 ± 1.5 (21%)
RV911-BOX096	793	264	4.9 ± 2.2	4.6 ± 3.5 (0%)	4.5 ± 3.6 (9%)	**4.5** ± 3.7 (15%)	4.6 ± 3.8 (18%)	4.8 ± 3.9 (19%)
RV911-BOX010	1133	382	13.7 ± 5.6	13.6 ± 9.1 (0%)	12.8 ± 8.8 (10%)	**12.6** ± 8.9 (14%)	12.8 ± 9.1 (18%)	13.1 ± 3.4 (22%)
RV912-BOX075	458	139	0.8 ± 0.1	0.8 ± 0.1 (1%)	0.6 ± 0.1 (30%)	0.5 ± 0.1 (46%)	0.5 ± 0.1 (52%)	**0.5** ± 0.1 (54%)
RV912-BOX258	608	153	1.8 ± 0.5	1.8 ± 0.8 (3%)	1.2 ± 0.5 (43%)	0.9 ± 0.4 (59%)	0.8 ± 0.4 (65%)	**0.8** ± 0.4 (69%)
RV912-BOX154	1076	243	5.3 ± 1.2	5.3 ± 2.0 (1%)	4.7 ± 1.6 (14%)	4.3 ± 1.4 (26%)	4.2 ± 1.3 (31%)	**4.2** ± 1.3 (34%)
RV913-BOX158	671	69	1.1 ± 0.2	1.3 ± 0.3 (2%)	0.5 ± 0.2 (66%)	0.3 ± 0.1 (79%)	0.3 ± 0.1 (82%)	**0.3** ± 0.1 (83%)
RV913-BOX222	983	180	5.1 ± 0.8	4.9 ± 1.3 (5%)	2.7 ± 0.7 (47%)	1.9 ± 0.6 (65%)	1.7 ± 0.6 (71%)	**1.6** ± 0.6 (74%)
RV913-BOX063	1374	275	10.4 ± 1.5	10.6 ± 2.6 (5%)	7.3 ± 2.1 (54%)	5.0 ± 1.7 (70%)	4.3 ± 1.5 (75%)	**4.1** ± 1.4 (77%)

information about the *reliability* of the tree branches. Two datasets are used for illustration purpose, and the resulting trees are compared with those obtained with the well-known technique of Maximum Likelihood (ML).

3.1 The bicriteria method

Given a collection of non-dominated score sets, each one obtained for each pair of sequences, our approach consists of building a phylogenetic tree for each gap/indel value that arises from the union of all non-dominated scores sets. First, for each gap/indel value found, we collect the set of substitution score values, one for each pair of species under study. The substitution score value from the resulting score vector is normalized between zero and one. Then, the distance between two species for a given gap/indel value is computed as one minus the normalized substitution score value. In our experiments, we used the resulting distance matrix to build each phylogenetic tree by the Neighbor-Joining method (Saitou and Nei, 2007) using PHYLIP package (Felsenstein, 1985).

An important aspect of the analysis is to understand how often certain phylogenetic tree topologies exist. A less frequent topology, or tree branch, may indicate a less reliable relation between the corresponding species. Therefore, we count how many times each branch arises in all phylogenetic trees and add the corresponding relative frequency to each branch. This information is analogous to the bootstrap values introduced by Felsenstein (1985); instead of the sampling process, we use the non-dominated score sets. In the following, we describe two experiments with real-life data.

3.2 First experiment

The first dataset consists of *Candida* genes, *Candida albicans PAPa*, *C.albicans PAPalpha*, *Candida tropicalis PAPalpha*, *C.tropicalis PAPa* and *Candida dubliniensis PAPa*, as well as the genes *Pichia stipitis PAPa*, and *Saccharomyces cerevisiae PAP*. [See Butler *et al*. (2009) for a more detailed description of these genes.] For each pair of genes, we computed the non-dominated score set with respect to Problem (VSGP) with substitution matrix $\mathbf{M}[i,i] := 1$ and $\mathbf{M}[i,j] := -1$, $i \neq j$; see the complete non-dominated score sets with staircase line representation in Figure 2. It is possible to observe that there exists a large number of non-extreme supported score vectors, as indicated by the straight lines, which cannot be found by a parametric sequence alignment (see Steuer, 1986, Chapter 14).

We constructed 567 phylogenetic trees by using our method, but only two different tree topologies were obtained. One of two tree topologies was discarded from the analysis, as it arose only once of the 576 trees. Figure 3a shows the remaining tree

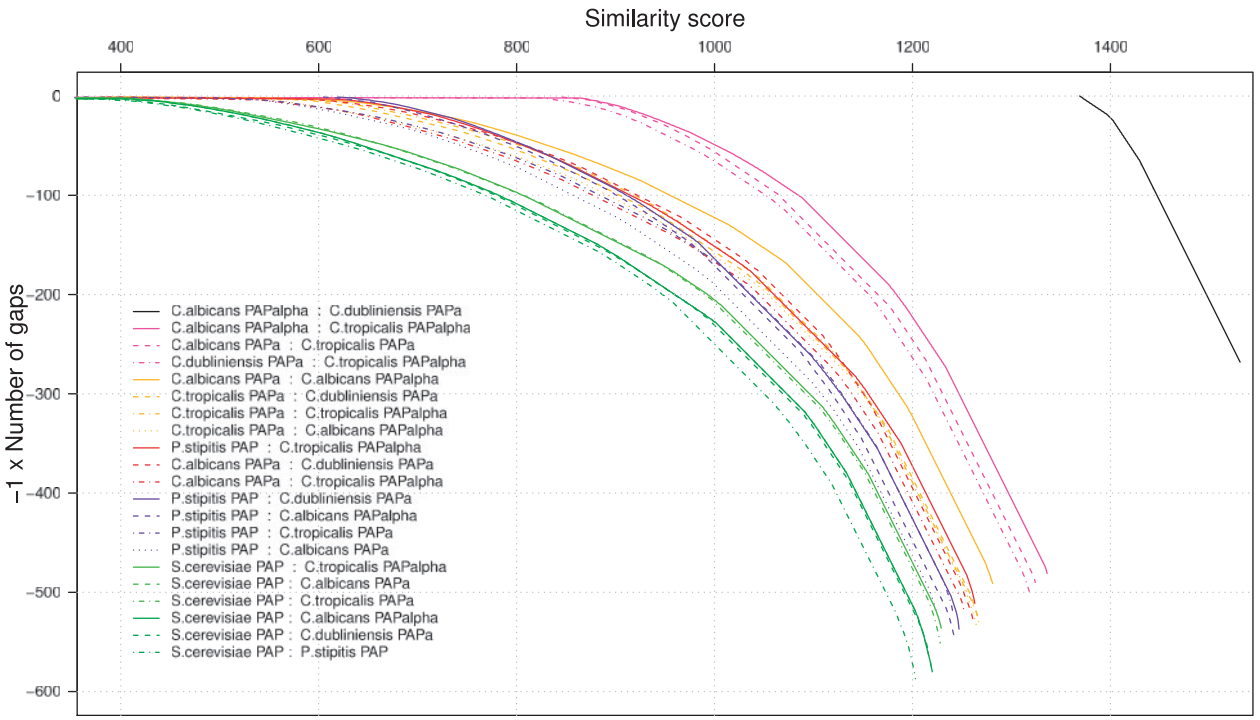

Fig. 2. Staircase line representation of the non-dominated score sets for the first experiment

obtained for a gap value of 283, the median of all gap values found; the value close to each branch indicates the relative frequency of that branch. For comparison, we computed the evolutionary tree using the ML method based on the model from Jukes and Cantor (1969) obtained by MEGA5 (Tamura *et al.*, 2011); see Figure 3b. The bootstrap consensus tree inferred from 1000 replicates (Felsenstein, 1985) was taken to represent the evolutionary history of the taxonomic analysis. There were a total of 1613 positions in the final dataset with all of them having <90% site coverage removed. The percentage of trees in which the associated taxa clustered together is shown next to the branches. Initial tree(s) for the heuristic search were obtained automatically as follows: when the number of common sites was <100 or less than one-fourth of the total number of sites, the maximum parsimony method was used; otherwise BIONJ method with MCL distance matrix was used. The tree is drawn to scale, with branch lengths measured in the number of substitutions per site.

By comparing both trees, it is possible to infer that they have a similar topology and similar relative branch frequency. This conclusion can also be understood by the regularity of the lines of Figure 2. The nearest genes are *C.albicans PAPalpha* and *C.dubliniensis PAPa*, which corresponds to the black line in Figure 2, with the largest substitution score. In the same figure, the three pink lines allow us to infer the positioning of C. *tropicalis PAPalpha* with respect to *C.albicans PAPalpha* and *C. dubliniensis PAPa*, as well as the branch (*C.albicansPAPa*, *C. dubliniensis PAPa*). As expected, *P.stipitis PAP* and *S.cerevisiae PAP* are the most distant species, as seen by the green dot-dash line in Figure 2.

Finally, both phylogenetic trees indicate a lower value for the branch that contains *P.stipitis PAP* gene. We relate this value to

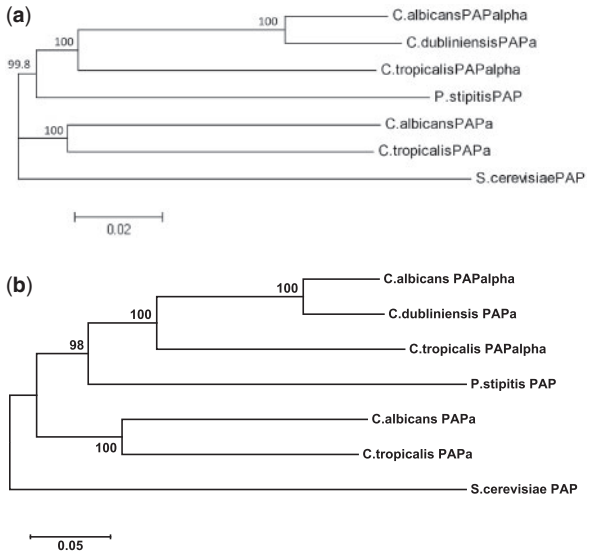

Fig. 3. Phylogenetic trees for the first experiment: Bicriteria model for a gap value of 283 (**a**) and ML model (**b**)

the crossed lines between the pairs formed by *P.stipitis PAP* and others genes (see blue lines in Fig. 2).

3.3 Second experiment

The second dataset is a classic example of comparison between primates: *Homo sapiens haplogroup J1c3*, *Homo sapiens neanderthalensis*, *Gorilla gorilla graueri*, *Pan troglodytes troglodytes* and *Pongo abelii* species. We performed the same analysis as

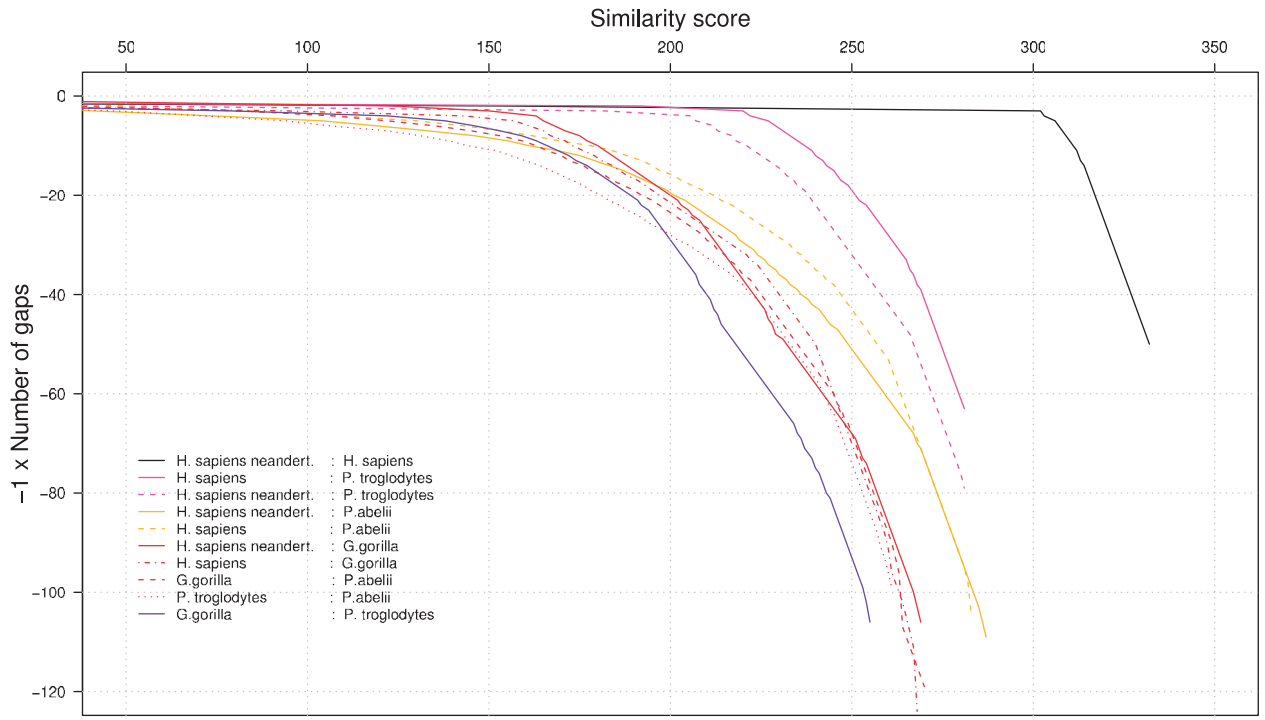

Fig. 4. Staircase line representation of the non-dominated score sets for the second experiment

described in the previous section. Figure 4 shows the non-dominated score sets with staircase line representation. The figure shows a large number of non-extreme supported score vectors and some intersecting lines. This last point indicates that the relationship between those species, in the context of evolutionary studies, may depend of the score vectors chosen (or the scalarized score function used). We conjecture that the existence of intersections may indicate a less reliable conclusion about the evolutionary relationship.

By using our bicriteria method, a total of 144 phylogenetic trees were obtained, which gave rise to two different tree topologies. Plots (a) and (b) of Figure 5 show the two trees topologies obtained for a gap value of 22 and 54, respectively; these gaps values correspond to the median of all gap values found in the phylogenetic trees with the same topology. The evolutionary tree using the ML method was also computed using the same method described in the previous section. The tree with the highest log likelihood (−1351.1512) is shown in Figure 5c. All positions with <60% site coverage were eliminated. That is, <40% alignment gaps, missing data and ambiguous bases were allowed at any position. There were a total of 376 positions in the final dataset.

The two trees obtained with our method differ slightly in relationship to the *Pan traglodytes traglodytes* with the remaining species: in plot (a), this species arose in the top of the branch (*Homo sapiens neanderthalensis*, *Homo sapiens haplogrop J1c3*), whereas in plot (b), it arises in a different clade, paired with *Gorilla gorilla graueri*. The relative branch frequencies suggest that the tree of plot (b) may be more reliable. Interestingly, this is also confirmed by the tree obtained with the ML method.

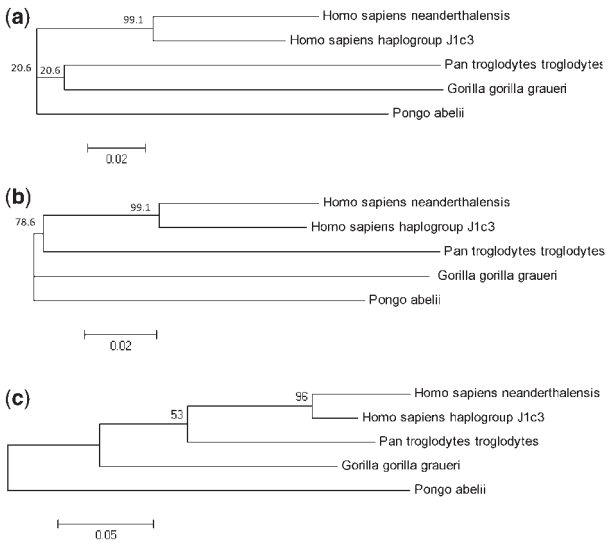

Fig. 5. Phylogenetic trees for the second experiment: Bicriteria model for a gap value of 22 (**a**) and 54 (**b**) and ML model (**c**)

4 CONCLUSION

In this article, a pruning technique is introduced to improve the performance of dynamic programming algorithms for bicriteria pairwise sequence alignment, which uses lower and upper bounds to discard states in early stages of the process. This technique can easily be extended for the case of affine gap by performing the necessary changes in the recurrence relation of matrices **S** and **T**

(see Section 2.1). They can also be extended for the three criteria case, where the substitution score, the number of indels and the number of gaps are simultaneously considered in the score vector function. In addition, this pruning technique can also be used in multi-criteria multiple sequence alignment, for instance, in the context of progressive alignment with sum-of-pairs score function.

In the second part of this article, we showed for the first time a successful link between non-dominated score sets and phylogenetic tree construction. We present a simple method based on our bicriteria framework that allows to construct phylogenetic trees as well as to give information about the reliability of the tree branches. The advantage of this method is that no assumption about a priori knowledge on users preferences is required, therefore, being less unbiased. The two real-life test cases showed that few phylogenetic trees can be obtained and are matched with those obtained with the ML method. Further research is needed to derive theoretical relation between ML estimations and the information provided by the non-dominated score set for tree branch reliability.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the anonymous referees for their valuable comments and suggestions on the paper.

Funding: Portuguese Foundation for Science and Technology and FEDER, Programa Operacional Factores de Competitividade - COMPETE, FEDER - FCOMP-01-0124-FEDER-010024, under the project 'Multiobjective Sequence Alignment' (PTDC/EIA-CCO/098674/2008).

Conflict of Interest: none declared.

REFERENCES

Beier,R. and Vöcking,B. (2011) The knapsack problem. In: *Algorithms Unplugged*. Springer-Verlag, Berlin, Heidelberg, pp. 375–381.

Butler,G. *et al.* (2009) Evolution of pathogenicity and sexual reproduction in eight candida genomes. *Nature*, **459**, 657–662.

Dayhoff,M. *et al.* (1978) A model of evolutionary change in proteins. In: Dayhoff,M. (ed.) *Atlas of Protein Sequence and Structure*, Vol. 5. National Biomedical Research, Washington, D.C., pp. 345–358.

Ehrgott,M. (2005) *Multicriteria Optimization*. Springer-Verlag, Berlin, Heidelberg.

Felsenstein,J. (1985) Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, **39**, 783–791.

Gusfield,D. *et al.* (1992) Parametric optimization of sequence alignment. In: *Proceedings of the Third Annual ACM-Siam Symposium on Discrete Algorithms*. pp. 432–439.

Gusfield,D. (1997) *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA.

Handl,J. *et al.* (2007) Multiobjective optimization in bioinformatics and computational biology. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **4**, 279–292.

Jukes,T.H. and Cantor,C.R. (1969) *Evolution of Protein Molecules*. Academy Press, New York.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–453.

Paquete,L. and Almeida,J. (2009) Experiments with bicriteria sequence alignment. In: *Cutting-Edge Research Topics on Multiple Criteria Decision Making*. Vol. 35 of *CCIS*. Springer-Verlag, Berlin, Heidelberg, pp. 45–51.

Roytberg,M. (1992) Fast algorithm for optimal aligning of symbol sequences. In: *Mathematical Methods of the Analysis of Biopolymer Sequences*. Vol. 8 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* AMS, Providence, RI, pp. 113–127.

Roytberg,M. *et al.* (1999) Pareto-optimal alignment of biological sequences. *Biophysics*, **44**, 565–577.

Saitou,N. and Nei,M. (2007) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, **4**, 406–425.

Schuh,R. and Brower,A. (2009) *Biological Systematics: Principles and Applications*. Cornell University Press, Ithaca, NY.

Steuer,R.E. (1986) *Multiple Criteria Optimization: Theory, Computation, and Application*. Wiley & Sons, New York, NY.

Tamura,K. *et al.* (2011) Mega5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. *Mol. Biol. Evol.*, **28**, 2731–2739.

Taneda,A. (2010) Multi-objective pairwise RNA sequence alignment. *Bioinformatics*, **26**, 2383–2390.

Thompson,J. *et al.* (2005) Balibase 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, **61**, 127–136.