Novel definition and algorithm for chaining fragments with proportional overlaps

Raluca Uricaru* Alban Mancheron* Eric Rivals*

8th May 2011

Abstract

Chaining fragments is a crucial step in genome alignment. Existing chaining algorithms compute a maximum weighted chain with no overlaps allowed between adjacent fragments. In practice, using local alignments as fragments, instead of MEMs, *i.e.* Maximal Exact Matches [6], generates frequent overlaps between fragments, due to combinatorial reasons and biological factors, *i.e.* variable tandem repeat structures that differ in number of copies between genomic sequences. In this paper, in order to raise this limitation, we formulate a novel definition of a chain, allowing overlaps proportional to the fragments lengths, and exhibit an efficient algorithm for computing such a maximum weighted chain. We tested our algorithm on a dataset composed of 694 genome pairs and accounted for significant improvements in terms of coverage, while keeping the running times below reasonable limits. Moreover, experiments with different ratios of allowed overlaps showed the robustness of the chains with respect to these ratios.

1 Introduction

In biology, genome comparison is used for gene annotation, phylogenetic studies, and even vaccine design [13, 2, 7]. Many bioinformatics programs for whole genome comparison involve a fragment chaining step, which seeks to maximize the total length of the chained fragments, eg [6]. Given the set of n shared genomic intervals, *i.e.* fragments, the Maximum Weighted Chain problem (MWC) is solved in $O(n \log n)$ time by dynamic programming when overlaps between adjacent fragments are forbidden [11, 1]. Alternatively, Felsner *et al.* showed that this problem is a special case of the Maximum Weighted Independent Set problem in a trapezoid graph, which they solve by a sweep line algorithm over an equivalent box order representation of the graph [5]. These algorithms [1, 5] can be extended to handle fixed length overlap between adjacent fragments, but this is not sufficient to deal with the large differences in fragment length obtained even with small bacterial genomes [15]. An $O(n \log n)$ time algorithm for the MWC with Fixed Length Overlaps problem was designed and used for mapping spliced RNAs on a genome [14], but the fixed bound on overlaps remains a limitation. To raise this limitation, we formulate the MWC with Proportional Length Overlaps problem (MWC-PLO) and exhibit the first chaining algorithms allowing for overlaps that are proportional to the fragment lengths, and whose chain weight function accounts for overlap. Following Felsner *et al.*, we use the box representation of a trapezoid graph and adapt the sweep line paradigm to this problem.

Small overlaps are often caused by equality over a few base pairs of fragment ends due to randomness, since the alphabet has only four letters. To handle such cases, one could set a constant, large enough, maximal allowed overlap threshold. However, biological structures like tandem repeats (TR) that vary in number of copy units generate overlaps that are large relatively to the fragments involved. To illustrate this case, let u, v, w be words and assume the sequences of two genomes G_a, G_b are $G_a = uvvw$ and $G_b = uvvvw$, *i.e.* contain a variable TR of motif v. Then, uvv generates a local alignment between G_a and G_b, as well as vvw, but both fragments overlap each other over v in both G_a and G_b. Since v can be arbitrarily large, the fragment overlap can be as large as v itself. Such cases cannot be circumvented with fixed length overlaps: only proportional overlaps can handle these. As variable-number tandem repeats occur in genomes of

*LIRMM, CNRS and Université de Montpellier 2, Montpellier, France

numerous species coming from all kingdoms of life, the problem of dealing with proportional overlaps is of great importance.

The paper is organised as follows. Section 2 presents the chaining problem without overlaps, while Section 3 defines chaining with proportional overlaps and sets the dynamic programming framework and algorithm that solves it. In Section 4 we exhibit a sweep line algorithm for this question, prove its correctness and discuss the complexities. In Section 5, we study the performance of our algorithm, implemented in a tool called OverlapChainer (OC), we investigate the robustness of the results with respect to different overlap ratios and compare the results obtained with proportional overlaps to those obtained with fixed overlap thresholds. We conclude in Section 6.

2 Preliminaries

Boxes are axis parallel hyper-rectangles in $\mathbb{R}^k$, where each genome is associated with one axis. For simplicity, we consider the two dimensional case where $k = 2$, *i.e.* comparing two genomes. The length on a genome of the fragment associated with a box is the projection of that box on the corresponding axis.

Let $\alpha \in \{1,2\}$ index the axis, and for any point $x \in \mathbb{R}^2$ let $P_\alpha(x)$ denote its projection on axis α. Let I be an interval of $\mathbb{R}$ and $\mathcal{I}$ be a set of disjoint intervals of $\mathbb{R}$; we denote by $|I|$ the length of I and by $|\mathcal{I}|$ the sum of the lengths of intervals in $\mathcal{I}$. Let B be a box of $\mathbb{R}^2$. The **upper right**, resp. **lower left**, corner of B is denoted by $u(B)$, resp. $l(B)$. By extension, the interval corresponding to the projection of B on axis α is denoted $P_\alpha(B)$. Let $<$ denote the classical **dominance order** between points of $\mathbb{R}^2$.

Definition 1 (Overlap free box dominance order)**.** *Let B_x, B_y be two boxes of $\mathbb{R}^2$. We say that B_y **dominates** B_x, denoted $B_x \ll B_y$, if $l(B_y)$ dominates $u(B_x)$ in $\mathbb{R}^2$. If neither B_x dominates B_y, nor B_y dominates B_x, then B_x and B_y are **incomparable**.*

Felsner *et al.* showed how to transform a trapezoid graph into a **box order**, *i.e.* a set of boxes equipped with the dominance order $\ll$ such that pairs of incomparable boxes are in one-to-one correspondence with trapezoid pairs linked by edges of the graph. Hence, the Maximum Weighted Independent Set problem in a trapezoid graph is equivalent to the MWC problem in the corresponding box order [5]. Given an order, recall that a **chain** is a set of mutually comparable elements, and a **maximal element** in a set is one with no other element dominating it. Each chain has exactly one maximal element.

3 A novel tolerance definition for the Maximum Weighted Chain problem in a box order

To formulate a MWC with Proportional Length Overlaps problem (MWC-PLO) in our framework, we need to redefine the dominance order to accept overlaps that are proportional to the boxes' projection lengths, and to propose a weight function that truly measures the coverage on each genome. By **coverage**, it is generally meant the total length of the genomic intervals covered by the selected fragments [10] . This requires that the chain weight counts only once a subinterval covered by several overlapping fragments.

Let $r \in [0,1[$ represent the maximal allowed overlap ratio between any two boxes.

Definition 2 (r tolerant dominance order)**.** *Let B_u and B_v be two boxes. B_v **dominates** B_u **on axis** α in this **tolerant dominance order**, denoted by $B_u \ll_{r,\alpha} B_v$, if and only if*

$$P_\alpha(u(B_u)) - P_\alpha(l(B_v)) \leq r \min(|P_\alpha(B_u)|, |P_\alpha(B_v)|).$$

*Now, we denote by $B_u \ll_r B_v$ the fact that B_v **dominates** B_u if and only if for each $\alpha \in \{1,2\}$ $B_u \ll_{r,\alpha} B_v$.*

It can be easily shown that the dominance between boxes implies the dominance between their upper, resp. lower, corners. Moreover, this tolerant dominance order is transitive.

Property 1. *Let B_t, B_u two boxes such that $B_t \ll_r B_u$. Then $l(B_t) < l(B_u)$ and $u(B_t) < u(B_u)$.*

Property 2. *The dominance order $\ll_r$ is transitive.*

Proof of Property 2 (transitivity of $\ll_r$). Let B_t, B_u, B_v be three boxes such that $B_t \ll_r B_u$ and $B_u \ll_r B_v$. We will show that $B_t \ll_r B_v$. Let $\alpha \in \{1,2\}$. By hypothesis and from Property 1, we obtain both $l(B_t) < l(B_u) < l(B_v)$ and $u(B_t) < u(B_u) < u(B_v)$. From these we get both

$$P_\alpha(u(B_t)) - P_\alpha(l(B_v)) < P_\alpha(u(B_t)) - P_\alpha(l(B_u)) \leq r \min(|P_\alpha(B_t)|, |P_\alpha(B_u)|), \tag{1}$$

and

$$P_\alpha(u(B_t)) - P_\alpha(l(B_v)) < P_\alpha(u(B_u)) - P_\alpha(l(B_v)) \leq r \min(|P_\alpha(B_u)|, |P_\alpha(B_v)|). \tag{2}$$

When combined, these equations imply

$$\begin{aligned} P_\alpha(u(B_t)) - P_\alpha(l(B_v)) &\leq r \min(|P_\alpha(B_t)|, |P_\alpha(B_u)|, |P_\alpha(B_v)|) \\ &\leq r \min(|P_\alpha(B_t)|, |P_\alpha(B_v)|)), \end{aligned}$$

and hence $B_t \ll_r B_v$. □

From Property 1, one deduces the following corollary, which will help to compute efficiently the weight of overlapping boxes in a chain.

Corollary 1. *Let B_t, B_u, B_v be three boxes such that $B_t \ll_r B_u \ll_r B_v$. Then: $(B_t \cap B_v) \subset (B_u \cap B_v)$.*

We define the **weight of a box** as the sum of lengths of its projections on all axis, and the **weight of a chain** of boxes as the sum of the coverages on each axis.

Definition 3 (Weight of a box, of a chain)**.** *Let B be a box and $\alpha \in [1,2]$. Its weight on axis α is $w_\alpha(B) := |P_\alpha(B)|$, and its weight is $w(B) := \sum_{\alpha=1}^{2} w_\alpha(B)$. Let $m \in \mathbb{N}$ and $C := (B_1 \ll_r \ldots \ll_r B_m)$ be a chain of m boxes. The weight of C on axis α, denoted $W_\alpha(C)$, is*

$$W_\alpha(C) := \left| \bigcup_{i=1}^{m} P_\alpha(B_i) \right|,$$

*while its **weight** is $W(C) := \sum_{\alpha=1}^{2} W_\alpha(C)$.*

Note also that the weight of a box only depends on the endpoints of its projection on each axis, and hence, can be computed in constant time.

Clearly, it can be easily seen that

$$\begin{aligned} W_\alpha(C) &= w_\alpha(B_m) + \sum_{j=1}^{m-1} \left| P_\alpha(B_j) \setminus \bigcup_{l=j+1}^{m} P_\alpha(B_l) \right| \\ &= w_\alpha(B_m) + \sum_{j=1}^{m-1} \left| P_\alpha(B_j) \setminus P_\alpha(B_{j+1}) \right| \quad \text{by Corollary 1.} \end{aligned} \tag{3}$$

The following easy property will also prove useful.

Property 3. *Let B_t, B_u two boxes such that $B_t \ll_r B_u$. Then*

- *$B_t \cap B_u$ is an, eventually empty, axis parallel rectangle of $\mathbb{R}^2$, and*
- *for $\alpha \in [1,2]$, $|P_\alpha(B_t) \setminus P_\alpha(B_u)| = |P_\alpha(B_t) \setminus P_\alpha(B_t \cap B_u)| = w_\alpha(B_t) - w_\alpha(B_t \cap B_u)$.*

Now, we can define the MWC-PLO problem. Let $\mathcal{B}' := \{B_2, \ldots B_{n-1}\}$ be the set of input boxes. For convenience, we add two dummy boxes, B_1, B_n, such that for all $1 < i < n$: $B_1 \ll_r B_i \ll_r B_n$. Additionally, we set $w(B_1) = w(B_n) := 0$. Now, the input consists in $\mathcal{B} := \{B_1, \ldots, B_n\}$.

Definition 4 (MWC with Proportional Length Overlaps)**.** *Let $r \in [0,1[$ and $\mathcal{B} := \{B_1, \ldots, B_n\}$ a set of boxes. The* MWC with Proportional Length Overlaps *problem is to find in $\mathcal{B}$, according to the dominance order $\ll_r$, the **chain** C that starts with B_1 and ends at B_n and whose weight $W(C)$ is maximal.*

The notation of r, $\mathcal{B}$, and $W(C)$ are valid throughout the paper. For any $1 \leq i \leq n$, let us denote by $\mathcal{C}_i$ the set of chains ending at B_i, and by $W(B_i)$ the weight of the maximal weighted chain in $\mathcal{C}_i$ (not to be confounded with $w(B_i)$). From now on, all the considered boxes belong to $\mathcal{B}$ unless otherwise specified.

3.1 A dynamic programming framework

Let us show that MWC-PLO can be solved by a dynamic programming algorithm. Equation 3 suggests a recurrence equation to compute $W(B_i)$, with $W(B_1) = 0$ and for all $1 < i \leq n$:

$$W(B_i) = \max_{B_j:\, B_j \ll_r B_i} W(B_j) + \sum_{\alpha=1}^{2} \left| P_\alpha(B_i) \setminus P_\alpha(B_j) \right| . \tag{4}$$

Obviously, this implies that for all $1 \leq j < n$ the value of $W(B_j)$ will be reused for computing $W(B_i)$ for every box B_i such that $B_j \ll_r B_i$. Thus, MWC-PLO consists of **overlapping subproblems**, which suits to the framework of dynamic programming [4, chap. 15]. However, it is correct to use Equation 4 only if our problem satisfies the condition of **optimal substructures** [4, chap. 15]. In Theorem 1, we show this is true.

Theorem 1 (Optimality of substructures)**.** *Let $m, i_1, \ldots, i_m$ be integers belonging to $[1,n]$, and let $D := (B_{i_1}, \ldots, B_{i_m})$ be an optimal weighted chain among the chains in $\mathcal{C}_{i_m}$. Thus, $D' := (B_{i_1}, \ldots, B_{i_{m-1}})$ is an optimal weighted chain among those in $\mathcal{C}_{i_{m-1}}$.*

Proof of Theorem 1 (Optimality of substructures). By hypothesis, Equation 3 and Property 3, one has

$$\begin{aligned}
W(D) &= W(B_{i_m}) \\
&= w(B_{i_m}) + \sum_{j=i_1}^{i_{m-1}} \sum_{\alpha} \left| P_\alpha(B_j) \setminus P_\alpha(B_{j+1}) \right| \\
&= w(B_{i_m}) - w(B_{i_m} \cap B_{i_{m-1}}) + w(B_{i_{m-1}}) + \sum_{j=i_1}^{i_{m-2}} \sum_{\alpha} \left| P_\alpha(B_j) \setminus P_\alpha(B_{j+1}) \right| \\
&= w(B_{i_m}) - w(B_{i_m} \cap B_{i_{m-1}}) + W(D') .
\end{aligned}$$

We proceed by contradiction and assume that E', rather than D', is an optimal weighted chain ending at $B_{i_{m-1}}$, *i.e.* $W(E') > W(D')$. Consider the chain $E := D' \cup \{B_{i_m}\}$. By the same reasoning as above, one has

$$W(E) = w(B_{i_m}) - w(B_{i_m} \cap B_{i_{m-1}}) + W(E'),$$

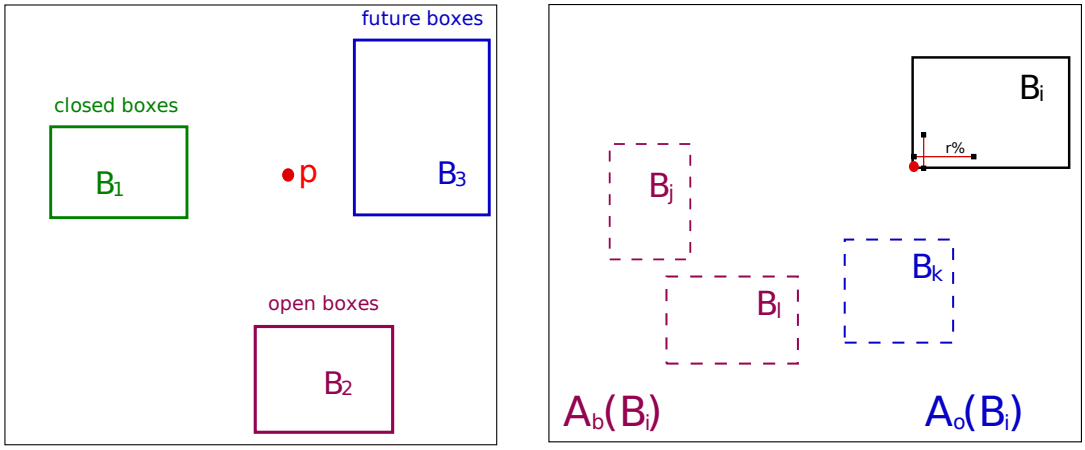
and hence, $W(E) > W(B_{i_m})$, contradicting the hypothesis that D is an optimal weighted chain ending at B_{i_m}. MWC-PLO satisfies the condition of **substructures' optimality**. □

The MWC with Proportional Length Overlaps can thus be solved by a dynamic programming algorithm, which uses two n-element arrays: $W[.]$ and Pred$[.]$ to store for all $1 \leq i \leq n$ resp. the values of $W(B_i)$ and the predecessor of B_i in an optimal weighted chain ending at B_i. This algorithm takes $O(n^2)$ time and $O(n)$ space; in Section 4 we prove an algorithm for MWC-PLO, more efficient in practice.

Theorem 2. *A dynamic programming algorithm (Algorithm DP) solves the* MWC with Proportional Length Overlaps *problem in $O(n^2)$ time and $O(n)$ space.*

4 A sweep line algorithm for MWC with Proportional Length Overlaps

Here, we exhibit a sweep line algorithm for the MWC with Proportional Length Overlaps problem (see Algorithm 1), prove it and study its complexity. Even though we were not able to prove a time complexity below quadratic, we show that in practice, Algorithm 1 is much more efficient than Algorithm DP.



(a) (b)

Figure 1: (a) E.g. of boxes in each disjoint set forming a partition of $\mathcal{B}$, when sweeping a point p. (b) Partition of the search space of possible predecessors of B_i in two areas, $A_b(B_i)$ and $A_o(B_i)$, according to the location of their upper corners: $A_b(B_i)$ at left from the dashed line, and $A_o(B_i)$ at its right.

4.1 Outline of the algorithm

Following Felsner *et al.*, we give a sweep line algorithm in which a vertical line sweeps the boxes in the plane by increasing x-coordinates of their corners, stopping at the lower left and upper right corners of each box. To avoid visiting, as in Algorithm DP, all possible predecessors when computing the best chain ending at B_x, we maintain a set, $\mathcal{A}$, of **active** boxes that can compete for being the optimal predecessor in that chain. But as predecessors can overlap B_x, this computation involves several steps, meaning that $W[B_x]$ and $\text{Pred}[B_x]$ can be updated several times before getting their final value; this differs from Algorithm DP.

Let $\mathcal{P}$ be an array containing the $2n$ points corresponding to $l()$ and $u()$ corners of the n boxes in $\mathcal{B}$. Points in $\mathcal{P}$ are ordered on their x-coordinates; among the points having identical x-coordinates, lower corners are placed before upper corners. For each point, we store to which box and to which corner it corresponds to. In Algorithm 1, the main loop sweeps the points of $\mathcal{P}$ and processes in a different manner lower (lines 5-8) and upper corners (lines 9-21). We say a box B_x is **open** when the sweep line is located between $l(B_x)$ and $u(B_x)$ inclusive, **closed** when the line has passed $u(B_x)$, and **future** when it lies before $l(B_x)$. These states are exclusive of each other, and partition at each moment $\mathcal{B}$ in three disjoint sets (see Figure 1a). All open boxes at each point are kept in a set O (lines 6, 10). The weight of a chain ending at, say B_i, and passing by a predecessor of B_i, B_x, can only be computed when B_x is closed (when $W[B_x]$ has reached its final value). If $P_1(u(B_x)) < P_1(l(B_i))$ then this can be done when stopping at $l(B_i)$ (lines 7-8), while if B_x overlaps B_i on x-axis, then this is done when stopping at $u(B_x)$, and at the same time for all open boxes having B_x as predecessor (lines 11-15). These two cases partition the possible predecessors of B_i according to the location of their upper corners in two areas $A_b(B_i)$ and $A_o(B_i)$ (see Figure 1b).

As above mentioned, we maintain in $\mathcal{A}$ the set of interesting predecessors for all future boxes. Boxes in $\mathcal{A}$ are **active** boxes. Hence, once closing a box (stopping at its upper corner), we test whether it should be turned active and inserted in $\mathcal{A}$ (lines 16-18). The current box, B_i, is inserted only if we cannot find a better predecessor in $\mathcal{A}$. Afterwards, if B_i has been added, currently active boxes are investigated to determine if they are less interesting than B_i, in which case they are deleted from $\mathcal{A}$ (lines 19-21). Active boxes are consulted when opening a box B_i, for computing the best chain ending at B_i with a predecessor in $A_b(B_i)$ (lines 7-8).

4.2 Correctness of the Algorithm

For $1 \leq i \leq n$, we show that $W[B_i]$ and $\text{Pred}[B_i]$ store the weight and the predecessor of B_i in a maximum weighted chain ending at B_i. First, several simple invariants emerge from Algorithm 1. I_1: At any point,

Algorithm 1: MWC_Tolerance_Box_Order ($\mathcal{P}$)

Data: $r \in [0,1[$, $\mathcal{B}$ a set of n boxes, $\mathcal{P}$ an array with the $2n$ box corners

Result: W a vector of weights, with $W[B_n]$ the weight of the best chain in $\mathcal{B}$, Pred a vector containing the previous boxes in the chain

begin
  sort_on_x_coordinate($\mathcal{P}$);
  $\mathcal{A} \longleftarrow B_1$; $W[B_1] \longleftarrow 0$; $\text{Pred}[B_1] \longleftarrow null$; $O \longleftarrow \emptyset$;
  foreach $p \in \mathcal{P}$ *in ascending order on x-coordinate* **do**
    if p *is a lower corner (i.e.* $\exists B_i : p = l(B_i)$*)* **then**
      $O \longleftarrow O \cup \{B_i\}$;
      $\text{Pred}[B_i] \longleftarrow \underset{B_j \ll_r B_i, B_j \in \mathcal{A}}{\arg\max} \left(W[B_j] + \sum_{\alpha=1}^{2} \left|P_\alpha(B_i) \setminus P_\alpha(B_j)\right|\right)$;
      $W[B_i] \longleftarrow W[\text{Pred}[B_i]] + \sum_{\alpha=1}^{2} |P_\alpha(B_i) \setminus P_\alpha(\text{Pred}[B_i])|$;
    else /* p is an upper corner, *i.e.* $\exists B_i : p = u(B_i)$ */
      $O \longleftarrow O \setminus \{B_i\}$;
      foreach $B_k \in O$ *with* $B_i \ll_r B_k$ **do**
        $w_k \longleftarrow W[B_i] + \sum_{\alpha=1}^{2} |P_\alpha(B_k) \setminus P_\alpha(B_i)|$;
        if $w_k > W[B_k]$ **then**
          $W[B_k] \longleftarrow w_k$;
          $\text{Pred}[B_k] \longleftarrow B_i$;
      $B \longleftarrow \underset{u(B_j) < u(B_i), B_j \in \mathcal{A}}{\arg\max} (W[B_j])$;
      if $W[B_i] \geq W[B]$ *or* $|P_2(B_i)| > |P_2(B)|$ **then**
        $\mathcal{A} \longleftarrow \mathcal{A} \cup \{B_i\}$;
        foreach $B_k \in \mathcal{A}$ *with* $P_2(u(B_k)) > P_2(u(B_i))$ **do**
          if $W[B_k] < W[B_i]$ *and* $(|P_2(B_k)| < |P_2(B_i)|$ *or* $P_2(l(B_k)) > P_2(u(B_i)))$ **then**
            $\mathcal{A} \longleftarrow \mathcal{A} \setminus \{B_k\}$;
  traceback($\text{Pred}[B_n]$);
end

the set O contains all open boxes. I_2: Both $W[B_i]$ and $\text{Pred}[B_i]$ store their final values once $u(B_i)$ has been processed, since they are not altered after that point. I_3: Hence, at any point all active boxes (*i.e.* boxes in $\mathcal{A}$), which are closed boxes, satisfy I_2. For conciseness, as $W[B_i]$ and $\text{Pred}[B_i]$ are computed jointly, from now on we deal only with $W[B_i]$. Since potential predecessors of B_i are partitioned in $A_b(B_i)$ (Figure 2a) and $A_o(B_i)$ (Figure 2b), we will prove two invariants: I_4: partial optimality over $A_b(B_i)$ at lower corners, and I_5: optimality at upper corners.

I_4: partial optimality over $A_b(B_i)$ at lower corners. We show that after processing $l(B_i)$, $W[B_i]$ stores the weight of a maximum weighted chain ending at B_i with predecessor in $A_b(B_i)$. Given line 7, this is equivalent to showing that no better chain ending at B_i passes through a potential predecessor that does not belong to $\mathcal{A}$ at that point, which we prove by contradiction. While processing $l(B_i)$, $\mathcal{A}$ contains a subset of boxes in $A_b(B_i)$, but obviously none from $A_o(B_i)$. Let B be a closed box of $\mathcal{B} \setminus \mathcal{A}$ such that $B \ll_r B_i$ and $w(B_i) - w(B \cap B_i) + W[B] > W[B_i]$, in other words, B makes a better predecessor for B_i than those in $\mathcal{A}$. From $B \ll_r B_i$, we get

$$P_2(u(B)) - P_2(l(B_i)) \leq r \min(|P_2(B)|, |P_2(B_i)|) \,. \tag{5}$$

As only two possibilities exist for B not belonging to $\mathcal{A}$, we distinguish two exclusive cases.

B was not turned active when sweeping $u(B)$ (lines 16-18). B did not satisfy the condition on line 17. Let $B' := \underset{B_j \in \mathcal{A}: u(B_j) < u(B)}{\arg\max} (W[B_j])$. Our hypothesis means that $u(B') < u(B)$ and

$$\begin{aligned} W[B] &< W[B'] && (6)\\ |P_2(B)| &\leq \left|P_2(B')\right| \,. && (7) \end{aligned}$$

For B does not overlap B_i and $u(B') < u(B)$, we have B' does not overlap B_i on the x-axis. From $u(B') < u(B)$, we get $P_2(u(B')) < P_2(u(B))$; this with equations 5 and 7 yields

$$\begin{aligned} P_2(u(B')) - P_2(l(B_i)) &< P_2(u(B)) - P_2(l(B_i)) \\ &\leq r \min(|P_2(B)|, |P_2(B_i)|) \\ &\leq r \min(\left|P_2(B')\right|, |P_2(B_i)|) \,. \end{aligned} \tag{8}$$

Equation 8 and B' not overlapping B_i on the x-axis imply $B' \ll_r B_i$. Finally, from equations 6, 7, and $u(B') < u(B)$ we obtain:

$$W[B] + \sum_{\alpha=1}^{2} (w_\alpha(B_i) - w_\alpha(B_i \cap B)) \quad < \quad W[B'] + \sum_{\alpha=1}^{2} (w_\alpha(B_i) - w_\alpha(B_i \cap B')),$$

and thus B' makes a better predecessor for B_i than B, a contradiction.

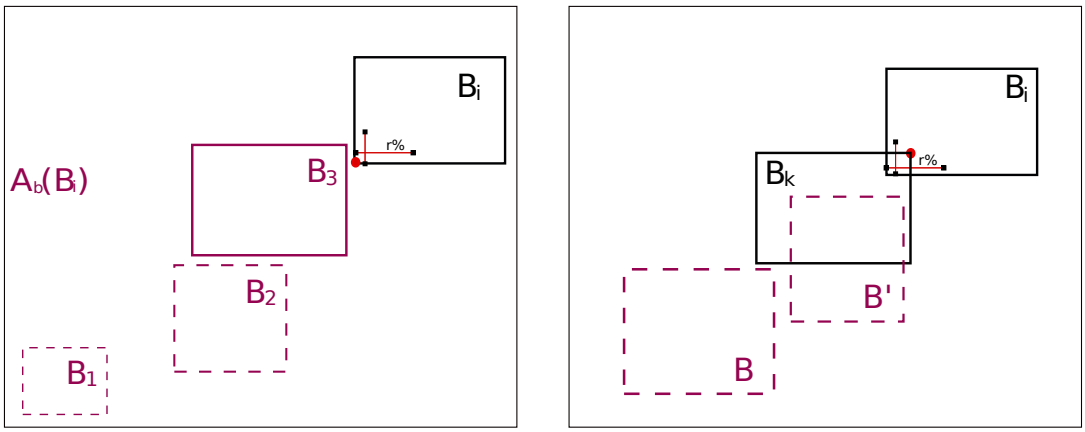
B was inactivated when sweeping $u(B_k)$ for some box B_k ending before $l(B_i)$ (lines 19-21). The hypothesis means that B was deleted from $\mathcal{A}$ for it satisfied $P_2(u(B_k)) < P_2(u(B))$, $W[B] < W[B_k]$, and at least one of the conditions (a) $|P_2(B)| < |P_2(B_k)|$ or (b) $P_2(u(B_k)) < P_2(l(B))$.

a) As above (see Eq. 8), from Equation 6, from $|P_2(B)| < |P_2(B_k)|$, and $P_2(u(B_k)) < P_2(u(B))$, we get

$$P_2(u(B_k)) - P_2(l(B_i)) \quad < \quad r \min(|P_2(B_k)|, |P_2(B_i)|) \,. \tag{9}$$

Moreover, as B_k does not overlap B_i on the x-axis, we obtain $B_k \ll_r B_i$. As $P_2(u(B_k)) < P_2(u(B))$, B_k and B do not overlap B_i on x-axis, and $W[B] < W[B_k]$, we finally derive

$$W[B] + \sum_{\alpha=1}^{2} |P_\alpha(B_i) \setminus P_\alpha(B)|) < W[B_k] + \sum_{\alpha=1}^{2} |P_\alpha(B_i) \setminus P_\alpha(B_k)|) \,. \tag{10}$$



(a) (b)

Figure 2: (a) When passing $l(B_i)$, Pred$[B_i]$ is a partial optimum on the set of possible predecessors of B_i lying in $A_b(B_i)$ (in the example, B_3). (b) When passing $u(B_k)$, Pred$[B_i]$ is a partial optimum on the set of possible predecessors of B_i from $A_b(B_i) \cup \{B \in A_o(B_i) / P_1(u(B)) < P_1(u(B_k))\}$.

b) By hypothesis, we know that $P_2(u(B_k)) < P_2(l(B)) < P_2(l(B_i))$, and since neither B_k nor B overlap B_i on the x-axis, we directly obtain $B_k \ll B_i$ ($B_i \cap B_k = \emptyset$). Thus, $W[B] < W[B_k]$ also implies Equation 10.

With either condition (a) or (b), B_k makes a better predecessor for B_i than B, a contradiction.

Finally, after processing $l(B_i)$, $W[B_i]$ stores the weight of a maximum weighted chain ending at B_i with predecessor in $A_b(B_i)$, which concludes the proof of I_4.

I_5: optimality at upper corners. We show that after processing $u(B_i)$, $W[B_i]$ stores $W(B_i)$ (a Maximum Weighted Chain with a predecessor in $A_b(B_i) \cup A_o(B_i)$). As all predecessors of B_i are closed, let us denote by B, the right most predecessor of B_i on the x-axis: $B := \arg\max_{B_j \ll_r B_i}(P_1(u(B_j)))$.

1. If $u(B) \in A_b(B_i)$ then all predecessors of B_i are contained in $A_b(B_i)$. Hence, this situation was handled when processing $l(B_i)$, and Invariant I_4 regarding the **partial optimality at lower corners**, ensures that $W[B_i]$ stores $W(B_i)$.

2. If $u(B) \in A_o(B_i)$, $W[B_i]$ has been correctly updated (lines 11-15), while B_i was open, when sweeping $u(B_k)$ for each box $B_k \in \mathcal{B}$ such that $B_k \ll_r B_i$ and $u(B_k) \in A_o(B_i)$.

Hence, all predecessors of B_i have been taken into account, and $W[B_i]$ stores $W(B_i)$. This concludes the proof of I_5, and closes the correctness proof.

4.3 Time and space analysis

Here, we detail the complexities of our sweep-line algorithm (Algorithm 1) and compare it to the dynamic programming algorithm (Algorithm DP) in term of running time.

Obviously, the sets O and $\mathcal{A}$ contain at most n boxes, and thus require together with arrays Pred$[.]$ and $W[.]$, $O(n)$ space. We use balanced binary search trees (BST) to store $\mathcal{A}$ and O, with boxes at the leaves ordered on $P_2(u(.))$, resp. $P_1(l(.))$. Hence, the amortised time needed for all insertions, deletions, and rebalancing is $O(n \log n)$ [9, chap. 6]. However, looking for the active boxes that can be deleted at each execution of the outer loop (lines 19-21) may force us to examine all boxes in $\mathcal{A}$. As this is the more complex operation in the outer loop, we obtain in the worst case an $O(n^2)$ time and $O(n)$ space complexity. Algorithm 1 maintains the subset of potential predecessors in $\mathcal{A}$ instead of searching through the whole box set as in Algorithm DP, which makes the practical difference.

The experimental running times observed when performing 694 whole genome pairwise comparisons (see Section 5) show that this optimisation yields significant improvements: Algorithm 1 needed ≈ 8*hours* to compute the 694 chains, while Algorithm DP took 3.5 days for the same computation. The improvement increases with the number of input fragments, and becomes considerable above $50,000$ fragments. In fact, below $50,000$ fragments, both Algorithm 1 and Algorithm DP take seconds, *i.e.* less than 1 minute. However, for n ranging from $50,000$ to $100,000$ Algorithm 1 still takes less than a minute, while Algorithm DP needs between 1 and 6 minutes to compute the chain. Moreover, the gap between the two algorithms is widening beyond the threshold of $100,000$ fragments. The following examples illustrate this statement: for $n = 144,685$, Algorithm DP takes ≈ 16 minutes, while Algorithm 1 only needs < 1 minute; for $n = 197,310$, Algorithm DP takes 34 minutes and Algorithm 1 less than 3 minutes; for $n = 307,852$, Algorithm DP needs 57 minutes and Algorithm 1 only 7 minutes. Finally, for $1,000,000$ fragments, Algorithm 1 needs ≈ 1 hour, while Algorithm DP ended after 12 hours. More details on the running times of Algorithm 1 can be found in Figure 3b.

5 Results

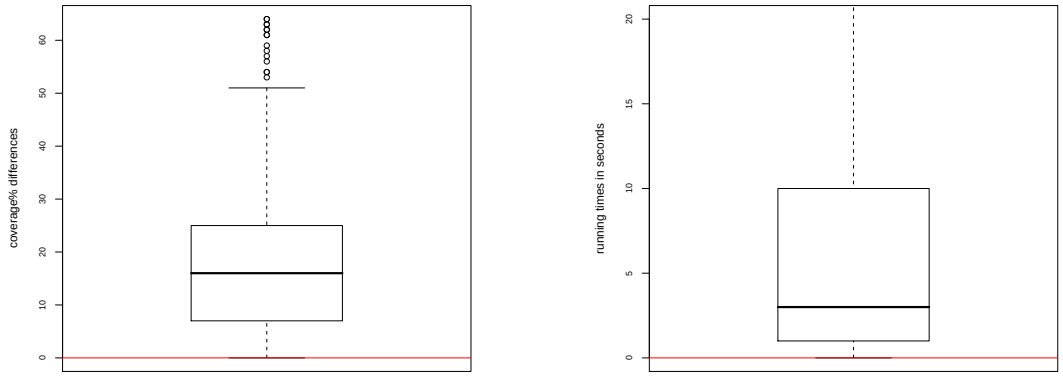
A first issue to examine is in which measure allowing for overlaps improves the chain weight (here, the genome coverage) when comparing genomes, and at which computational cost. To investigate this, we compared the running times and coverages (for a definition of the coverage see page 2) on 694 pairwise genome comparisons, obtained with Chainer [1] when overlaps are not allowed, and with Algorithm 1, implemented in our tool OC, when proportional overlaps are taken into account. As OC is not yet made publicly available, should anyone need to use it, please contact the authors.

Our comparison set consists in all pairwise genome comparisons of strains of the same bacteria (intra-species comparisons) as of Jan 2010: it comprises 346 different genomes from 88 species retrieved from Genome Reviews database [8]. First, we searched for local alignments between genome pairs using YASS with default parameters [12]. The output local alignments are the fragments given as input to the chaining step.

For this first experiment, we ran in parallel Chainer and Algorithm 1, with the weight of a fragment on each genome being the length of the aligned sequence. We authorised overlaps measuring up to 10% of the fragments' lengths ($r = 0.1$). Hence, the total chain weight, *i.e.* the sum of the chained fragments lengths minus the overlaps, computed by the chaining step gives the **genome coverage**, which we report as a percentage of the genome length. It is straightforward that, as both chaining algorithms provide an optimal solution in their setup, the coverage with overlaps is always larger than without overlaps.

Figure 3a plots for each genome pair the difference of coverage percentages between both algorithms: (Algorithm 1 coverage% - Chainer coverage%); a value of 10 means that chaining with overlaps covers 10% more of the genome than chaining without overlaps. The box-and-whiskers plot shows that, for this collection of bacterial genome pairs, the improvement has a median of 15% and reaches values up to 60%. Since bacterial genomes have a median length of 2.8 Mbp, a difference of one percent means at least 28 Kbp of additionally aligned sequences. Thus, one can make the following basic estimation. Knowing that in average 87% of these genomes are coding and bacterial genes are 1 Kbp long [16], 15% more coverage should involve 365 additional genes compared to a solution without overlaps; this is important from the biological perspective. Chainer takes < 1 s. in average and at most 17 s. We plot the running times of Algorithm 1 in Figure 3b: it stays below 10 seconds in 75% of the comparisons, and between 3 and 67 minutes in only 30 cases (those with a number of fragments ranging from $> 200,000$ to $1,000,000$). Thus, allowing for overlaps improves the coverage significantly at a reasonable cost.

A biological question regards the causes of overlaps. For example, when comparing the two strains *CP*000046 and *BA*000018 of *S. aureus*, the classical chaining results in a coverage of 65%, where Algorithm 1 yields a 94% coverage. In fact, the chain obtained without allowing overlaps is interrupted by 17 holes of more than 10 kbp each. For 14 of these holes, at least one large fragment (average size 37 kbp) was not included in the chain, because of an overlap with an adjacent fragment on one or both genomes. All overlaps measure between 1 bp and 1.8 kbp in length (average at 218 bp). This example shows that overlaps' lengths cannot be easily bounded by a constant, thus revealing the necessity of using proportional overlaps. If short overlaps are mostly due to random equality between base pairs on fragment ends, we ob-



(a) (b)

Figure 3: (a) Differences in coverage obtained on 694 bacterial genome comparisons between our algorithm and the classical chaining, presented as a box-and-whiskers plot: in 50% of the cases, allowing for proportional overlaps increases the coverage% with more than 15%. (b) Running times in seconds of our algorithm: in 75% of the cases the algorithm needs less than 10 seconds.

served that large overlaps are often caused by variable tandem repeat structures that differ in number of copies between the strains (see a toy example in Introduction on page 1). Correctly aligning such structures without breaking the region in two overlapping fragments requires a more general alignment model and specific algorithms [3].

Another issue to investigate is the robustness of results with respect to the ratio of allowed overlaps. For this, we examine the results obtained with three different overlap ratios: 5% of the fragments' lengths ($r = 0.05$), 10% ($r = 0.1$), and 15% ($r = 0.15$) on the same set composed of 694 pairwise genome comparisons. Finally, we study the advantage due to the use of proportional overlaps compared to fixed overlaps. For this purpose, we modified our algorithm in order to allow a fixed maximal overlap between fragments, instead of a ratio of their lengths. This means that an overlap between two fragments is allowed if it is below the fixed maximal overlap size and does not entirely cover one of the fragments. We thus tested four maximal overlap thresholds: 10 bp, 100 bp, 1 Kbp, and 10 Kbp.

The results for different overlap ratios, and fixed overlap thresholds are summarised in figures 4, 5, and 6, as averages on each one of the 88 species, for eight configurations of our program OC. These configurations correspond to a version without overlaps (OC 0%), a version with proportional overlaps for three overlap ratios (OC 5%, OC 10%, and OC 15%) and a version with fixed overlaps configured with four overlap thresholds (OC 10 bp, OC 100 bp, OC 1 Kbp, and OC 10 Kbp). In figures 4, 5, and 6, the results for the 88 species are partitioned as follows: 30 species in the first figure, 28 in the second one, and 30 in the third one, for the sake of legibility. From the first figure to the third one, species are sorted on the average coverage% given by the version of the algorithm without overlaps (OC 0%).

Figures 4a, 5a, and 6a show how the coverage percentages vary among the eight chaining configurations. Here, we define the **identity%** as the ratio between the number of pairs of identical positions in the covered regions and the genomes' lengths. Thus, the identity% over the coverage% (also computed as a ratio of the genomes' lengths) gives a quality measure of the covered regions. As overlap chaining covers additional genomic regions compared to overlap free chaining, it is important to assess the alignment quality of these regions. We use the following measure: assume an overlap chaining configuration yields a coverage percentage of $X\%$ and an identity percentage of $Y\%$, while the overlap free chaining obtains $X_0\%$ and $Y_0\%$, respectively, then we compute quality variation as $(X/Y - X_0/Y_0)$. Figures 4b, 5b, and 6b report the quality variation of each overlap chaining configuration compared to overlap free chaining. We chose to include the results for the whole data set to illustrate the generality of our conclusions, at least for

bacterial genomes.

These figures must be considered as a whole, as it is important to understand that the following observations remain valid on all species, from lower to higher coverages.

- As already noticed in Figure 3a, overlaps improve significantly the coverages compared to overlap free chaining (mostly for proportional overlaps). This can be observed when examining the plots of the six configurations of OC compared to the version without overlaps, OC 0%. This trend is perfectly depicted in the zoom for *Burkholderia mallei* species in Figure 4a.

- Different overlap ratios generate very small variation among coverages: OC 5%, OC 10%, and OC 15% have very similar coverage values for all species. Thus, the total chain weight is robust to changes in the overlap ratio. Moreover, in most of the cases, these three configurations with proportional overlaps obtain coverages as high as the best of the other five configurations of OC. Clearly, the higher the maximal overlap ratio, the better the coverages are. However, if for small overlap ratios we observe a certain improvement, *i.e.* from 5 to 10, and below 5% (results not presented in this paper), above 10% improvements are not significant. Therefore, 10% seems to be a fair choice for the overlap ratio for bacterial comparisons, as overlaps in our dataset rarely reach 10% of fragments lengths.

- When dealing with fixed overlaps, one needs to set a relatively high threshold (10 Kbp) to get a slightly better coverage, especially when compared to the results obtained with proportional overlaps. OC 10 bp, OC 100 bp, and OC 1 Kbp systematically yield lower coverage than OC with proportional overlaps. Indeed, overlap sizes are widely distributed. Large overlaps penalize most the total coverage as they prevent from incorporating large fragments into the chain. Hence, fixed overlap chaining can cope with such overlaps only if a high threshold is chosen, *i.e.* 10 Kbp. Moreover, the performance of fixed overlap solutions vary among genome pairs, making them less robust. Thus, fixed overlap thresholds are difficult to set. Comparatively, proportional overlap chaining proves more robust because of proportionality to fragments lengths.

- Importantly, the quality variation compared to overlap free chaining (OC 0%) are quite small: between -2 and 2%, except for five cases that exhibit extremely low coverages (less than 5% coverage). Moreover, regarding quality variation, proportional overlaps configurations never yield the worst result among overlap chaining solutions. Thus, the alignment quality of covered regions remains stable when allowing proportional overlaps.

Finally Figure 7 reports, for each configuration of OC, the number of species whose computed chain reaches that coverage value in average over all comparisons within that species. The red square at coordinates $(60, 65)$ means that 65 species obtained a coverage of 60% with OC 100 bps threshold. All curves are superimposed in the range $[0, 30]$ of coverage%. Beyond 30% coverage, the curves of proportional overlaps configurations are superimposed, again illustrating the robustness with respect to the overlap ratio. Moreover, above 30%, overlap free and fixed overlap chaining, except that with 10 Kbp threshold, generally perform worse than proportional overlap solutions.

6 Conclusion

To fulfil new needs in computational biology, we extended the classical framework of Maximum Weighted Chain by authorizing overlaps between fragments in the computed chain, and formalised the Maximum Weighted Chain with Proportional Length Overlaps problem where overlaps are proportional to the fragment lengths. Difficulties arise from the fact that the weights of overlaps are deduced from the chain weight. We exhibited the first two algorithms for this problem, which both solve it in quadratic time in function of the number of fragments. Experiments on real data sets show that our sweep line algorithm outperforms the truly quadratic dynamic programming solution in practice. However, the study of the average time complexity of the sweep line algorithm remains open. By comparing the genome coverage obtained by different configurations of our program, OverlapChainer (OC), when applied to intra-species, bacterial comparisons, we observe that: i/ overlaps improve significantly the coverage of genomes (median of 15%),

while maintaining the same quality level, ii/ results obtained with proportional overlaps are robust with respect to different overlap ratios, contrarily to fixed overlaps. This robustness may be explained by the limited overlap size in bacterial genomes comparisons. In bacterial case, a ratio of 0.1 yield good results on a wide panel of 694 pairwise genome comparisons.

Investigating the performance of OverlapChainer on eukaryotic genomes is an interesting perspective. Our algorithms can be extended for multiple genome comparisons, however future researchs should also consider this case in practice. Moreover, the optimal complexity of chaining with proportional overlaps remains open.

Acknowledgements: This work, AM and RU, were supported by the French National Research Agency (CoCoGen project) [BLAN07-1_185484]. AM and RU were also supported respectively by CNRS funding and a grant from the Ministry of Higher Education and Research of France.

References

[1] Mohamed Ibrahim Abouelhoda and Enno Ohlebusch. Chaining algorithms for multiple genome comparison. *J. of Discrete Algorithms*, 3:321–341, 2005.

[2] Bastien Boussau and Vincent Daubin. Genomes as documents of evolutionary history. *Trends in Ecology & Evolution*, 25(4):224 – 232, 2010.

[3] Sèverine Bérard and Eric Rivals. Comparison of minisatellites. *J. Comp. Biol.*, 10(3-4):357–372, 2003.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[5] Stefan Felsner, Rudolf Muller, and Lorenz Wernisch. Trapezoid graphs and generalizations, geometry and algorithms. *Disc. App. Math.*, 74:13–32, 1995.

[6] Michael Hohl, Stefan Kurtz, and Enno Ohlebusch. Efficient multiple genome alignment. *Bioinformatics*, 18(S1):S312–320, 2002.

[7] Andrew P. Jackson, John A. Gamble, Tim Yeomans, Gary P. Moran, David Saunders, David Harris, Martin Aslett, Jamie F. Barrell, Geraldine Butler, Francesco Citiulo, David C. Coleman, Piet W.J. de Groot, Tim J. Goodwin, Michael A. Quail, Jacqueline McQuillan, Carol A. Munro, Arnab Pain, Russell T. Poulter, Marie-Adèle Rajandream, Hubert Renauld, Martin J. Spiering, Adrian Tivey, Neil A.R. Gow, Barclay Barrell, Derek J. Sullivan, and Matthew Berriman. Comparative genomics of the fungal pathogens Candida dubliniensis and Candida albicans. *Genome Res.*, 19(12):2231–2244, 2009.

[8] Paul Kersey, Lawrence Bower, Lorna Morris, Alan Horne, Robert Petryszak, Carola Kanz, Alexander Kanapin, Ujjwal Das, Karine Michoud, Isabelle Phan, Alexandre Gattiker, Tamara Kulikova, Nadeem Faruque, Karyn Duggan, Peter Mclaren, Britt Reimholz, Laurent Duret, Simon Penel, Ingmar Reuter, and Rolf Apweiler. Integr8 and Genome Reviews: integrated views of complete genomes and proteomes. *Nucleic Acids Res.*, 33(S1):D297–302, 2005.

[9] D.E. Knuth. *The Art of Computer Programming*, volume 3 / Sorting and Searching. Addison-Wesley, 1998.

[10] Claire Lemaitre and Marie-France Sagot. A small trip in the untranquil world of genomes. *Theor. Comp. Sci.*, 395:171–192, 2008.

[11] Gene Myers and Webb Miller. Chaining multiple-alignment fragments in sub-quadratic time. In *Proc. of the sixth annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 38–47, 1995.

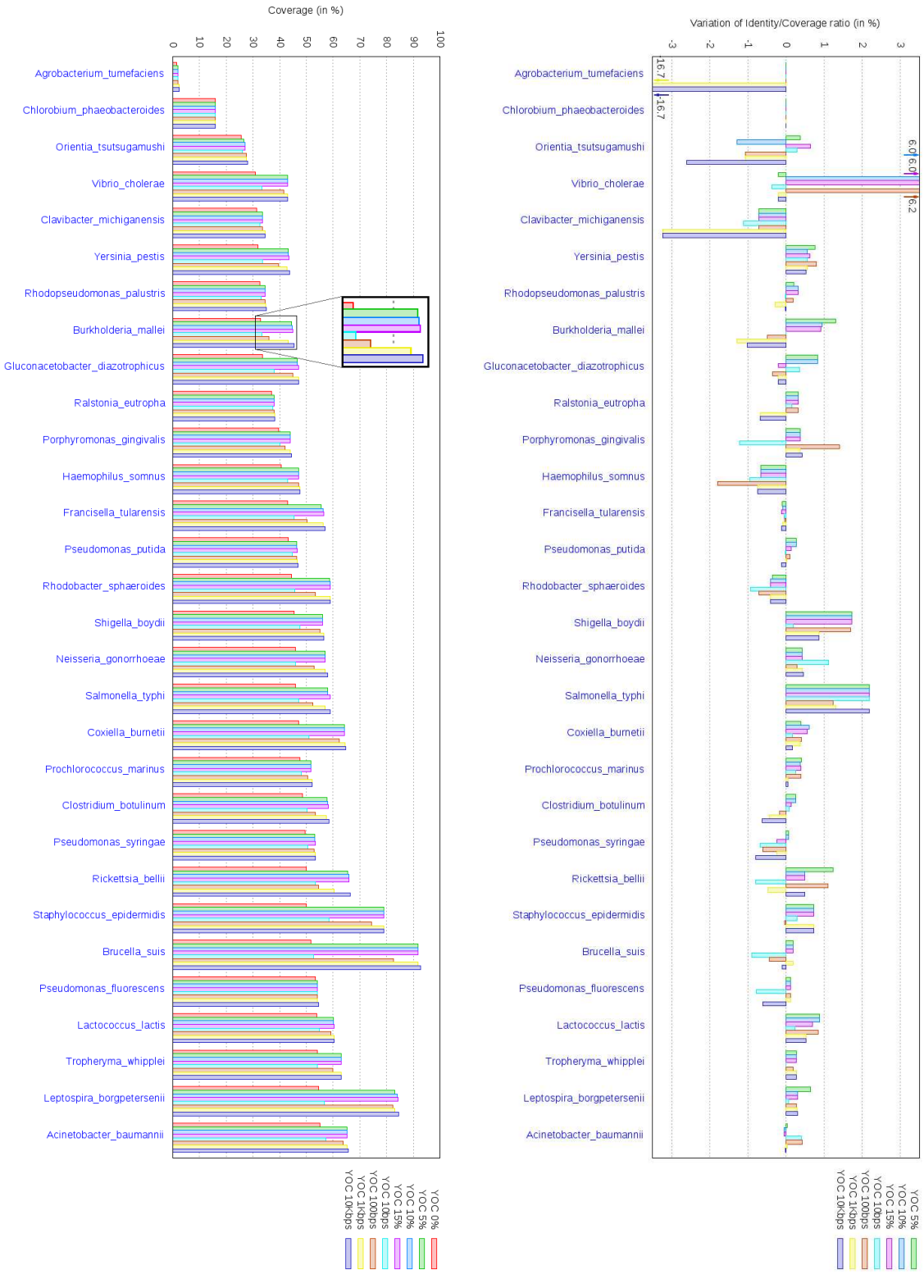
[12] Laurent Noé and Gregory Kucherov. YASS: enhancing the sensitivity of DNA similarity search. *Nucleic Acids Res.*, 33(S2):W540–543, 2005.

[13] Davide Serruto and Rino Rappuoli. Post-genomic vaccine development. *FEBS Letters*, 580(12):2985 – 2992, 2006.

[14] Tetsuo Shibuya and Igor Kurochkin. Match chaining algorithms for cDNA mapping. In *Proc. Workshop on Algorithms in Bioinformatics (WABI)*, volume 2812 of *Lecture Notes in Computer Science*, pages 462–475. Springer, 2003.

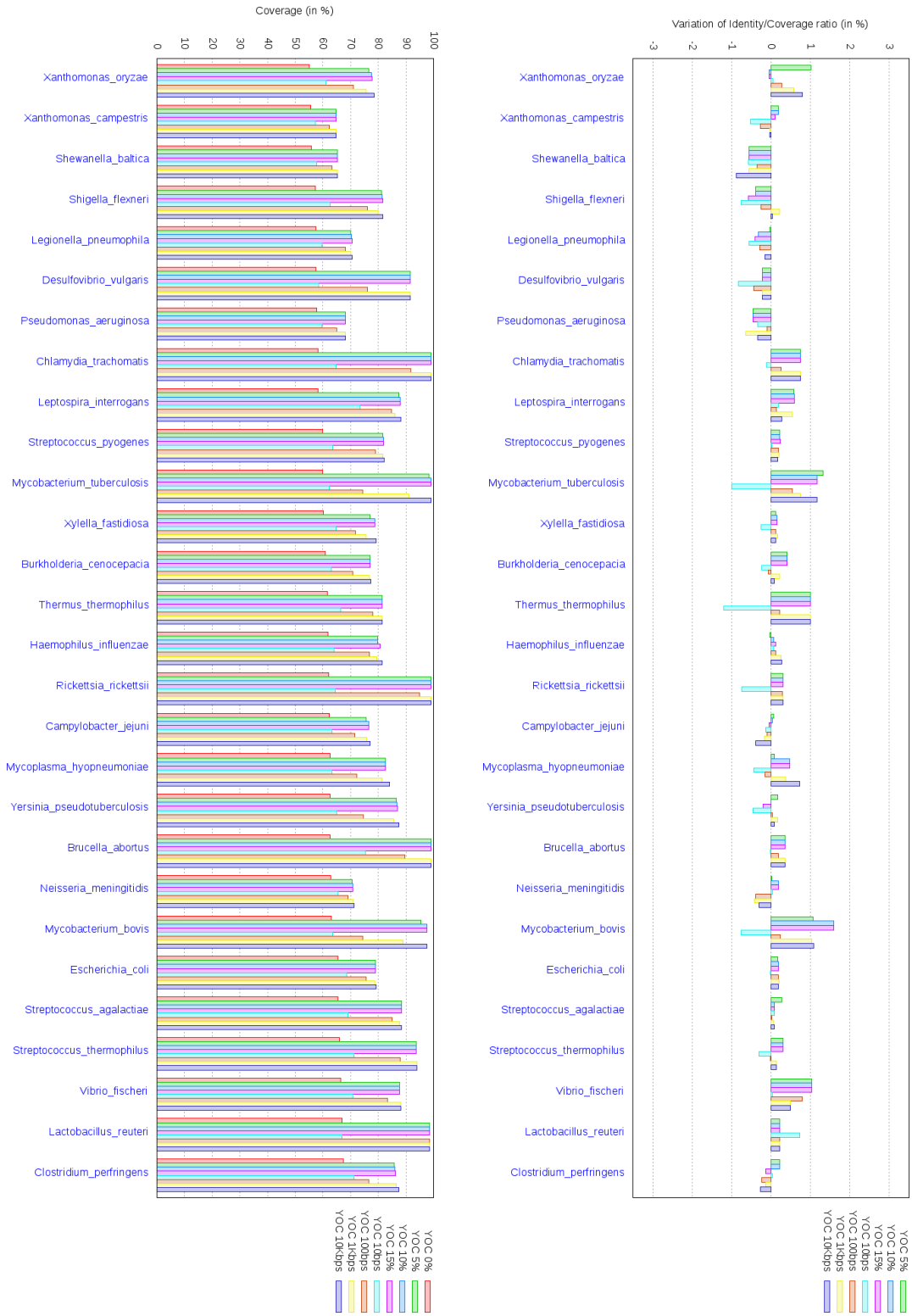
[15] Raluca Uricaru, Célia Michotey, Laurent Noé, Hélène Chiapello, and Eric Rivals. Improved sensitivity and reliability of anchor based genome alignment. In Irena Rusu et Eric Rivals, editor, *Actes des Journées Ouvertes Biologie Informatique Mathématiques (JOBIM)*, pages 31–36, 2009.

[16] Lin Xu, Hong Chen, Xiaohua Hu, Rongmei Zhang, Ze Zhang, and Z. W. Luo. Average gene length is highly conserved in prokaryotes and eukaryotes and diverges only between the two kingdoms. *Mol. Biol. Evol.*, 23(6):1107–8, 2006.



(a) (b)

Figure 4: Average results for each species among 30 species, given by eight configurations of our program OC. (a) Variation of coverage%. Plots mainly show the robustness of results for different overlap ratios compared to fixed overlaps. (b) Variation of the ratios between identity% and coverage% relatively to OC 0%, showing that allowing overlaps does not imply covering lower quality regions.



(a) (b)

Figure 5: Similar to Figure 4, average results for each species among another 28 species, given by the eight configurations of our program, OC.

(a) (b)

Figure 6: Similar to Figure 4 and Figure 5, average results for each species among another 30 species, given by the eight configurations of our program, OC.

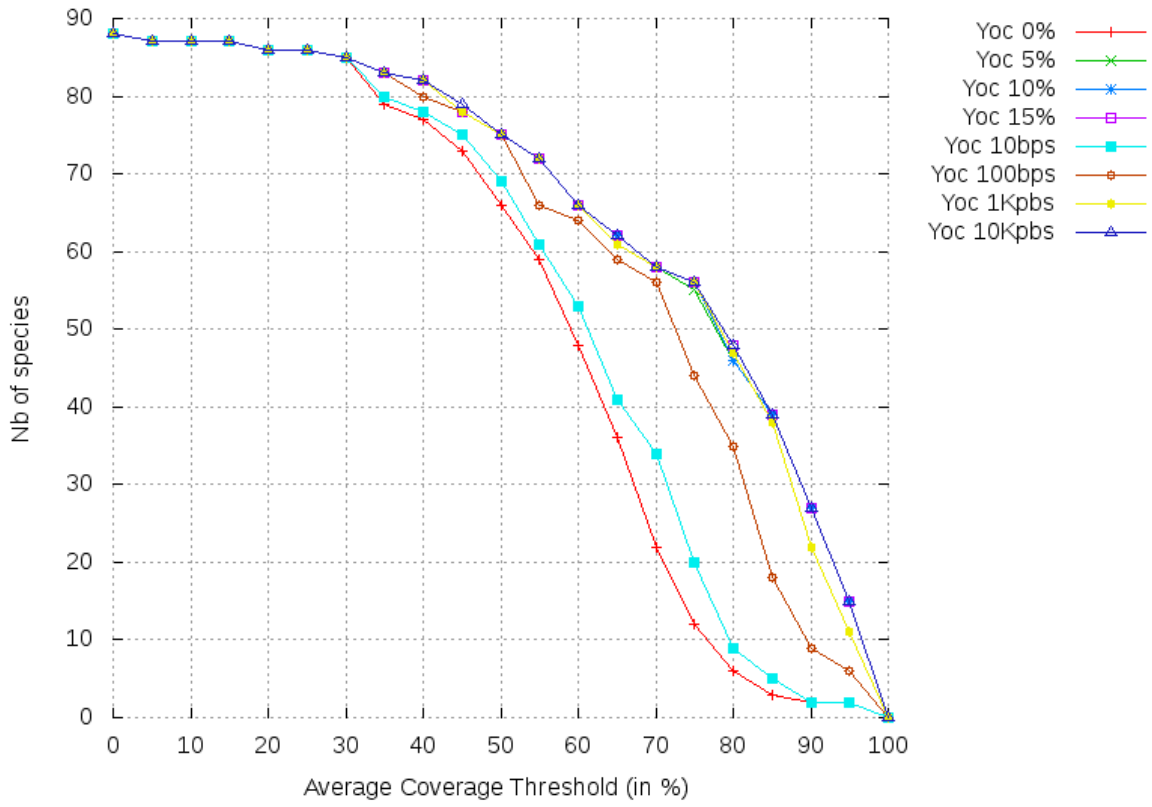


Figure 7: The variation of the number of species for each coverage% threshold, for the eight configurations of our program, OC.