# Inferring Positional Homologs with Common Intervals of Sequences

Guillaume Blin, Annie Château, Cedric Chauve, Yannick Gingras

# Inferring positional homologs with common intervals of sequences*

Guillaume Blin[1], Annie Chateau[2,3], Cedric Chauve[2,3,4], Yannick Gingras[3]

[1] IGM-LabInfo - UMR CNRS 8049, Université Marne-la-Vallée
5 bd. Descartes 77454 Marne-la-Vallée Cedex 2, France
gblin@univ-mlv.fr
[2] LaCIM, Université du Québec À Montréal
CP 8888, Succ. Centre-Ville, H3C 3P8, Montréal (QC), Canada
[chateau,chauve]@lacim.uqam.ca
[3] CGL, Université du Québec À Montréal
ygingras@ygingras.net
[4] Department of Mathematics, Simon Fraser University
8888 University Drive, V5A 1S6, Burnaby (BC), Canada

**Abstract.** Inferring orthologous and paralogous genes is an important problem in whole genomes comparisons, both for functional or evolutionary studies. In this paper, we introduce a new approach for inferring candidate pairs of orthologous genes between genomes, also called positional homologs, based on the conservation of the genomic context. We consider genomes represented by their gene order – i.e. sequences of signed integers – and common intervals of these sequences as the anchors of the final gene matching. We show that the natural combinatorial problem of computing a maximal cover of the two genomes using the minimum number of common intervals is NP-complete and we give a simple heuristic for this problem. We illustrate the effectiveness of this first approach using common intervals of sequences on two datasets, respectively 8 $\gamma$-proteobacterial genomes and the human and mouse whole genomes.

## 1 Introduction

In the comparison of two genomes, a first natural task is to compare the sequences of their genes (nucleotides or amino-acids) in order to identify homologous genes, that are pairs of genes whose sequence similarity is strong enough to suggest a common ancestral gene. However, due to the evolutionary mechanisms that shape genomes – rearrangements, duplications, both of genomic segments or of whole genomes, gene losses or lateral transfers – homologous relations are often ambiguous and do not induce clear one-to-one relationships between genes of the two genomes [8]. Instead, non-trivial gene families, occurring in several positions in one or both genomes, make difficult to distinguish pairs of orthologous genes. Several methods have been proposed that use the genomic context to distinguish

* Work supported by grants from Génome Québec, NSERC and Coopération Franco-Québecoise

pairs of genes, called *positional homologs*, that are good candidates to be pairs of orthologous genes [8]. In this paper we describe a new approach using the genomic context, based on the notion of *common intervals of two sequences*.

There are two main approaches for inferring positional homologs using the genomic context. In the *exemplar* approach, introduced by Sankoff [21], for every non-trivial gene family, all but one copy in each genome are deleted. The pair of genes that is conserved for each family is called a pair of *ancestral homologs*. The *gene matching* approach is more general as it allows to conserve more than one copy of a gene family and seeks for an unambiguous one-to-one matching between these copies [12]. Both approaches have in common that they lead to represent the two compared genomes by two permutations. These permutations can then be used in several contexts: computing a genomic distance [23], phylogenetic reconstruction [5] or proposing candidate pairs of orthologous genes [12, 14].

In both families of methods – exemplar and gene matching –, several combinatorial criteria have been considered in order to clear ambiguous homology relations between genes of the same family. A classical approach is to try to find the resulting pair of permutations that will minimize a given genomic distance between them, like the reversal distance [12, 21, 23, 24] or the reversal and translocation distance [14]. Another approach looks for the pair of permutations that maximizes the conservation of some combinatorial structures in the two resulting permutations, like adjacencies [5] or common intervals of permutations [6]. However, it is important to note that, up to date, all these problems have been shown to be NP-hard [3, 7, 11]

In the present work, we are interested in computing a gene matching with a method that is inspired by algorithms for global alignment of long genomic sequences. Indeed, given two genomes represented by two sequences of integers – where each integer labels a gene family – a gene matching is nothing else than a global alignment of these two sequences. Here we propose to use the occurrences of common intervals of these two sequences, which are segments having the same gene content with no constraint on gene order or multiplicity, as *anchors* for the final gene matching. Then we recursively match common intervals, using a simple heuristic for the MINIMUM INEXTENSIBLE BOX COVERING problem – which is a NP-hard problem (see Appendix) – until all possible pairs of positional homologs have been chosen. Hence, one of the originality of our method is that it does not try to compute the pair of permutations that is optimal for a given combinatorial criterion, but rely on a greedy approach of recursively matching genomic segments having a common combinatorial structure.

Our method can be seen as an extension of a previous gene matching algorithm, that iteratively matches longest common substrings (LCS) of the two genomes – such LCS representing perfect colinear segments of genes – [23, 5], but using a less constrained notion of conserved structure between genomes (common intervals vs. LCS). Common intervals of sequences are interesting with respect to LCS for two reasons. From the biological point of view, they are more adapted to detect sequences of genes that evolved from a common ancestral sequence with events like segmental or tandem duplications, or local rearrangements. From the

algorithmical point of view, the set of all common intervals of two sequences can be computed in quadratic time [22]. Note however that gene deletions or insertions, that are natural evolutionary events, can destroy common intervals, and we discuss this issue in Section 4.

In Section 2, we describe precisely our method to compute a gene matching. In Section 3, we present two experimental studies of our method. We first consider 8 genomes of $\gamma$-proteobacteria, and we compare the results of our method with the LCS method of [5]; this experiment raises interesting facts about the properties of these two notions of conserved structures in the comparison of more than two genomes. Next, we consider the human and mouse genomes and we compare our results with the results obtained with the MSOAR method that tries to find the most parsimonious pairs of permutations for the reversals and translocations distance [14]. In Section 4, we describe several ways to improve our initial approach.

## 2 The new method

A *matching* between two sequences of integers $s_1$ and $s_2$, where each of these integers represents a family of genetic markers – genes here –, is a one-to-one mapping between a subset of the first sequence and a subset of the second sequence, such that pairs of mapped markers belong to the same family [5]. In the method we propose, we compute a matching in four main steps. First, we define candidate anchors for the alignment between the genomes using common intervals of sequences. We briefly recall in the next paragraph some definitions and properties of common intervals of sequences. Then, we exclude anchors that do not exhibit enough structure to produce an accurate matching, using some reasonable selection criteria. Finally we extract from the remaining anchors a consistent subset, and apply the method recursively in each anchor, until we are left only with anchors containing markers of a single family, then we match markers in these boxes. The final result is a matching between the two considered sequences that describe one-to-one correspondences between the genes of the two corresponding genomes.

### 2.1 First step: finding the anchors

Given a sequence of integers $A$ representing a genome, the alphabet $\Sigma$ of its gene content, and a subset $S$ of $\Sigma$, we define a *location* of $S$ in $A$ as a substring of $A$ that is a word on $S$. Hence, from a genomic point of view, a location of $S$ in $A$ represents a contiguous region in $A$ with gene content exactly $S$. The location is *maximal* if this substring cannot be extended on the right or on the left, meaning that the contiguous characters are not in $S$. A *factor* between two genomes is a subset of $\Sigma$ which has at least one location in each genome. Note that, for a given factor, there can be several locations in the same genome.

In the following, a *box* will refer to a set of two maximal locations, one in each genome, for a factor $S$, that is called the *alphabet* of the box[1]. A box is said to be *trivial* if its alphabet has cardinality 1. We represent a box by the quadruplet of the coordinates of the two corresponding locations.

Example: $S = \{7, 8, 9, 10, 11\}$ is a factor between $A$ and $B$, with one location in $A$ and two in $B$, which gives two boxes for $S$, $(5, 10, 4, 8)$ and $(5, 10, 12, 16)$.

Genome $A$: 1 2 3 12 [11 8 7 9 9 10] 4 3 1 1 5
Genome $B$: 6 1 4 [8 9 10 11 7] 2 13 13 [7 8 9 10 11]

The first step of our method consists in the computation of all the boxes between the two considered genomes, that we use as the basic structures to define the final matching. To compute the set of all possible boxes, we use the quadratic-time algorithm of Schmidt and Stoye [22].

### 2.2 Second step: filtering the set of boxes

Our experiments with the LCS method showed that a significant number of false positives matched gene pairs (roughly speaking, pairs of matched genes that contain two different genes according to gene names in the Uniprot database, see Section 3 for more details) was removed when LCS of short length were discarded from the analysis. This suggests that subsequences of the two genomes that do not exhibit a strong common combinatorial structure (here measured in terms of the length of the LCS) are more likely to produce wrong pairs of matched genes.

This is why we decided to introduce a similar feature in our method, that discards putative anchors, here boxes, that do not exhibit enough combinatorial structure. For LCS, the natural parameter that defines such a structure is the length, due to the conservation of the order in a LCS. However this is not the case with boxes, due to the less constrained structure that defines them, an issue that we discuss in Section 4.

In this first investigation of using common intervals of sequences that we present in this work, we chose to use a simple geometrical criterion to exclude boxes, that is the length of the smaller location for a given box, called the *minimum side* of the box. The intuition for this choice is that this parameter extends naturally the criterion of length that we used with the LCS method. This filtering step, with the simple criterion we consider, can be done in linear-time in the number of boxes.

We show in Section 3 some experiments using a dataset of 8 bacterial genomes that illustrate the impact of this filtering step on the resulting matchings.

### 2.3 Third step: extracting a consistent subset of candidate boxes

The third step of the method we propose consists in the computation of the gene matching by a recursive process, that takes as inputs the boxes that were not discarded during the second step.

[1] Box are called *common intervals* in [22].

Before describing this process, we define some combinatorial notions about sets of boxes:

- Intuitively, a box $(x_1, x_2, y_1, y_2)$ defines a rectangle in the 2D plane, defined by the points $(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)$.
- Two boxes are said to be *compatible* if there exists no vertical or horizontal line that can cross both boxes at once. Formally, $B = (x_1, x_2, y_1, y_2)$ and $B' = (x'_1, x'_2, y'_1, y'_2)$ are compatible if $[x_1, x_2] \bigcap [x'_1, x'_2] = \emptyset$ and $[y_1, y_2] \bigcap [y'_1, y'_2] = \emptyset$.
- A box $B$ id said to be *enclosed* in a box $B'$ if the rectangle $B$ is completely included in the rectangle $B'$. Formally, $B = (x_1, x_2, y_1, y_2)$ is enclosed in $B' = (x'_1, x'_2, y'_1, y'_2)$ if $[x_1, x_2] \subseteq [x'_1, x'_2]$ and $[y_1, y_2] \subseteq [y'_1, y'_2]$.
- Given a set $\mathcal{B}\{B_1, \ldots, B_n\}$ of boxes, a subset $\mathcal{B}'$ of $\mathcal{B}$ of boxes that are pairwise compatible is said to be *inextensible* if every box of $\mathcal{B}$ that does not belong to $\mathcal{B}'$ is not compatible with at least one box of $\mathcal{B}'$.

The principle of this third step is to select a subset $\mathcal{B}'$ of boxes that is inextensible and of minimum cardinality with respect to this property of being inextensible, then to recursively repeat this process inside each box of $\mathcal{B}'$.

To extract the set $\mathcal{B}'$ of boxes, we consider the MINIMUM INEXTENSIBLE BOX COVERING (MIBC) optimization problem: given a set of $n$ boxes $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$, find a subset$\mathcal{B}' \subseteq \mathcal{B}$ of minimum cardinality such that (1) any pair $(B_i, B_j)$ of boxes of $\mathcal{B}'$ is a pair of compatible boxes, and (2) $\mathcal{B}'$ is inextensible. Note that this problem is a variant of the MAXIMUM COMMON STRING PARTITION problem (MCSP) that occurs naturally in computing a gene matching that minimizes the number of breakpoints in the resulting permutations [3, 12]. This is the main reason that led us to introducing this problem for computing gene matchings, as we think it is a very natural way to compute recursively a gene matching from a set of boxes. However, the problem MCSP is NP-hard [16, 3] and it is then not surprising that the same result holds for the problem MIBC (the proof is given in Appendix).

**Theorem 1.** *The* MINIMUM INEXTENSIBLE BOX COVERING *problem is* NP*-hard.*

This hardness result leads us to develop a simple greedy heuristic, inspired from the MIBC problem, that selects an inextensible set of pairwise compatible boxes, by iteratively selecting the box with maximal area.

```
MIBC_heuristic(Input: set of boxes B)
   Create an empty set of boxes B'
   While (B is not empty)
      Select the biggest box Bi, in terms of area, in B
      Add Bi to B'
      Remove every box Bj of B which is incompatible with Bi
   Return B'
```

This heuristic procedure is used recursively until only trivial boxes are left.

```
MIBC_main(Input: set of boxes B)
   Let M=MIBC(B)
   While (M contains at least one non-trivial box)
      For (every non trivial box Bi of M)
         Remove Bi from M
         Let B' be the set of all boxes of B enclosed in Bi
         Add MIBC(B') to M
   Return M
```

The time complexity of this step is polynomial in the number of boxes given in input, which is itself quadratic in the size of the two considered genomes.

### 2.4 Fourth step: matching genes in trivial boxes

Once the third step is completed, we obtain a sequence of trivial boxes, that is boxes of alphabet of size 1, that forms the core of the final matching. Indeed, for all squared trivial boxes of side length 1, that are boxes of area 1, we add the pair of corresponding genes to the final matching.

The case of boxes such that more than two genes are involved is more problematic, as there are several ways to match the genes of such boxes. In the present work, we used the naive strategy that matches genes in increasing order of their positions in the two sequences representing the two genomes. We discuss briefly in Section 3 the influence of this strategy and, in Section 4, we describe several strategies to improve final matching inside the trivial boxes.

## 3 Experimental results

We implemented our method in a software called CIGAL (Common Intervals Global ALigner), and we now discuss two experimental studies on two datasets: first 8 genomes of $\gamma$-proteobacteria, then the human and mouse genomes.

### 3.1 Bacterial genomes

We considered the genomes of the following organisms, that span a wide spectrum in the phylogeny of $\gamma$-proteobacteria [2]:

- *Buchnera aphidicola APS* (GenBank accession number NC_002528),
- *Escherichia coli K12* (NC_000913),
- *Haemophilus influenzae Rd* (NC_000907),
- *Pasteurella multocida Pm70* (NC_002663),
- *Pseudomonas aeruginosa PA01* (NC_002516),
- *Salmonella typhimurium LT2* (NC_003197),
- *Xylella fastidiosa 9a5c* (NC_002488),
- *Yersinia pestis CO_92* (NC_003143).

We computed gene families as described in [5]. For each of the 28 pairs of genomes, we computed 6 different gene matchings: first three matchings using the method described in Section 2, respectively with no filtering (the corresponding matching is denoted CI1), with filtering boxes of minimum side 1 (CI2) and 2 (CI3), then three matchings using the LCS method of [5], respectively with no filtering (LCS1), with filtering LCS of length 1 (LCS2) and 2 (LCS3).

We considered two ways to assess the quality of the obtained gene matchings and compare the two methods:

- To assess the accuracy of the matching between genes, we compared the name of coding gene as given by the database UniProt [1], and we defined a gene pair as a *true positive* if both genes have the same name or synonymous names, a *false positive* if the names are different and an *unknown pair* if one of the two genes is not present in UniProt.
- To assess the internal consistency of both methods, we considered the 28 pairwise matchings between pairs of genomes as a graph, called the *combined matchings graph*, whose vertices are the genes of the 8 genomes and edges are given by the matchings. We then computed the proportion of the connected components of this graph that contain at least two genes of a same genome – we call such component *inconsistent components* and components with at most one gene of each genome *consistent components*. Indeed such a situation can be seen as inconsistent with respect to the goal of inferring positional homologs that are candidates to be orthologous genes.
- Subsequently we classified consistent components into two categories: *perfect*, if all genes in a component have the same name, and *imperfect* if at least two genes have different names (note that we discarded components with genes that are not present in Uniprot). Finally, we studied the distribution of perfect components with respect to their size.

The results are given in Tables 1 and 2 below.

| | LCS1 | LCS2 | LCS3 | CI1 | CI2 | CI3 |
|---|---|---|---|---|---|---|
| True positives | 14548 | 13635 | 10729 | 14954 | 15909 | 17180 |
| False positives | 821 | 736 | 596 | 1114 | 1661 | 2433 |
| Unknown pairs | 7240 | 5339 | 3558 | 8078 | 9121 | 11030 |
| Number of components | 3439 | 3850 | 3606 | 3539 | 3408 | 3480 |
| Consistent components | 2907 | 3704 | 3537 | 3117 | 3147 | 3382 |
| Ratio of consistent components | 0.85 | 0.96 | 0.98 | 0.88 | 0.92 | 0.97 |
| Number of perfect components | 1531 | 1723 | 1538 | 1628 | 1817 | 1962 |
| Ratio perfect/consistent | 0.53 | 0.47 | 0.43 | 0.52 | 0.58 | 0.58 |
| Number of imperfect components | 252 | 240 | 190 | 320 | 515 | 687 |
| Ratio imperfect/consistent | 0.09 | 0.06 | 0.05 | 0.1 | 0.16 | 0.2 |

**Table 1.** Quality of gene pairs and components.

| | LCS1 | LCS2 | LCS3 | CI1 | CI2 | CI3 |
|---|---|---|---|---|---|---|
| Perfect components of size 8 | 258 | 138 | 71 | 254 | 259 | 263 |
| Imperfect components of size 8 | 19 | 15 | 9 | 29 | 30 | 32 |
| Perfect components of size 7 | 209 | 155 | 100 | 218 | 227 | 244 |
| Imperfect components of size 7 | 34 | 13 | 9 | 37 | 45 | 86 |
| Perfect components of size 6 | 157 | 145 | 107 | 172 | 185 | 215 |
| Imperfect components of size 6 | 32 | 24 | 19 | 46 | 57 | 114 |
| Perfect components of size 5 | 206 | 269 | 199 | 222 | 251 | 299 |
| Imperfect components of size 5 | 32 | 39 | 24 | 44 | 76 | 135 |
| Perfect components of size 4 | 236 | 290 | 246 | 253 | 344 | 368 |
| Imperfect components of size 4 | 65 | 59 | 49 | 77 | 178 | 186 |
| Perfect components of size 3 | 465 | 726 | 815 | 509 | 551 | 573 |
| Imperfect components of size 3 | 70 | 90 | 80 | 87 | 129 | 134 |

**Table 2.** Distribution of perfect and imperfect components by size.

First, it is interesting to notice that the number of gene pairs decreases when one reduces the minimal length of matched LCS, which is expected, while it increases when one reduces the minimum side of discarded boxes when using common intervals. This different behavior is probably due to the fact that some boxes with short side can have a long side and then a bigger area, and would then be picked by our heuristic for the MIBC problem. We can also notice that the number of true positive gene pairs increases when discarding boxes of short minimum side, at the price of more false positives and unknown pairs. However, a preliminary study of these imperfect components showed that a significant number seems to be due to the fact that in some cases, when matching two occurrences of a common interval, one has to deal with multiple occurrences of a same family, a phenomenon that does not happen with LCS as the gene order is conserved. We discuss how to improve our method on this point in Section 4.

In terms of components, and especially of consistent components, that are, from our point of view, more important than single gene pairs, it seems that using common intervals lead to the discovery of more (both in terms of number and of ratio) perfect components, here again at the price of more imperfect components. However, it is interesting to notice an important difference between both methods, in terms of the size of the inferred perfect components. Discarding short LCS lead to the loss of large consistent components, in particular perfect components, which is probably due to the fact that LCS represent perfectly conserved colinear segments, and then long LCS that are present in several genomes are quite rare and significant. Hence discarding short LCS clearly improves the accuracy of the results but the price is that small components dominate in the combined matchings graph. On the other hand using common intervals of sequences allows to find more consistent perfect components of large size – at the price of more imperfect components – due to the less strict constraints imposed on matched segments.

From a more biological point of view, we can notice that there seems to be a set of approximatively 250 genes that form maximal perfect components in the combined matchings graph of these 8 genomes. It would be interesting to study

more precisely these genes and to compare them with the set of genes used in the phylogenomics analysis of $\gamma$-proteobacteria described in [20], or with the set of essential bacterial genes defined in [15].

### 3.2 Human and mouse genomes

In a second experiment, we considered the human and mouse genomes and we compared the results of using common intervals with the method based on LCS and with the recent MSOAR algorithm [14]. We downloaded the two genomes from the MSOAR website and we used the gene pairs of the MSOAR *hit graph* (see [14, Section 3.1]) and a single-linkage clustering to define gene families. In a subsequent step, we deleted from the two genomes all genes that belonged to a family present in only one genome.

Then, we computed 6 gene matchings using both the common intervals method and the LCS methods. These matchings are denoted CI1, CI2, CI3, LCS1, LCS2 and LCS3 as in the previous experiment. Then we classified pairs of positional homologs as true positives (TP) and false positives (FP)on the base of the gene names in Uniprot. The results are summarized in Table 3 below. The results for MSOAR were computed from the result file available on the MSOAR website.

| | MSOAR | LCS1 | LCS2 | LCS3 | CI1 | CI2 | CI3 |
|---|---|---|---|---|---|---|---|
| Matched pairs | 13218 | 13380 | 12386 | 11764 | 13301 | 12737 | 12394 |
| Number of TP | 9214 | 9227 | 8865 | 8491 | 9186 | 9008 | 8792 |
| Ratio of TP | 0.7 | 0.69 | 0.72 | 0.72 | 0.69 | 0.71 | 0.71 |
| Number of FP | 2240 | 2357 | 1921 | 1749 | 2327 | 2071 | 1996 |
| Ratio of FP | 0.17 | 0.18 | 0.16 | 0.15 | 0.17 | 0.16 | 0.16 |

**Table 3.** Classification of matched pairs in the gene matchings between human and mouse.

It appears that here again the LCS method performs better, but does lose information faster when short LCS are filtered in comparison to the common interval approach. Moreover, for the same reasons that we described in the previous section, we expect that improving the matchings of genes belonging to boxes with duplicated genes will improve the accuracy of the method based on common intervals. It would also be interesting to understand the reason why discarding boxes with a short minimum side does not increase the number of gene pairs, that could be due either to some properties of the method we used or to differences in the two considered datasets, both from the combinatorial point of view (8 genomes vs. 2 genomes) or the biological point of view.

## 4 Future work

In this section we present several ideas that would improve the quality of the matchings computed from common intervals of sequences. In particular, we be-

lieve that the general structure of our method allows very naturally to integrate the scores of the all-against-all sequence comparison used to define gene families.

For example, a preliminary analysis of the false positives in both experiments presented in Section 3 suggests that a significant number of them could be corrected by a less naive strategy for matching genes in trivial boxes in the fourth step of our method. A natural improvement could consist in considering the sequence comparison score between the genes of these trivial boxes, picking only pairs of genes that form a best bi-directional hit.

In the second step, we use a basic filtering criterion which rely on the size of the boxes. But it would be of interest to consider other criteria to select the best candidates. It could be useful, for example, to integrate here again the sequence comparison scores as a measure of quality of boxes. This would be a very natural way to balance the effects of the single-linkage strategy used generally to define gene families. One could also think to more sophisticated measures of "quality" of boxes, based on their combinatorial structure: a trivial box of size $3 \times 3$ is not necessarily a better candidate than a box of size $2 \times 2$ with an alphabet of size 2. Some examples of criteria, like "nestedness", defining what is a "good gene cluster" for genome comparison, can be found in [19].

In the third step, we considered a natural optimization problem, the MIBC problem, that involves only the geometry of boxes, both in its definition and in the heuristic we proposed. It would be interesting to integrate in this step the notion of quality of the boxes, used in the second step, which would lead to a weighted version of the MIBC problem.

From a conceptual point of view, the method we described can be seen as an extension of the LCS method used in [5], where the constraints on the notion of conserved structure used as anchors have been relaxed to accept rearrangements and paralogs. Previous works have already relaxed the notion of LCS, that correspond from a genomic point of view to perfectly colinear segments, allowing insertion or deletion of genes in colinear segments [17, 10]. It would be interesting to try to combine these two models and use as anchors common segments with rearrangements, paralogs, insertions and deletions. Several models exist like common intervals with errors [13] or gene teams, that can be computed in polynomial time in the case of two genomes [18]. However, with such relaxed models, the notion of quality of boxes would become more important in order to avoid to use boxes with a weak signal but good geometrical properties.

When considering a dataset of more than two genomes, it can happen that a phylogenetic tree is known for this dataset. In such a case, it would be interesting to perform our pairwise genomes comparisons according to this tree, as it is classical to do it in multiple sequences alignment. We think that such an approach could significantly reduce the number of imperfect components.

Finally in some cases, it would be interesting to consider that a gene in a genome can be matched with more than one other gene in the second genome. One can think for example to genomes that have undergone one or several whole genome duplication, like the yeasts genomes [9]. We think that common intervals of sequences are a good model to compute such generalized matchings.

## 5 Conclusion

We presented in this work a first study about using common intervals of sequences for computing positional homologs. The experimental results we obtained are very encouraging, despite using very simple approaches for each of the four steps of our method. In particular, the comparison of 8 bacterial genomes seems to indicate that this approach offers a good basis for the comparison of multiple genomes datasets.

Hence, based on this preliminary study, we believe that common intervals of sequences should be considered as a good model for the comparison of whole genomes, which differs from their initial application as a gene cluster model [22].

Moreover, as we described it in Section 4, the very pragmatic approach we proposed, based on four steps, allows to integrate easily more sophisticated technics, and we are currently developing some of these extensions, that will be available in the first release of CIGAL, our Common Intervals Global ALigner.

*Acknowledgments.* We thank Z. Fu for providing the hit graph of MSOAR [14].

## References

1. A. Bairoch *et al.*. The Universal Protein Resource (UniProt). *Nucleic Acids Res.* 33:D154–D159, 2005.
2. E. Belda, A. Moya, F. J. Silva. Genome rearrangement distances and gene order phylogeny in $\gamma$-proteobacteria. *Mol. Biol. Evol.*, 22(6):1456–1467, 2005.
3. G. Blin, C. Chauve, and G. Fertin. The breakpoints distance for signed sequences. In *CompBioNets 2004: Algorithms & computational methods for biochemical and evolutionary networks*, vol. 3 of *Texts in Algorithmics*, p. 3–16. King's Coll. Pub., London, 2004.
4. G. Blin, R. Rizzi. Conserved interval distance computation between non-trivial genomes. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*, *LNCS* 3595:22–31. Springer, Berlin, 2005.
5. G. Blin, C. Chauve, G. Fertin. Gene order and phylogenetic reconstruction: application to $\gamma$-proteobacteria. In *Comparative Genomics, RECOMB 2005 International Workshop, RCG 2005, Dublin, Ireland, September 18-20, 2005, Proceedings*, *LNCS/LNBI*, 3678:11–20. Springer, Berlin, 2005.
6. G. Bourque, Y. Yacef, N. El-Mabrouk. Maximizing synteny blocks to identify ancestral homologs. In *Comparative Genomics, RECOMB 2005 International Workshop, RCG 2005, Dublin, Ireland, September 18-20, 2005, Proceedings*, *LNCS/LNBI*, 3678:21–34. Springer, Berlin, 2005.
7. D. Bryant. The complexity of calculating exemplar distances. In *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, p. 207–212. Kluwer Acad. Press., Dordrecht, 2000.
8. I. J. Burgetz, S. Shariff, A. Pang, E. R. M. Tillier. Positional homology in bacterial genomes. *Evolutionary Bioinformatics Online*, 2:42–55, 2006.
9. K. P. Byrne, K, H, Wolfe. The Yeast Gene Order Browser: combining curated homology and syntenic context reveals gene fate in polyploid species. *Genome Res.*, 15(10):1456–1461, 2005.

10. S. B. Cannon, A. Kozik, B. Chan, R. Michelmore, N. D. Young. DiagHunter and GenoPix2D: programs for genomic comparisons, large-scale homology discovery and visualization. *Genome Biology* 4:R68, 2003.
11. C. Chauve, G. Fertin, R. Rizzi, S. Vialette. Genomes containing duplicates are hard to compare. In *Computational Science - ICCS 2006, 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part II.*, *LNCS*, 3992:783–790. Springer, Berlin, 2006.
12. X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4):302–315, 2005.
13. C. Chauve, Y. Diekmann, S. Hebber, J. Mixtacki, S. Rahmann, J. Stoye. On Common Intervals with Errors Report 2006-02, Technische Fakultät der Universität Bielefeld, Abteilung Informationstechnik. 2006.
14. Z. Fu, X. Chen, V. Vacic, P. Nan, Y. Zhong, J. Tang. A parsimony approach to genome-wide ortholog assignment. In *Research in Computational Molecular Biology, 10th Annual International Conference, RECOMB 2006, Venice, Italy, April 2-5, 2006, Proceedings*, *LNCS/LNBI*, 3909:578–594. Springer, Berlin, 2006.
15. J. I. Glass *et al.*. Essential genes of a minimal bacterium. *Proc. Natl. Acad. Sci. USA*, 103(2):425–430, 2006.
16. A. Goldstein, P. Kolman, J. Zheng. Minimum common string partition problem: hardness and approximations. In *Algorithms and Computation, 15th International Symposium, ISAAC 2004, HongKong, China, December 20-22, 2004, Proceedings*, *LNCS*, 3341:473–484. Springer, Berlin, 2004.
17. B. J. Haas, A. L. Dechler, J. R. Wortman, S. L. Salzberg. DAGchainer: a tool for mining segmental genome duplication and synteny. *Bioinformatics*, 20(18):3643–3646, 2004.
18. X. He, M.H. Goldwasser. Identifying conserved gene clusters in the presence of homology families. *J. Comput. Biol.*, 12(6):638–656, 2005.
19. R. Hoberman, D. Durand. The incompatible desiderata of gene cluster properties. In *Comparative Genomics, RECOMB 2005 International Workshop, RCG 2005, Dublin, Ireland, September 18-20, 2005, Proceedings*, *LNCS/LNBI*, 3678:73–87. Springer, Berlin, 2005.
20. E. Lerat, V. Daubin, N. A. Moran. From gene trees to organismal phylogeny in prokaryotes: the case of the $\gamma$-Proteobacteria. *PLoS Biol.*, 1(1):E19, 2003.
21. D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.
22. T. Schmidt, J. Stoye. Quadratic time algorithms for finding common intervals in two and more sequences. In *Combinatorial Pattern Matching, 15th Annual Symposium, CPM 2004, Istanbul,Turkey, July 5-7, 2004, Proceedings*, *LNCS*, 3109:97–108. Springer, Berlin, 2004.
23. K. M. Swenson, M. Marron, J. V Earnest-DeYoung, B. M. E. Moret. Approximating the true evolutionary distance between two genomes. In *Proceedings of the Seventh Workshop on Algorithm Engineering and Experiments and the Second Workshop on Analytic Algorithmics and Combinatorics (ALENEX/ANALCO)*, p. 121–129. SIAM Press, New York, 2005.
24. K. M. Swenson, N. D. Pattengale, B. M. E. Moret. A framework for orthology assignment from gene rearrangement data. In *Comparative Genomics, RECOMB 2005 International Workshop, RCG 2005, Dublin, Ireland, September 18-20, 2005, Proceedings*, *LNCS/LNBI*, 3678:11–20. Springer, Berlin, 2005.

## A Proof of NP-completeness of MIBC

We consider here the decision versions of the problems MIBC and MCSP. We first state formally the MCSP problem. In the following, given a string $S$, let $S[i]$ and $S[i..j]$ denote respectively the $i^{th}$ character of $S$ and the substring starting at position $i$ and ending at position $j$ of $S$. Given two strings $S$ and $T$, a *common substring* of $S$ and $T$ is defined by a quadruplet $(i, j, k, l)$ such that $S[i, j] = T[k, l]$. Two strings $S$ and $T$ are *balanced* if every letter appears the same number of times in $S$ and $T$. According to [16], a *partition* of a string $S$ is a sequence $P = (P_1, P_2, \ldots, P_m)$ of strings whose concatenation is equal to $S$. Let $P$ (resp. $Q$) be a partition of a string $S$ (resp. $T$). The pair $(P, Q)$ is a *common partition* of $S$ and $T$ if $Q$ is a permutation of $P$. Given a partition $P = (P_1, P_2, \ldots, P_m)$ of a string, each string $P_i$ is called a *block* of $P$. We say that a block $P_i = S[k..l]$ *covers* any substring of $S[k..l]$. Two blocks $P_i$ and $P_j$ are *disjoint* if they do not both cover the same character. A common partition $(P, Q)$ of $S$ and $T$ can be naturally interpreted as a bijective mapping from $P = (P_1, P_2, \ldots, P_m)$ to $Q = (Q_1, Q_2, \ldots, Q_m)$. Given a common partition $(P, Q)$, we note $\pi(i) = j$ if $P_i$ is mapped to $Q_j$. The NP-hard problem [16] MCSP is the following: given two balanced strings $S$ and $T$ and a positive integer $s$, find a common partition $(P, Q)$ of $S$ and $T$ with at most $s$ blocks.

For the sake of clarity, we recall here the version of the MIBC problem that we consider below: Given a set of $n$ boxes $B = \{B_1, B_2, \ldots, B_n\}$ and a positive integer $s'$, the problem asks to find a subset $B' \subseteq B$ of cardinality lower than or equal to $s'$, such that (1) given any pair $(B_i, B_j)$ of boxes of $B'$, $B_i$ and $B_j$ are compatible and (2) given any box $B_m \in B$ such that $B_m \notin B'$, $\exists B_i \in B'$ such that $B_i$ and $B_m$ are not compatible ($B'$ is said to be *maximal*).

Clearly, MIBC problem is in NP since given a set $B'$ of boxes, one can check in polynomial-time if (1) any pair of boxes of $B'$ is compatible, (2) $B'$ is maximal and (3) $|B'| \leq s'$. We show that given any instance $(S, T, s)$ of MCSP problem, we can construct in polynomial-time an instance $(B, s')$ of MIBC problem such that there exists a common partition $(P, Q)$ of $S$ and $T$ with at most $s$ blocks iff there exists a maximal subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to $s'$.

We detail this construction hereafter. Let $(S, T, s)$ be any instance of MCSP problem. For each common substring $(i, j, k, l)$ of $S$ and $T$, we create a box $(i, j, k, l)$ in $B$. This can be done in polynomial-time by the use of a generalized suffix-tree for instance. In order to complete the definition of the MIBC problem instance, we define $s'$: $s' = s$. We denote by *box-construction* any construction of this type. An illustration of a box-construction is given in Figure 1.

We now turn to proving that our construction is a polynomial-time reduction from MCSP problem to MIBC problem.

**Lemma 1.** *Let $(S, T, s)$ be an instance of* MCSP *problem, and $(B, s')$ an instance of* MIBC *problem obtained by a box-construction from $(S, T, s)$. There exists a common partition $(P, Q)$ of $S$ and $T$ with at most $s$ blocks iff there ex-*

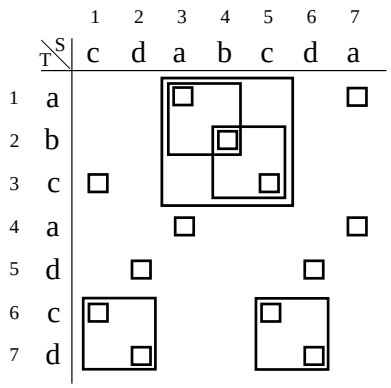


**Fig. 1.** A schematic view of the set of boxes $B$ obtained by a box-construction of $S = cdabcda$ and $T = abcadcd$. For instance, the largest box is defined by the quadruplet $(3, 5, 1, 3)$.

*ists a maximal subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to $s'$.*

*Proof.* ($\Rightarrow$) Suppose we have a common partition $(P, Q)$ of $S$ and $T$ with $m$ blocks ($m \leq s$). We look for a subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to $s$. We define this set of boxes as follows. For each pair $(P_i, Q_j)$ such that $P_i = S[s_i..e_i]$, $Q_j = T[s_j..e_j]$, $\pi(i) = j$, $1 \leq i, j \leq m$, $1 \leq s_i, s_j, e_i, e_j \leq |S|$, add the box $(s_i, e_i, s_j, e_j)$ to $B'$.

By construction, notice that the set $B'$ is of cardinality equal to $m$, which is by definition lower than $s$. Remains to us to prove that (1) $B'$ is a set of compatible boxes and (2) $B'$ is maximal. By definition, since $(P, Q)$ is a bijective mapping from $P$ to $Q$, for any $1 \leq i \leq m$ there exists only one $1 \leq j \leq m$ such that $\pi(i) = j$. Moreover, since $P$ (resp. $Q$) is a partition, the blocks composing $P$ (resp. $Q$) are disjoint. Therefore, given any box $(s_i, e_i, s_j, e_j)$ of $B'$, there is no box $(s_k, e_k, s_l, e_l)$ in $B'$ such that $[s_i, e_i] \bigcap [s_k, e_k] \neq \emptyset$ or $[s_j, e_j] \bigcap [s_l, e_l] \neq \emptyset$. Thus, $B'$ is a set of compatible boxes.

Let us now prove that $B'$ is maximal. Suppose it is not the case. Thus, there exists a box $(s_i, e_i, s_j, e_j)$ of $B$ that can be added to $B'$ such that $B'$ is still a set of compatible boxes. Then, by construction, the corresponding substrings $S[s_i, e_i]$ and $T[s_j, e_j]$ are not covered by any block of $P \bigcup Q$. Therefore, $P$ and $Q$ are not partitions of $S$ and $T$; a contradiction. Therefore, if there exists a common partition $(P, Q)$ of $S$ and $T$ with at most $s$ blocks then there exists a maximal subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to $s'$.

($\Leftarrow$) Suppose we have a maximal subset of compatible boxes $B' \subseteq B$ of cardinality equal to $m$ ($m \leq s$). We look for two partitions $P$ and $Q$ of $S$ and $T$ each with at most $s$ blocks. We define $P$ and $Q$ as follows. For each box $(s_i, e_i, s_j, e_j)$ of $B'$, we define a block $P_i = S[s_i..e_i]$ and a block $Q_j = T[s_j..e_j]$. Let $P = (P_1, P_2, \ldots, P_m)$ and $Q = (Q_1, Q_2, \ldots, Q_m)$.

By construction, for each box $(s_i, e_i, s_j, e_j)$ of $B'$, there exist a block $P_i = S[s_i..e_i]$ and a block $Q_j = T[s_j..e_j]$. By definition of a box-construction, $S[s_i..e_i]$ and $T[s_j..e_j]$ are common substrings of $S$ and $T$ (*i.e.* $S[s_i..e_i] = T[s_j..e_j]$). Therefore one can built a bijective mapping of $P$ to $Q$ where for each box $(s_i, e_i, s_j, e_j)$ of $B'$, the corresponding blocks $P_i$ and $Q_j$ are mapped (*i.e.* $\pi(i) = j$). Clearly, $Q$ is thus a permutation of $P$. Moreover, the partitions $P$ and $Q$ are composed of exactly $m$ blocks which is by definition lower than or equal to $s$. In order to prove that $(P, Q)$ is a common partition of $S$ and $T$, it remains to us to prove that $P$ (resp. $Q$) is a partition of $S$ (resp. $T$).

First, notice that since $B'$ is a set of compatible boxes, in $P$ (resp. $Q$) blocks are disjoint two by two. Let us prove that the blocks of $P$ (resp. $Q$) cover the string $S$ (resp. $T$). Suppose it is not the case. Since $S$ and $T$ are balanced strings, there exist a position $i$ (resp. $j$) in $S$ (resp. $T$) such that $S[i]$ (resp. $T[j]$) is not covered by any block of $P$ (resp. $Q$) and $S[i] = T[j]$. By definition, since $S[i] = T[j]$, no box in $B'$ covers $S[i]$ and $T[j]$. Moreover, by construction, there exists a box $(i, i, j, j)$ in $B$ which could be added to $B'$ such that $B'$ will still be a set of compatible boxes. Therefore, $B'$ is not maximal; a contradiction. We just prove that if there exists a maximal subset of compatible boxes $B' \subseteq B$ of cardinality lower than or equal to $s'$ then there exists a common partition $(P, Q)$ of $S$ and $T$ with at most $s$ blocks. □