Inferring gene orders from gene maps using the breakpoint distance

Guillaume Blin[1], Eric Blais[2], Pierre Guillon[1], Mathieu Blanchette[2], and Nadia El-Mabrouk[3]

[1] IGM-LabInfo - UMR CNRS 8049 - Université de Marne-la-Vallée - France
{gblin,pguillon}@univ-mlv.fr
[2] McGill Centre for Bioinformatics - McGill University - H3A 2B4 -Canada
{eblais,blanchem}@mcb.mcgill.ca
[3] DIRO - Université de Montréal - H3C 3J7 - Canada
mabrouk@iro.umontreal.ca

Abstract. Preliminary to most comparative genomics studies is the annotation of chromosomes as ordered sequences of genes. Unfortunately, different genetic mapping techniques usually give rise to different maps with unequal gene content, and often containing sets of unordered neighboring genes. Only partial orders can thus be obtained from combining such maps. However, once a total order O is known for a given genome, it can be used as a reference to order genes of a closely related species characterized by a partial order P. In this paper, the problem is to find a linearization of P that is as close as possible to O in term of the breakpoint distance. We first prove an NP-complete complexity result for this problem. We then give a dynamic programming algorithm whose running time is exponential for general partial orders, but polynomial when the partial order is derived from a bounded number of genetic maps. A time-efficient greedy heuristic is then given for the general case, with a performance higher than 90% on simulated data. Applications to the analysis of grass genomes are presented.

1 Introduction

Despite the increase in the number of sequencing projects, the choice of candidates for complete genome sequencing is usually limited to a few model organisms and species with major economical impact. For example, the rice genome is the only crop genome that has been completely sequenced. Other grasses of major agricultural importance such as maize and wheat are unlikely to be sequenced in the short term, due to their large size and highly repetitive composition. In this case, all we have are partial maps produced by recombination analysis, physical imaging and other mapping techniques that are inevitably missing some genes (or other markers) and fail to resolve the ordering of some sets of neighboring genes. Only partial orders can thus be obtained from combining such maps. The question is then to find an appropriate order for the unresolved sets of genes. This is important not only for genome annotation, but also for the study of evolutionary relationships between species. Once total orders have been identified,

the classical genome rearrangement approaches can be used to infer divergence histories in terms of global mutations such as reversals [2, 9, 14].

In a recent study [16, 18], Sankoff *et. al.* generalized the rearrangement by reversal problem to handle two partial orders. The idea was to resolve the partial orders into two total orders having the minimal reversal distance with respect to each other. The problem has been conjectured NP-hard, and a branch-and-bound algorithm has been developed for this purpose. The difficulty of this approach is partly due to the fact that both compared genomes have partially resolved gene orders. However, once a total order is known for a given genome, it can be used as a reference to order markers of closely related species. For example, as the grass genomes maintain a high level of conserved synteny [11, 7], maps of the completely sequenced rice genome can be used to deduce an order of markers in other grass genomes such as maize.

In this paper, given a reference genome characterized by a total order O and a related genome characterized by a partial order P, the problem is to find a total order coherent with P minimizing the breakpoint distance with respect to O. The underlying criterion is a parsimony one assuming a minimum number of genomic rearrangements. After introducing the basic concepts in Section 2, we show in Section 3 that the considered problem is NP-complete. We then give, in Section 4, two dynamic programming algorithms. First is an algorithm that solves exactly the problem on arbitrary partial orders, and whose worst-case running time is exponential in the number of genes. However, when the partial order considered is the intersection of a bounded number of genetic maps of bounded width, the algorithm runs in polynomial time. We then present a fast and accurate heuristic for the general problem in Section 5. We finally report results on simulated data, and applications to grass genetic maps in Section 6.

2 A graph representation of gene maps

Hereafter, we refer to elementary units of a map as genes, although they could in reality be any kind of markers. Moreover, as the transcriptional orientation of genes is usually missing from genetic maps, we consider unsigned genes.

A *genetic map* is represented as an ordered sequence of gene subsets or *blocks* $B_1, B_2, \ldots, B_q$, where for each $1 \leq i \leq q$, genes belonging to block B_i are incomparable among themselves, but precede those in blocks $B_{i+1}, \ldots, B_q$ and succeed those in blocks $B_1, \ldots, B_{i-1}$. For example, in Figure 1.a, $\{4, 5\}$ is a block, meaning that genes 4 and 5 are assigned to the same position, possibly due to lack of recombination between them. A genetic map is thus a partial order of genes.

Maps $M_1, \ldots, M_m$ obtained from various protocols can be combined to form a more complex partial order P on the union set of genes of all maps as follows: a gene a precedes a gene b in P if there exists a map M_i where a precedes b. However, combining maps can be a problem in itself, due to possible inconsistencies, which would create precedence cycles (e.g. a precedes b in M_1 but b precedes a in M_2). Breaking cycles can be done in different ways, the parsimo-

nious method consisting in eliminating a minimum number of precedence rules. Another potential problem is the presence of multiple loci (markers that are assigned to different positions in the same map). These issues have been considered in previous studies [17, 18, 16], and a software is available for combining genetic maps [10]. In this paper, we assume that the partial order P is already known.

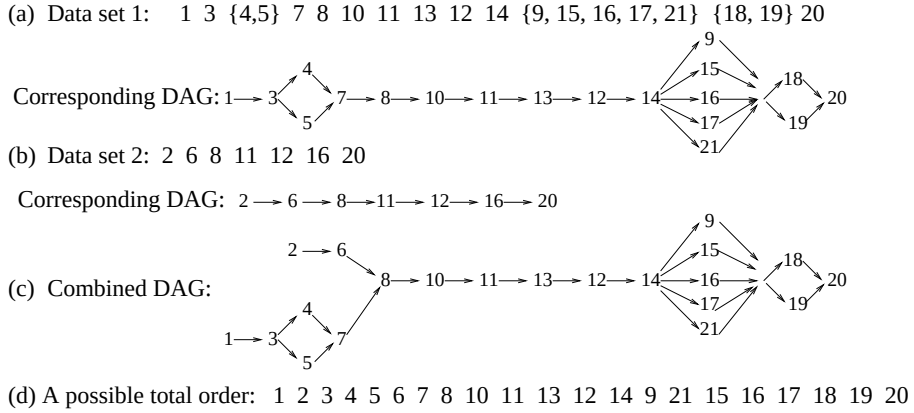


Fig. 1. Data extracted from the comparison of maize and sorghum in the Gramene database. The identity permutation $(1\ 2\ 3\ 4\ \cdots)$ represents the order of markers in the "IBM2 neighbors 2004"[15] map for maize chromosome 5. The corresponding marker's partial orders in sorghum are deduced from (a) "Paterson 2003" [4] map of the chromosome labelled C and (b) "Klein 2004" [13] map of the chromosome labelled LG-01; (c) is the partial order obtained by combining (a) and (b); (d) is a linearization of (c) minimizing the number of breakpoints.

As proposed in previous studies [17, 18], we represent such a partial order P as a directed acyclic graph (or DAG) (V_P, E_P), where the vertex set V_P represents the set of genes along the chromosome and the edge set E_P represents the available order information (Figure 1). We consider a minimum set of edges, in the sense that any edge of E_P can not be deduced by transitivity from other edges. In particular, a total order of genes is represented by a DAG such that each edge connects a pair of consecutive genes.

Let P be a partial order represented by a DAG (V_P, E_P). We say that a vertex a is *P-adjacent* to a vertex b and write $a <_P b$ iff there is an edge in E_P from a to b. We say that a *precedes* b in P and write $a \ll_P b$ iff there is a path from a to b. We say that vertices a and b are *incomparable* if neither $a \ll_P b$ nor $b \ll_P a$, and denote this by $a \sim_P b$. A *linearization* of P is a total order O' on the same set of genes, such that $a \ll_P b \Rightarrow a \ll_{O'} b$.

Given a partial order P and a total order O, our goal is to find a linearization O' of P, in such a way that O' is as "similar" as possible to O. The distance measure used here is the number $bkpts(O, O')$ of breakpoints between O and O', where a *breakpoint* is a pair of consecutive vertices (a, b) of O' that are not

consecutive in O ($a <_{O'} b$ but $a \not<_O b$). Equivalently, we may try to maximize the number of O-adjacencies of O', which is defined as the number of pairs of consecutive genes in O' that are also consecutive in O (Figure 1.d). Formally:

Minimum-Breakpoint Linearization (MBL) problem
Given: A partial order P and a total order O on the set of genes $\{1, 2, \ldots, n\}$,
Find: A linearization of P into a total order O' so that $bkpts(O, O')$ is minimized.

Without loss of generality, we assume from now on that O is the identity permutation $(1, 2, \ldots, n)$, that is $i <_O j \Leftrightarrow j = i + 1$.

3 Hardness results

In this section, we prove that the decision version of the MBL problem is **NP**-complete: given a complete order O and a partial order P defined on the same set of genes and an integer k', can one find a linearization O' of P such that $bkpts(O, O') \leq k'$?

We propose a reduction from the **NP**-complete problem Maximum Independent Set [8]: given a graph $G = (V, E)$ and an integer k, can one find an independent set of vertices of G – *i.e.* a set $V' \subseteq V$ such that no two vertices of V' are connected by an edge in E – of cardinality greater than or equal to k ?

We initially note that the MBL problem is in NP since given a complete order O and a linearization O' of P, one can compute the number of breakpoints in linear time. In order to prove that the MBL problem is **NP**-complete, we show that from any instance of Maximum Independent Set with a parameter k, we are able to construct – in polynomial time – an instance of the MBL problem such that k' depends on k. We detail this construction hereafter.

For convenience, we define a reduction from a slightly different set of instances for the Maximum Independent Set problem: connected graphs. This can be done w.l.o.g. since the problem is still **NP**-complete in that case. Let $G = (V, E)$ be a connected graph of n vertices. We define the complete and the partial orders O and P of the MBL problem as follows. The complete order O is defined as a string $O = \delta\ \alpha_1\ \beta_1\ \gamma_1\ \alpha_2\ \beta_2\ \gamma_2\ \ldots\ \alpha_n\ \beta_n\ \gamma_n\ \epsilon$, and the partial order P as a DAG $P = (V_P, E_P)$ with $V_P = \{\delta, \alpha_1, \alpha_2 \ldots \alpha_n, \beta_1, \beta_2 \ldots\ \beta_n, \gamma_1, \gamma_2, \ldots \gamma_n, \epsilon\}$ and $E_P = \{(\delta, \gamma_1)\} \cup \{(\gamma_i, \gamma_{i+1}) | 1 \leq i < n\} \cup \{(\gamma_n, \alpha_i), (\gamma_n, \beta_i) | 1 \leq i \leq n\} \cup \{(\beta_i, \alpha_j), (\beta_j, \alpha_i) | \forall (v_i, v_j) \in E\} \cup \{(\alpha_i, \epsilon), (\beta_i, \epsilon) | 1 \leq i \leq n\}$.

In order to complete the instance of the MBL problem, we define $k' = (3n + 1) - k$. In the following, we will refer to any such construction as a *MBL-construction*.

Theorem 1. *A connected graph $G = (V, E)$ admits an independent set of vertices $V' \subseteq V$ of cardinality greater than or equal to k if and only if there exists a linearization O' of P such that $bkpts(O, O') \leq (3n + 1) - k$, where O and P result from a MBL-construction of G.*

Proof. See Appendix.

Corollary 1. *The MBL problem is NP-complete.*

4 Exact dynamic programming algorithms

Hereafter, we describe two exact dynamic programming algorithms for solving the MBL problem. The first algorithm works on an arbitrary partial order P, but has a running time that can be exponential in $|V_P|$. However, we show that the algorithm's running time is polynomial in the more realistic case where P is built from a sbounded et of genetic maps. The second algorithm applies to the case where P is built from a single genetic map, and runs in linear time.

We begin with some preliminary definitions. Let A be a subset of vertices of V_P. A is a *border* of P iff any pair of vertices of A are incomparable, and a *maximum border* iff any other vertex of V_P is comparable to at least one vertex of A. We also define, for any subset $B \subseteq V_P$, $front(B) = \{x \in B : x \text{ has no successor in } B\}$. Note that the front of any set B is a border. Finally, we denote $pred(A) = A \cup \{x \in V_P : \exists y \in A \text{ s.t. } x \ll_P y\}$.

4.1 A dynamic algorithm for arbitrary partial orders

Let A be a maximum border. We denote by $X_{A,i}$ the maximum number of adjacencies that can be obtained from a linearization of $pred(A)$ that is consistent with the partial order P, and that ends with vertex i (i.e., i is the rightmost vertex in the total order of $pred(A)$). It is easy to see that the number of adjacencies in the global optimal solution is $\max_{i \in F} X_{F,i}$ adjacencies, where $F = front(V_P)$. The following theorem provides a recursive formula for the computation of $X_{A,i}$.

Theorem 2. *For any border A and any vertex $i \in A$,*

$$X_{A,i} = \max_{j \in A'} X_{A',j} + \begin{cases} 1 & if\, |j - i| = 1 \\ 0 & otherwise \end{cases}$$

where

$$A' = front(pred(A) \setminus \{i\})) = (A \setminus \{i\}) \cup \{k \,|\, (k,i) \in E_P \text{ and } k \notin pred(A \setminus \{i\})\}$$

.

begins with $A = F = border(V_P)$ and stops as soon as A is the empty set.

Computing all the entries of the dynamic programming table only requires operations which can be done in linear time. If the partial order P admits $b(P)$ possible borders, the running time is $O(b(P) \cdot |V_P|^2)$.

In the general case, the number of borders of P can be as much as $2^{|V_P|}$, if P consists of a single block of incomparable vertices. However, we are more interested in the case where P is obtained by combining m genetic maps, where each map contains a maximum of q blocks and the size of each block is at most k. In this case, there are at most q^m maximal borders in P. Furthermore, two elements that are in the same border cannot be in different blocks on a

genetic map, so each maximal border is of size at most km, which allows 2^{km} possible subsets. Therefore, the total number of borders of P is bounded above by $b(P) \in O(q^m 2^{km})$. Since, in practice, only two or three different genetic maps are combined to form a partial order, the dynamic algorithm yields a practical and exact solution to the MBL problem.

4.2 A linear-time algorithm for single genetic map

When P is a genetic map consisting of a list of blocks $B_1, B_2, \ldots, B_q$, a much faster linearization algorithm exists. Let X_i be the maximum linearization score obtained in the partial subset $B_1 \cup \cdots \cup B_i \subseteq V_P$. The maximum linearization score of P is thus equal to X_q. Let L_i represent the set of elements in B_i that can be placed at the last position in a total ordering of $B_1 \cup \cdots \cup B_i$ that achieves the score X_i. Define the functions $g_1(X, Y) = \{x \,|\, x \in X \text{ and } x + 1 \in Y\}$ and $g_2(X, Y) = \{y \,|\, y \in Y \text{ and } y - 1 \in X\}$. Then, the values X_i and L_i can be determined recursively as follows.

Theorem 3. *Define* $X_0 = 0$ *and* $L_0 = \{\}$. *Then, for any* $1 \leq i \leq q$,

$$X_i = X_{i-1} + |g_1(B_i, B_i)| + \begin{cases} 1 \text{ , } \textit{if } |g_2(L_{i-1}, B_i)| \geq 1 \\ 0 \text{ , } \textit{otherwise} \end{cases}$$

and

$$L_i = \begin{cases} B_i \setminus g_1(B_i, B_i) & \text{, } \textit{if } |g_2(L_{i-1}, B_i)| \neq 1 \textit{ or } |B_i| = 1 \\ B_i \setminus (g_1(B_i, B_i) \cup g_2(L_{i-1}, B_i)) & \text{, } \textit{otherwise} \end{cases}$$

The intuition behind the recursive definition of X_i is as follows: to get the maximum linearization score, we always want to join as many elements $x, x+1$ within a same block. Furthermore, as much as possible, we want to join consecutive elements in neighboring blocks as well. The set L_i is used to keep track of which elements can be put last in the ordering of B_i and therefore possibly be matched with an element in the block B_{i+1}. If the elements of B_i are stored in an ordered list, then the recursive definition of Theorem 3 can be implemented in a recursive algorithm for which each iteration requires $O(|B_i| + |L_{i-1}|)$ time to run, for a total time complexity of $O(n)$ in the case of a genetic map of n genes.

5 An efficient heuristic

Since our exact dynamic programming has a worst-case running time that is exponential in the number of genes, a faster heuristic is required to solve large problem instances. In this section, a greedy heuristic is developed for general partial orders obtained from the concatenation of an arbitrary number of maps. It aims to find a maximum number of O-adjacencies coherent with a partial order P. At each step, the partial order is updated by incorporating adjacencies

of the longest O-adjacency path that can be part of a linearization of P. The algorithm does not necessarily end up with a total order. Rather, it stops as soon as no more adjacencies can be found. All linearizations of the obtained partial order are then equivalent in the sense that they all give rise to the same number of adjacencies.

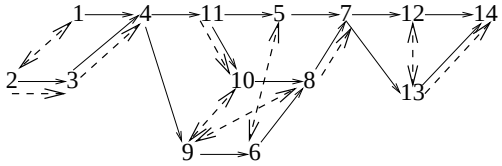


Fig. 2. Dotted edges are all O-adjacencies that can, individually, be part of a linearization of P. A bi-directional edge represents the concatenation of two edges, one in each direction. An adjacency path of P is a directed sequence of consecutive dotted edges.

A *direct* (resp. *indirect*) *adjacency path* of P is a sequence of vertices of form $(i, i+1, i+2, \cdots i+k)$ (resp. $(i+k, \cdots i+2, i+1, i)$) such that for any $0 \leq j < k$, either $i + j <_P i + j + 1$ (resp. $i + j + 1 <_P i + j$), or $i + j$ and $i + j + 1$ are incomparable. For example, in Figure 2, $(1, 2, 3, 4)$ (resp. $(11, 10, 9, 8, 7)$) is a direct (resp. indirect) adjacency path. Notice that adjacencies of this indirect path can not belong to any linearization of P, as gene 5 should be located after 11 but before 7.

We say that an adjacency path p of P is *valid* iff there is a linearization O' of P such that p is a subsequence of O'. Lemma 1 gives the conditions for an adjacency path to be valid. We need a preliminary definition.

Definition 1. *Given two vertices i and j, we say that i is* compatible *with j iff the two following conditions hold:*

1. *i and j are either incomparable or $i \ll_P j$;*
2. *Any vertex v verifying $i \ll_P v \ll_P j$ belongs to the interval $[i, j]$ (or $[j, i]$ if $j < i$).*

Lemma 1. *A direct (resp. indirect) adjacency path of P from i to $i + k$ (resp. from $i + k$ to i) is valid if and only if, for any j_1, j_2 such that $0 \leq j_1 < j_2 \leq k$, $i + j_1$ is compatible with $i + j_2$. (resp. $i + j_2$ is compatible with $i + j_1$).*

A preliminary preprocessing of $P = (V_P, E_P)$ is required to efficiently compute successive adjacency paths.

1. Create the matrix M of size $|V_P| \times |V_P|$ verifying, for any $i, j \in V_P$, $M(i, j) = 1$ iff $i <_P j$ and $M(i, j) = 0$ otherwise.
2. Compute the transitive closure of M, that is the matrice M^T of size $|V_P| \times |V_P|$ verifying, for any $i, j \in V_P$,

$$M^T(i,j) = \begin{cases} 1 & \text{iff } i <_P j \\ 2 & \text{iff } i \ll_P j \text{ but } i \not<_P j \\ 0 & otherwise \end{cases}$$

After the preprocessing step, the following Steps 1 and 2 are iterated as long as P contains an adjacency path.

- **Step 1:** Find a longest valid direct or indirect adjacency path. An algorithm for this step is described in Figure 3.
- **Step 2:** Incorporate the new adjacencies in M^T, and compute the transitive closure of M^T.

Algorithm Find-Valid-Path (Figure 3) only considers the case of direct paths, though generalization to indirect paths is straightforward. Valid paths are computed beginning with paths of size 2. For a fixed k, any path $p = (i, i+1, \cdots i+k)$ of size k is obtained from a concatenation of two valid consecutive paths $p_1 = (i, i+1, \cdots i+k-1)$ and $p_2 = (i+1, i+2, \cdots i+k)$ of size $k-1$. As p_1 and p_2 are valid paths, the path p is valid iff i is compatible with $i+k$.

Algorithm Find-Valid-Path (P)
{*Compute the list L of all adjacency paths of size 2*}
For $1 \leq i \leq |V|$ *do*
  If ($i <_P i+1$) or (i and $i+1$ are incomparable) *then*
    Add $(i, i+1)$ to L;
End For
$k = 2$;

{*As long as L contains at least two elements, concatenate paths of size k to paths of size* $k+1$}
While $|L| \geq 2$ *do*
  For $j = 1$ to $|L|$ *do*
    If L_{j+1} and L_j are consecutive paths *then*
      If $L_j[1]$ is compatible with $L_{j+1}[k]$ *then*
        L' = Concatenate(L_j, L_{j+1});
        Add L' to LNew;
  End For
  If $|LNew| > 0$ *then* $L = LNew$; Clear(LNew);
  $k = k+1$;
End While
Return (L_1);

Fig. 3. Finding a longest valid adjacency path of P. L is the list of adjacency paths of size k, L_j denotes the j^{th} path of L, and $L_j[i]$ the i^{th} vertex of L_j.

Complexity: Computing the transitive closure of the adjacency matrix in the preprocessing phase, as well as in Step 2, is done using the Floyd-Warshall algorithm [6] in time complexity $O(n^3)$ where n is the number of vertices of the corresponding graph. As each condition of *Algorithm Find-Valid-Path* can be checked in constant time and L contains at most $|V| - 1$ elements, the time complexity of Step 1 is in $O(n)$. Moreover, Steps 1 and 2 are iterated at most

$|V|$ times. Therefore, the worst time complexity of the greedy algorithm is in $O(n^4)$.

6 Experimental results

We first test the efficiency of the heuristic compared to the dynamic programming algorithm for general partial orders on simulated data, and then illustrate the method on grass maps obtained from Gramene (http://www.gramene.org/).

SIMULATED DATA: We simulate DAGs of fixed size n that can be represented as a linear expression involving the operators '$\rightarrow$' and ',' where P-adjacent genes are separated by a '$\rightarrow$' and incomparable genes by a ','. Such a representation is similar to the one used in [12, 17]. For example, the DAG in Figure 1.c has the following string representation:

$$\{2 \rightarrow 6, 1 \rightarrow 3 \rightarrow \{4, 5\} \rightarrow 7\} \rightarrow 8 \cdots 14 \rightarrow \{9, 15, 16, 17, 21\} \rightarrow \{18, 19\} \rightarrow 20$$

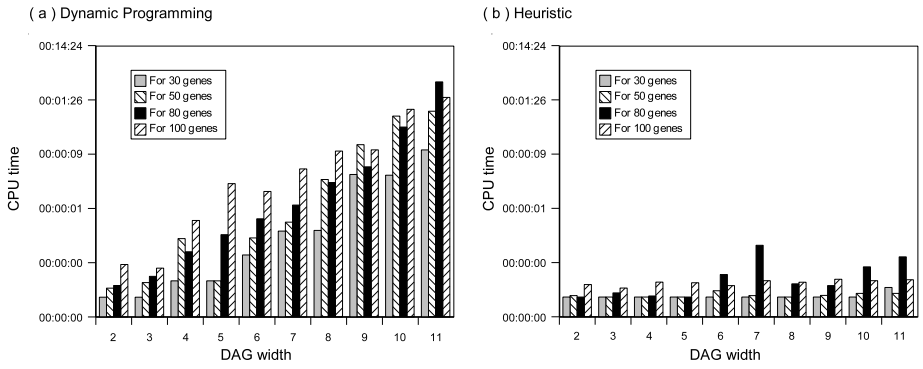


Fig. 4. CPU time expended by (a) the dynamic programming algorithm and (b) the heuristic, for DAGs of a given size and width. Each result is obtained from 10 runs (10 different simulated DAGs). The Y axis is logarithmic.

DAGs are generated according to two parameters: the *order rate* p that determines the number of ',' in the expression, and the *gene distribution rule* q corresponding to the probability of possible O-adjacencies. We simulated twenty different instances for each triplet of parameters (n, p, q) with $k \in \{30, 50, 80, 100\}$, $p \in \{0.7, 0.9\}$ and $q \in \{0.4, 0.6, 0.8\}$. We did not consider p values lower than 0.7, as the dynamic programming algorithm exponential-time prevented us from testing such instances.

Figure 4 shows that the running time of the dynamic programming algorithm grows exponentially with the DAG's largest border size, while the heuristic is

not affected by it (this was expected, as the time-complexity only depends on the DAG's size). We note that the greedy heuristic can easily handle partial orders consisting of thousands of genes.

We now evaluate the heuristic's optimality, namely the number of O-adjacencies resulting from the obtained linearization compared to the optimal solution (obtained with the dynamic programming algorithm). As illustrated by Table 1, the performance of the greedy algorithm is almost always higher than 90%, and usually close to 100%.

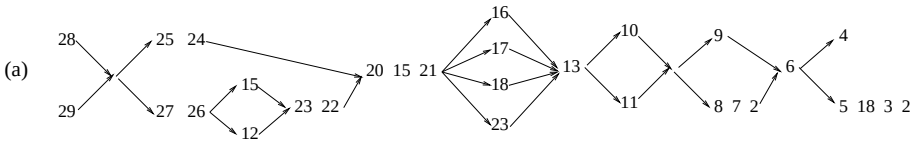
\ DAG Width Genome size \	2	3	4	5	6	7	8	9	10	11
30	100	100	98,15	97,41	94,33	96,18	93,18	95,54	100	100
50	100	98,04	96,43	97,62	95,25	98,61	100	86,96	95,26	94,94
80	100	98,21	97,90	87,54	96,79	93,89	100	95,83	98,33	100
100	100	98,81	95,65	96,83	89,70	93,95	90,38	95,30	94,63	94,95

Table 1. Percentage of O-adjacencies resulting from the heuristic's linearization compared to the optimal solution (obtained with the dynamic programming algorithm). Results are obtained by running the heuristic and dynamic programming algorithm on 10 different simulated DAGs for a given size and width.

Illustration on grass genomes: The Gramene database contains a large variety of maps of different grass genomes such as rice, maize and oats, and provides tools for comparing individual maps. A visualization tool allows to identify regions of 'homeology' between species, that is a linear series of markers in one genome that maps to a similar series of loci on another genome. Integrating marker orders between different studies remains a challenge to geneticists. However, as total orders are already obtained for widely studied species such as rice which has been completely sequenced, one can use this information to order markers on another species by using the adjacency maximization criterion.

Extracting the linear orders of markers using the Gramene visualization tool remains unpractical for hundreds of markers, as no automatic tool is provided for this purpose. We therefore illustrate the method on maps that are small enough to be extracted manually. Maize has been chosen instead of rice as it has shorter maps, though non-trivial, that can be represented graphically.

We used the "IBM2 Neighbors 2004" [15] map for chromosomes 5 (Figure 1) and 1 (Figure 5) of maize as a reference, and compared it with the "Paterson 2003" [4] and "Klein 2004" [13] maps of the chromosomes labeled C and LG-01, respectively, of sorghum. We extracted all markers of maize indicated as having a homolog in one of the databases of sorghum. All are found completely ordered in maize. This linear order is considered as the identity permutation. For markers of sorghum that are located on maize chromosome 5 (resp. 1), a total order maximizing the adjacency criterion is indicated in Figure 1.d (resp. Figure 5.b).



(b) 29 28 27 26 25 24 15 12 23 22 20 15 21 16 17 18 23 13 11 10 9 8 7 2 6 5 4 18 3 2

Fig. 5. (a) The partial order of markers in sorghum that are located and totally ordered on the maize chromosome 1; (b) A total order maximizing the adjacency criterion.

7 Conclusion

We have presented a detailed complexity result and algorithmic study for the problem of linearizing a partial order that is as close as possible to a given total order, in term of the breakpoint distance. Applications on the grass genomes show that this may be helpful to order unresolved sets of markers of some species using the totally ordered maps of well studies species such as rice. However, preliminary to the application of our algorithms is generating the appropriate partial orders. For this purpose, an automated preprocessing of the Gramene comparative database would be required to output the considered genetic maps, and then combine them on a single partial order. The absence of such tools prevented us from presenting more consequent applications.

The next step of this work will be to generalize our approach to two (or more) partial orders, as previously considered in [16, 18] for the reversal distance. As conjectured by Sankoff, an NP-complete result for this problem should be proved. A dynaming programming approach may also be envisaged for this case.

Considering the breakpoint (or similarly the adjacency) distance is a first step towards more general distances such as the number of conserved or common intervals [1, 3, 5]. Indeed, an adjacency of two genes is just a common interval of size 2. A simple extension of the greedy heuristic would be to order genes that remain unordered after maximizing adjacencies, by using the constraint of maximizing intervals of size 3, 4 and so on.

References

1. S. Bérard, A. Bergeron, and C. Chauve. Conservation of combinatorial structures in evolution scenarios. In *LNCS*, volume 3388 of *RECOMB 2004 sat-meeting comp. gen.*, pages 1 - 14. Springer, 2004.
2. A. Bergeron, J. Mixtacki, and J. Stoye. Reversal distance without hurdles and fortresses. volume 3109 of *LNCS*, pages 388 - 399. Springer-Verlag, 2004.
3. G. Blin and R. Rizzi. Conserved interval distance computation between non-trivial genomes. COCOON, 2005.
4. J.E. Bowers, C. Abbey, A. Anderson, C. Chang, X. Draye, A.H. Hoppe, R. Jessup, C. Lemke, J. Lennington, Z.K. Li, Y.R. Lin, S.C. Liu, L.J. Luo, BS. Marler,

R.G. Ming, S.E. Mitchell, D. Qiang, K. Reischmann, S.R. Schulze, D.N. Skinner, Y.W. Wang, S. Kresovich, K.F. Schertz, and A.H. Paterson. A high-density genetic recombination map of sequence-tagged sites for Sorghum, as a framework for comparative structural and evolutionary genomics of tropical grains and grasses. *Genetics*, 2003.

5. M. Figeac and J.S. Varré. Sorting by reversals with common intervals. In *LNBI*, volume 3240 of *WABI 2004*, pages 26 - 37. Springer-Verlag, 2004.
6. Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 1962.
7. M.D. Gale and K.M. Devos. Comparative genetics in the grasses. *Proceedings of the National Academy of Sciences USA*, 95:1971- 1974, 1998.
8. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
9. S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Journal of the ACM*, 48:1–27, 1999.
10. B.N. Jackson, S. Aluru, and P.S. Schnable. Consensus genetic maps: a graph theory approach. In *IEEE Computational Systems Bioinformatics Conference (CSB'05)*, pages 35- 43, 2005.
11. B. Keller and C. Feuillet. Colinearity and gene density in grass genomes. *Trends Plant Sci.*, 5:246- 251, 2000.
12. S.E Lander, P. Green, J. Abrahamson, and A. Barlow amd M.J Daly *et al.* MAPMAKER: an interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics*, 1:174 - 181, 1987.
13. M.A. Menz, R.R. Klein, J.E. Mullet, J.A. Obert, N.C. Unruh, and P.E. Klein. A High-Density Genetic Map of Sorghum Bicolor (L.) Moench Based on 2926 Aflp, Rflp and Ssr Markers. *Plant Molecular Biology*, 2002.
14. P.A. Pevzner and G. Tesler. Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *Proc. Natl. Acad. Sci. USA*, 100:7672 - 7677, 2003.
15. M.L. Polacco and Jr Coe E. IBM neighbors: a consensus GeneticMap. 2002.
16. D. Sankoff, C. Zheng, and A. Lenert. Reversals of fortune. *Proceedings of the 3rd RECOMB Comparative Genomics Satellite Workshop*, 3678:131–141, 2005.
17. I.V. Yap, D. Schneider, J. Kleinberg, D. Matthews, S. Cartinhour, and S. R. McCouch. A graph-theoretic approach to comparing and integrating genetic, physical and sequence-based maps. *Genetics*, 165:2235- 2247, 2003.
18. C. Zheng, Aleksander Lenert, and D. Sankoff. Reversal distance for partially ordered genomes. *Bioinformatics*, 21, 2005. in press.

8 Appendix

8.1 Proof of Theorem 1

First, let us present some interesting properties of any instance of Minimum-Breakpoint Linearization problem obtained by a MBL-construction. Then, we will use those properties to prove that the Minimum-Breakpoint Linearization problem is **NP**-complete. An illustration of a MBL-construction of a graph G of 6 vertices is illustrated in Figure 6.

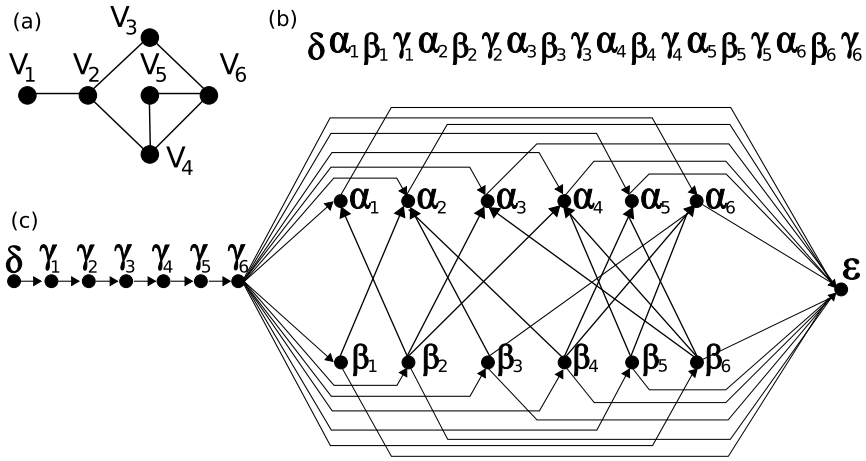


Fig. 6. Example of a MBL-constrction. The graph (a) is a connected graph of 6 vertices. The sequence (b) represents the complete order O and the graph (c) represents the partial order P obtained from the graph (a) by a MBL-construction.

Lemma 2. *Let $G = (V, E)$ be a graph and $P = (V_P, E_P)$ be a partial order obtained from G by a MBL-construction. There exists no linearization O' of P where both $\alpha_i <_{O'} \beta_i$ and $\alpha_j <_{O'} \beta_j$, for any $\alpha_i, \alpha_j, \beta_i, \beta_j$ of V_P such that $(v_i, v_j) \in E$.*

Proof. By contradiction, let us assume that there exists such a linearization O' where $\alpha_i <_{O'} \beta_i$ and $\alpha_j <_{O'} \beta_j$. Since $(v_i, v_j) \in E$, we have $(\beta_i, \alpha_j) \in E_P$ and $(\beta_j, \alpha_i) \in E_P$. Therefore, in any linearization of P – and consequently O' – $\beta_i \ll_{O'} \alpha_j$ and $\beta_j \ll_{O'} \alpha_i$ – which leads, by transitivity, to $\beta_i \ll_{O'} \alpha_i$; a contradiction. □

Lemma 3. *Let $G = (V, E)$ be a graph of n vertices, O and $P = (V_P, E_P)$ be respectively a complete and a partial order obtained from G by a MBL-construction. Given any linearization O' of P, $bkpts(O, O') = (3n+1) - k$ where k is the number of couples (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$.*

Proof. By construction, in O, (i) $\delta <_O \alpha_1 <_O \beta_1 <_O \gamma_1$, (ii) $\forall 1 < i \leq n$, $\gamma_{i-1} <_O \alpha_i <_O \beta_i <_O \gamma_i$ and (iii) $\gamma_n <_O \epsilon$. In any linearization O' of P, (i) $\delta <_{O'} \gamma_1$, (ii) $\forall 1 < i \leq n$, $\gamma_{i-1} <_{O'} \gamma_i$ and (iii) either $\alpha_i <_{O'} \epsilon$ or $\beta_i <_{O'} \epsilon$ for a given $1 \leq i \leq n$. Therefore, in any linearization O' of P, the only adjacencies that can be preserved are the ones of the form $\alpha_i <_O \beta_i$ for some $1 \leq i \leq n$. Let k be the number of couples (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$. If $k = 0$ then no adjacencies at all are preserved, therefore $bkpts(O, O') = (3n+1)$. And consequently, if $k > 0$ then $bkpts(O, O') = (3n + 1) - k$. □

We now turn to prove the following theorem.

Theorem 1. A connected graph $G = (V, E)$ admits an independent set of vertices $V' \subseteq V$ of cardinality greater than or equal to k if and only if there exists a linearization O' of P such that $bkpts(O, O') \leq (3n + 1) - k$, where O and P result from a MBL-construction of G.

Proof. ($\Rightarrow$) Let $V' \subseteq V$ such that $|V'| \geq k$ and V' is an independent set. Let O' be a linearization of P defined by $O' = \delta\ P_1\ P_2\ P_3\ P_4\ \epsilon$ where:

- P_1 is the linearization of the subset of vertices $V'_1 = \{\gamma_i | 1 \leq i \leq n\}$ such that $\forall 1 < i \leq n$, $\gamma_{i-1} <_{P_1} \gamma_i$;
- P_2 is **any** linearization of the subset of vertices $V'_2 = \{\beta_i | v_i \in V - V'\}$;
- P_3 is **any** linearization of the subset of vertices $V'_3 = \{\alpha_i, \beta_i | v_i \in V'\}$ such that $\forall v_i \in V'$, $\alpha_i <_{P_3} \beta_i$;
- P_4 is **any** linearization of the subset of vertices $V'_4 = \{\alpha_i | v_i \in V - V'\}$.

By Lemma 3, we can affirm that $bkpts(O, O') = (3n + 1) - |V'|$. Since, by hypothesis, $|V'| \geq k$, we obtain $bkpts(O, O') \leq (3n + 1) - k$.

($\Leftarrow$) Suppose we have a linearization O' of P such that $bkpts(O, O') \leq (3n + 1) - k$. Let $V' \subseteq V$ be the set of vertices such that:

$\forall(\alpha_i, \beta_i)$ such that $\alpha_i <_{O'} \beta_i$, add v_i to V'

By Lemma 2, we can affirm that V' is an independent set. Let us verify that $|V'| \geq k$. By Lemma 3, $Bkpts(O, O') = (3n + 1) - k$ where k is the number of couples (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$. Therefore, we obtain $|V'| = k$. □