Gene Maps Linearization using Genomic Rearrangement Distances

Guillaume Blin[*] Eric Blais[†] Danny Hermelin[‡] Pierre Guillon[§]
Mathieu Blanchette[¶] Nadia El-Mabrouk[‖]

Abstract

A preliminary step to most comparative genomics studies is the annotation of chromosomes as ordered sequences of genes. Different genetic mapping techniques often give rise to different maps with unequal gene content and 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. Our goal is to find a linearization of P that is as close as possible to O, in term of a given genomic distance. We first prove NP-completeness complexity results considering the breakpoint and the common interval distances. We then focus on the breakpoint distance and 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 and is empirically shown to produce solutions within 10% of the optimal solution, 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

[*]IGM-LabInfo, UMR CNRS 8049, Université de Marne-la-Vallée, France. gblin@univ-mlv.fr
[†]McGill Centre for Bioinformatics, McGill University, H3A 2B4, Canada. eblais@mcb.mcgill.ca
[‡]Department of Computer Science, University of Haifa, Mount Carmel, Haifa, Israel. danny@cri.haifa.ac.il
[§]IGM-LabInfo, UMR CNRS 8049, Université de Marne-la-Vallée, France. pguillon@univ-mlv.fr
[¶]McGill Centre for Bioinformatics, McGill University, H3A 2B4, Canada. blanchem@mcb.mcgill.ca
[‖]DIRO, Université de Montréal, H3C 3J7, Canada. mabrouk@iro.umontreal.ca

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 inversions, transpositions and translocations (Bergeron et al., 2004; Bourque et al., 2005b; El-Mabrouk, 2000; Hannenhalli and Pevzner, 1999; Pevzner and Tesler, 2003; Tang and Moret, 2003).

In a recent study, Sankoff *et. al.* generalized the rearrangement by reversal problem to handle two partial orders (Sankoff et al., 2005; Zheng et al., 2005). The idea was to find two total orders with the minimal reversal distance. The problem has been conjectured NP-hard, and a branch-and-bound algorithm has been developed for this purpose. The difficulty of this problem 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 (for example a completely sequenced genome), it can be used as a reference to order markers of closely related species.

Given a reference genome characterized by a total order O and a related genome characterized by a partial order P, the goal is to find a permutation consistent with P minimizing a given genomic distance with respect to O. We consider the breakpoint distance, which is the simplest measure of gene order conservation usually used as a first attempt to solve any genome rearrangement problem. Moreover, half the breakpoint distance gives a lower bound for the inversion distance. We also consider a more general measure of synteny, the common interval distance, that has been widely studied in the last two years (Blin and Rizzi, 2005; Figeac and Varré, 2004; Bérard et al., 2004; Bourque et al., 2005a; Blin et al., 2006).

After introducing the basic concepts in Section 2, we prove NP-complete results in Section 3. Section 4 then presents two exact dynamic programming algorithms for the breakpoint distance. The first algorithm applies to arbitrary partial orders. Its running time is exponential in the number of genes, but when the partial order is the intersection of a bounded number of genetic maps of bounded width, it runs in polynomial time. The second is a linear time algorithm that applies only to partial orders derived from a single genetic map. Section 5 then presents a fast and accurate heuristic for the general case. 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 genetic 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.

Formally, 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, indicating that the order of genes 4 and 5 is left undecided. 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 the 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), or the presence of multiple loci (markers that are assigned to different positions in the same map). These issues have been considered in previous studies (Yap et al., 2003; Zheng et al., 2005; Sankoff et al., 2005), and a software is available for combining genetic maps (Jackson et al., 2005). In this paper, we assume that the partial order P is already known.

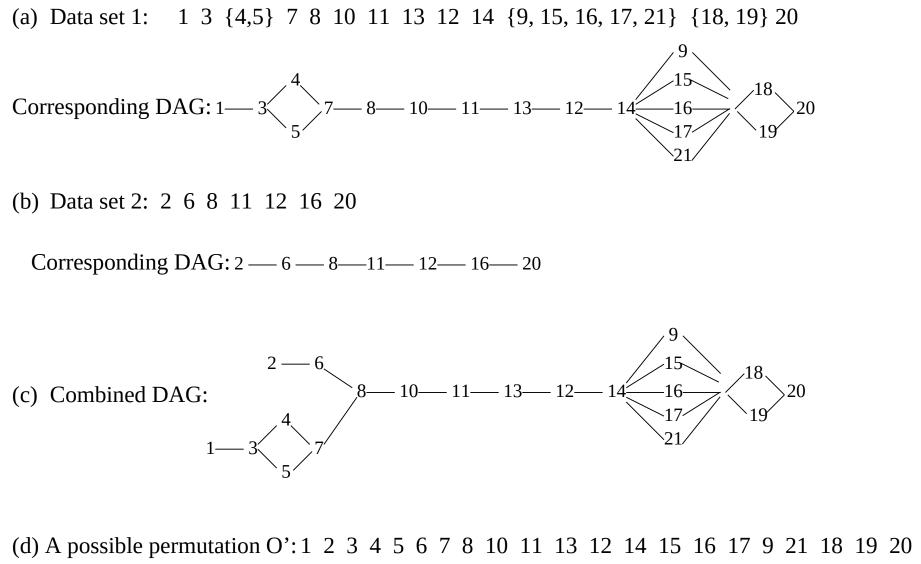

(a) Data set 1: 1 3 {4,5} 7 8 10 11 13 12 14 {9, 15, 16, 17, 21} {18, 19} 20
Corresponding DAG:
(b) Data set 2: 2 6 8 11 12 16 20
Corresponding DAG: 2 — 6 — 8 — 11 — 12 — 16 — 20
(c) Combined DAG:
(d) A possible permutation O': 1 2 3 4 5 6 7 8 10 11 13 12 14 15 16 17 9 21 18 19 20

Figure 1: Data extracted from the comparison of maize and sorghum in the Gramene database. The reference identity permutation O represents the order of markers in the "IBM2 neighbors 2004" map for maize chromosome 5. The corresponding marker's partial orders in sorghum are deduced from (a) "Paterson 2003" map of the chromosome labeled C and (b) "Klein 2004" map of the chromosome labeled LG-01; (c) is the partial order obtained by combining (a) and (b); (d) is a linearization of (c) minimizing the number of breakpoints.

A partial order P is represented as a DAG (directed acyclic graph) (V_P, E_P), where V_P is the set of vertices (genes) and E_P is the edge set representing the available order information (Figure 1 using data from "IBM2 neighbors 2004" (Polacco and E., 2002), "Paterson 2003" (Bowers et al., 2003) and "Klein 2004" (Menz et al., 2002)). E_P is a minimum set of edges, in the sense that an edge can not be deduced by transitivity from others.

2.1 Preliminary definitions

Let P be a partial order represented by a DAG (V_P, E_P). A vertex a is *P-adjacent* to a vertex b (denoted by $a <_P b$) if there is an edge from a to b, and *a precedes b* (denoted by $a \ll_P b$) if there is a directed path from a to b. The vertices a and b are *incomparable* (denoted by $a \sim_P b$) if neither $a \ll_P b$ nor $b \ll_P a$. A *total order* or *permutation* is just a partial order with no incomparable vertices.

A *linearization* of P is a permutation 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 permutation O on the same set of genes, our goal is to find a linearization O' of P, as close as possible to O. We consider two distance measures: the number $bkpts(O, O')$ of breakpoints of O' with respect to O, and the number $ICommon(O, O')$ of common intervals of O and O'.

Formally, a *breakpoint* of O' with respect to O is a pair (a, b) of vertices that are O'-adjacent but not O-adjacent. For example, the pair $(8, 10)$ is the leftmost breakpoint in the permutation O' (Figure 1.(d)) w.r.t the identity.

Let O and O' be two permutations on the set V_P of genes. A subset $\mathcal{V}$ of V_P is a *common interval* of O and O' if and only if both O and O' contain a sub-permutation (set of adjacent genes) whose gene content is exactly $\mathcal{V}$. In other words, the vertices in $\mathcal{V}$ are adjacent in both O and O', but not necessarily in the same order. For example, in Figure 1.(d), $\{10, 11, 13, 12, 14\}$ is a common interval of O and O'. In the following, a common interval is either represented as a set $\mathcal{V}$ of vertices or as an interval $[a, b]$ of O' where $a, b \in \mathcal{V}$, a precedes and b succeeds all the vertices of $\mathcal{V}$.

2.2 Problems

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

Maximum-Common Interval Linearization (MCIL) problem
Given: A partial order P and a permutation O on the set of genes $\{1, 2, \ldots, n\}$,
Find: A linearization of P into a permutation O' so that $ICommon(O, O')$ is maximized.

One may note that minimizing the number of breakpoints is equivalent to maximizing the number of O-adjacencies. Moreover, as an adjacency is a common interval of size 2, the MCIL problem is a generalization of the MBL problem.

W.l.o.g, we assume from now on that O is the identity permutation $(1, 2, \ldots, n)$.

Remark 1 *(***Signed genes and unequal gene content***) All hardness results and algorithmic solutions developed in this paper hold for signed genes as well, using the classical definition of breakpoints: a breakpoint of O' with respect to O is a pair (a, b) such that $a <_{O'} b$, but neither $a \not<_O b$ nor $-b \not<_O -a$. As for common intervals, the same definition holds in the case of signed permutations. Moreover, from a theoretical point of view, all algorithmic solutions developed are also applicable to a partial order and a permutation with unequal gene content. However, as the goal is to find an appropriate order of a genetic map G using a reference genome H, only common genes of G and H are of interest.*

3 Hardness results

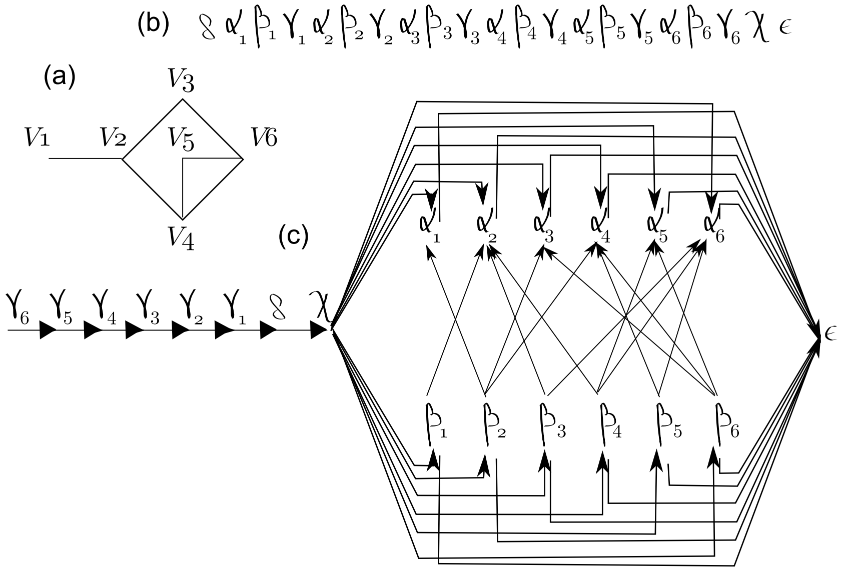
In this section, we prove that the decision version of both the MBL and the MCIL problems is **NP**-complete. The properties of partial orders linearization that are presented in this section and that are required to prove the hardness result are also useful in the understanding of the two exact dynamic programming algorithms presented in the next section.

We propose a reduction from the **NP**-complete problem Maximum Independent Set (Garey and Johnson, 1979): 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 and the MCIL problems are in **NP** since given a permutation O and a linearization O' of P, one can compute the number of breakpoints in linear time and the number of common intervals in quadratic time.

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 permutation O and the partial order P as follows. The permutation O is defined as a string $O = \delta\ \alpha_1\ \beta_1\ \gamma_1\ \alpha_2\ \beta_2\ \gamma_2\ \ldots\ \alpha_n\ \beta_n\ \gamma_n\ \chi\ \epsilon$, and the partial order P as a DAG $P = (V_P, E_P)$ with $V_P = \{\delta, \alpha_1, \alpha_2, \ldots, \alpha_n, \beta_1, \beta_2, \ldots, \beta_n, \gamma_1, \gamma_2, \ldots, \gamma_n, \chi, \epsilon\}$ and $E_P = \{(\gamma_1, \delta), (\delta, \chi)\} \cup \{(\gamma_{i+1}, \gamma_i) | 1 \leq i < n\} \cup \{(\chi, \alpha_i), (\chi, \beta_i) | 1 \leq i \leq n\} \cup \{(\beta_i, \alpha_j), (\beta_j, \alpha_i) | \forall (v_i, v_j) \in E\} \cup \{(\alpha_i, \epsilon), (\beta_i, \epsilon) | 1 \leq i \leq n\}$. In the following, we will refer to any such construction as a *PO-construction.* An illustration of a PO-construction of a graph G with 6 vertices is illustrated in Figure 2.

First, let us present some interesting properties of any instance (O, P) obtained by a PO-construction (Lemmas 1 and 2) relative to the breakpoints distance. Then, we will present other interesting properties (Lemmas 3 and 4) relative to common intervals. Finally, we will use those properties to prove that both the Minimum-Breakpoint Linearization and the Maximum-Common Interval Linearization problems are **NP**-complete.



(b) $\delta\ \alpha_1\ \beta_1\ \gamma_1\ \alpha_2\ \beta_2\ \gamma_2\ \alpha_3\ \beta_3\ \gamma_3\ \alpha_4\ \beta_4\ \gamma_4\ \alpha_5\ \beta_5\ \gamma_5\ \alpha_6\ \beta_6\ \gamma_6\ \chi\ \epsilon$
(a)
V3
V1
V2
V5
V6
V4
(c)
$\gamma_6\ \gamma_5\ \gamma_4\ \gamma_3\ \gamma_2\ \gamma_1\ \delta\ \chi$
$\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4\ \alpha_5\ \alpha_6$
ϵ
$\beta_1\ \beta_2\ \beta_3\ \beta_4\ \beta_5\ \beta_6$

Figure 2: Example of a PO-construction. The graph (a) is a connected graph of 6 vertices. The sequence (b) represents the permutation O and the graph (c) represents the partial order P obtained from the graph (a) by a PO-construction.

Lemma 1 *Let $G = (V, E)$ be a graph and $P = (V_P, E_P)$ be a partial order obtained from G by a PO-construction. There exists no linearization O' of P where both $\alpha_i <_{O'} \beta_i$ and*

$\alpha_j <_{O'} \beta_j$, for any $\alpha_i, \alpha_j, \beta_i, \beta_j \in V_P$ 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 2 *Let $G = (V, E)$ be a graph of n vertices, O and $P = (V_P, E_P)$ be respectively a permutation and a partial order obtained from G by a PO-construction. Given any linearization O' of P, $bkpts(O, O') = (3n + 2) - k$ where k is the number of pairs (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) $\gamma_n <_O \chi <_O \epsilon$. In any linearization O' of P, (i) $\gamma_1 <_{O'} \delta$, (ii) $\forall 1 < i \leq n$, $\gamma_i <_{O'} \gamma_{i-1}$, (iii) $\delta <_{O'} \chi$ and (iv) either $\alpha_j <_{O'} \epsilon$ or $\beta_j <_{O'} \epsilon$ for a given $1 \leq j \leq n$. 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 pairs (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$. If $k = 0$ then no adjacencies at all are preserved, therefore $bkpts(O, O') = (3n+2)$. Consequently, if $k > 0$ then $bkpts(O, O') = (3n + 2) - k$. □

Lemmas 1 and 2 will be used afterwords in the proof of the **NP**-completeness of the MBL-problem. We now state some properties relative to the types and maximum numbers of common intervals between the permutation and any linearization of the partial order obtained by a PO-construction.

Lemma 3 *Let $G = (V, E)$ be a graph of n vertices, O and $P = (V_P, E_P)$ be respectively a permutation and a partial order obtained from G by a PO-construction. Given any linearization O' of P, any non-trivial common interval between O and O' is one of the following: $\{[\delta, \chi], [\delta, \epsilon], [\alpha_i, \beta_i]\}$, where $1 \leq i \leq n$.*

Proof. Let us first characterize any common interval of size greater or equal to 3. By construction, considering O as the reference, any interval of that size is of one of the following forms: (i) $\{\alpha_j, \beta_j, \gamma_j\}$, (ii) $\{\beta_i, \gamma_i, \alpha_{i+1}\}$, (iii) $\{\gamma_i, \alpha_{i+1}, \beta_{i+1}\}$, (iv) $\{\gamma_n, \chi, \epsilon\}$, (v) $\{\delta, \alpha_1, \beta_1\}$ for $1 \leq j \leq n$ and $1 \leq i < n$.

Since, by construction, in any linearization O' of P $\gamma_i \ll_{O'} \delta <_{O'} \chi \ll_{O'} \{\alpha_i, \beta_i\}$ for $1 \leq i \leq n$, any common interval containing an α and a γ has to contain also χ and δ. Therefore, there is no common interval of type (i), (ii) or (iii); the smallest common interval including simultaneously an α and a γ is indeed $[\delta, \chi]$.

By construction, in any linearization O' of P, $\gamma_n \ll_{O'} \delta <_{O'} \chi$. Thus, any common interval of type (iv) has to contain also δ. Therefore, there is no common interval of type (iv); the smallest common interval including simultaneously γ_n, ϵ and χ is indeed $[\delta, \epsilon]$.

Finally, in any linearization O' of P, $\delta <_{O'} \chi \ll_{O'} \{\alpha_i, \beta_i\}$ for any $1 \leq i \leq n$. Therefore, any common interval of type (v) has to contain also χ. Therefore, there is no common interval of type (v); the smallest common interval including simultaneously δ and an α or a β is indeed $[\delta, \chi]$.

We just proved that any common interval of size greater or equal than 3 is either $[\delta, \chi]$ or $[\delta, \epsilon]$. Let us now characterize the common intervals of size 2. By construction any of these intervals is of one of the following forms: (i) $\{\delta, \alpha_1\}$, (ii) $\{\alpha_i, \beta_i\}$, (iii) $\{\beta_i, \gamma_i\}$, (iv) $\{\gamma_j, \alpha_{j+1}\}$, (v) $\{\gamma_n, \chi\}$, (vi) $\{\chi, \epsilon\}$ for $1 \leq i \leq n$ and $1 \leq j < n$.

By construction, in any linearization O' of P, $\gamma_i \ll_{O'} \delta <_{O'} \chi \ll_{O'} \{\alpha_i, \beta_i\}$ for any $1 \leq i \leq n$. Therefore, any common interval of type (i), (iii), (iv) or (v) has to contain also χ and δ. Therefore, there is no common interval of those types. Finally, in any linearization O' of P, $\chi \ll_{O'} \{\alpha_i, \beta_i\} \ll_{O'} \epsilon$. Whereas in O, $\beta_n <_{O'} \gamma_n <_{O'} \chi <_{O'} \epsilon$. Therefore, any common interval of type (vi) has to contain also δ and γ_n.

We have thus proved that given any linearization O' of P, any common interval of size at least two between O and O' is one of the following: $\{[\delta, \chi], [\delta, \epsilon], [\alpha_i, \beta_i]\}$ where $1 \leq i \leq n$. $\square$

Lemma 4 *Let $G = (V, E)$ be a graph of n vertices, O and $P = (V_P, E_P)$ be respectively a permutation and a partial order obtained from G by a PO-construction. Given any linearization O' of P, ICommon$(O, O') = k + 3n + 5$ where k is the number of pairs (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$ or $\beta_i <_{O'} \alpha_i$.*

Proof. Let $(v_i, v_j) \in E$. By construction, in P we have $\beta_j <_P \alpha_i$ and $\beta_i <_P \alpha_j$. Let O' be a linearization of P s.t. $\alpha_i <_{O'} \beta_i$. In O', $[\alpha_i, \beta_i]$ is thus a common interval between O and O'. We will prove that $[\alpha_j, \beta_j]$ cannot be a common interval between O and O'.

Indeed, there are three cases: $\alpha_j <_{O'} \beta_j$, $\beta_j <_{O'} \alpha_j$ or none of those two. By Lemma 1, the first case is not possible. Therefore, consider first that $\beta_j <_{O'} \alpha_j$. Then, in O' we have $\beta_j \ll_{O'} \alpha_i <_{O'} \beta_i \ll_{O'} \alpha_j$. Consequently, $[\alpha_j, \beta_j]$ cannot be a common interval. Similarly, consider that in O', $\beta_i <_{O'} \alpha_i$. In O', $[\alpha_i, \beta_i]$ is a common interval between O and O'. If $\alpha_j <_{O'} \beta_j$, in O' we have $\beta_i \ll_{O'} \alpha_j <_{O'} \beta_j \ll_{O'} \alpha_i$; $[\alpha_i, \beta_i]$ is not a common interval anymore, a contradiction. If $\beta_j <_{O'} \alpha_j$ then by construction we have $\beta_j \ll_{O'} \beta_i <_{O'} \alpha_i \ll_{O'} \alpha_j$. In both cases, $[\alpha_j, \beta_j]$ cannot be a common interval. Therefore, by Lemma 3, in any linearization of P, there are $k + 2 + 3(n + 1)$ common intervals where k is the number of pairs (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$ or $\beta_i <_{O'} \alpha_i$. $\square$

We now turn to the proof of the following theorems.

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$(O, O') \leq (3n + 2) - k$, where O and P result from a PO-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 $O' = P_1\ \delta\ \chi\ P_2\ P_3\ P_4\ \epsilon$ where:

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

For example, given the instances illustrated in Figure 2 and an independent set $V' = \{1, 3, 4\}$, O' is defined by:
$O' = \gamma_6\ \gamma_5\ \gamma_4\ \gamma_3\ \gamma_2\ \gamma_1\ \delta\ \chi\ \beta_2\ \beta_5\ \beta_6\ \alpha_1\ \beta_1\ \alpha_3\ \beta_3\ \alpha_4\ \beta_4\ \alpha_2\ \alpha_5\ \alpha_6\ \epsilon$.

By Lemma 2, we can affirm that $bkpts(O, O') = (3n + 2) - |V'|$. Since, by hypothesis, $|V'| \geq k$, we obtain $bkpts(O, O') \leq (3n + 2) - k$.

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

Theorem 2 *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 $ICommon(O, O') \geq k + 3n + 5$, where O and P result from a PO-construction of G.*

Proof. ($\Rightarrow$) The proof is almost the same as for Theorem 1. Let $V' \subseteq V$ such that $|V'| \geq k$ and V' is an independent set. Let O' be a linearization of P defined as in the proof of Theorem 1 (i.e. $O' = P_1\ \delta\ \chi\ P_2\ P_3\ P_4\ \epsilon$).

By Lemma 4, we can affirm that $ICommon(O, O') = |V'| + 5 + 3n$. Since, by hypothesis, $|V'| \geq k$, we obtain $ICommon(O, O') \geq k + 5 + 3n$.

($\Leftarrow$) Suppose we have a linearization O' of P such that $ICommon(O, O') \geq k + 5 + 3n$. Let $V' \subseteq V$ be the set of vertices such that:

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

By Lemma 3, we can affirm that V' is an independent set. Let us verify that $|V'| \geq k$. By Lemma 4, $ICommon(O, O') = k + 5 + 3n$ where k is the number of pairs (α_i, β_i) such that $\alpha_i <_{O'} \beta_i$ or $\beta_i <_{O'} \alpha_i$. Therefore, we obtain $|V'| = k$ □

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 bounded set of genetic maps of bounded width. 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 *maximal border* iff any other vertex of V_P is comparable to at least one vertex of A. We also define, for any subset $B \subseteq V_P$, $front(B) = \{x \in B : x$ has no successor in $B\}$. Finally, for any subset $A \subseteq V_P$, we denote $pred(A) = A \cup \{x \in V_P : \exists y \in A$ s.t. $x \ll_P y\}$.

4.1 A dynamic algorithm for arbitrary partial orders

Let A be a 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 ordering 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 3 *For any border A and any vertex $i \in A$,*

$$X_{A,i} = \max_{j \in A'} X_{A',j} + \left\{ \begin{array}{ll} 1 & \textit{if } |j-i| = 1 \\ 0 & \textit{otherwise} \end{array} \right.$$

where

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

A recursive algorithm follows from the previous theorem. The recursion begins with $A = F = front(V_P)$ and stops as soon as A is the empty set.

Computing each entry 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 a small number m of genetic maps, where each map contains a maximum of q blocks and the size of each block is at most some small number 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 $b(P) \in O(q^m \cdot 2^{km})$. 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 built from a single 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 4 *Define $X_0 = 0$ and $L_0 = \{\}$. Then, for any $1 \le i \le q$,*

$$X_i = X_{i-1} + |g_1(B_i,B_i)| + \left\{ \begin{array}{lll} 1 & , & \textit{if } |g_2(L_{i-1},B_i)| \ge 1 \\ 0 & , & \textit{otherwise} \end{array} \right.$$

and

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

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 4 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 for the general problem 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 intersection of an arbitrary number of maps. It aims to find a maximum number of O-adjacencies consistent 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.

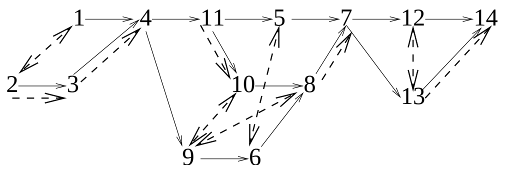


Figure 3: Dotted edges are all O-adjacencies that can, individually, be part of a linearization of P. A bi-directional edge represents the superposition of two edges, one in each direction. An adjacency path of P is a directed sequence of consecutive dotted edges.

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 3, $(1, 2, 3, 4)$ (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'. For example, $(1, 2, 3, 4)$ is a valid path of the partial order in Figure 3. Lemma 5 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 5 *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 $i + j_1$).*

```
Algorithm Find-Valid-Direct-Path (P)
{Compute the list $L$ of all adjacency paths of size 2}
For $i = 1$ to $|V|$ 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 ($L_1$);
```

Figure 4: 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.

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 matrix M^T of size $|V_P| \times |V_P|$ verifying, for any $i, j \in V_P$,
$$M^T(i, j) = \begin{cases} 1 & \text{iff } i <_P j \\ 2 & \text{iff } i \ll_P j \text{ but } i \not<_P j \\ 0 & \text{otherwise} \end{cases}$$
M^T is computed from M using the Floyd-Warshall algorithm (Floyd, 1962).

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 (see details below).
- **Step 2:** Incorporate the new adjacencies in M^T, and compute the transitive closure of M^T.

Algorithm 4 describes the search of the longest valid direct path. Valid direct 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 $i+k$.

The algorithm for valid indirect paths is obtained by replacing the three first lines of Algorithm 4 by the following:

```
*For* $i = |V|$ to 1 *do*
    *If* $(i <_P i-1)$ or ($i$ and $i-1$ are incomparable) *then*
        Add $(i, i-1)$ to $L$;
```

Step 1 consists in running successively both algorithms for direct and indirect paths, and taking the longest resulting path.

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 (Floyd, 1962) 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 used simulated data to assess the performance of our greedy algorithm. 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 (Lander et al., 1987; Yap et al., 2003). For example, the DAG in Figure 1.c has the following string representation:

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

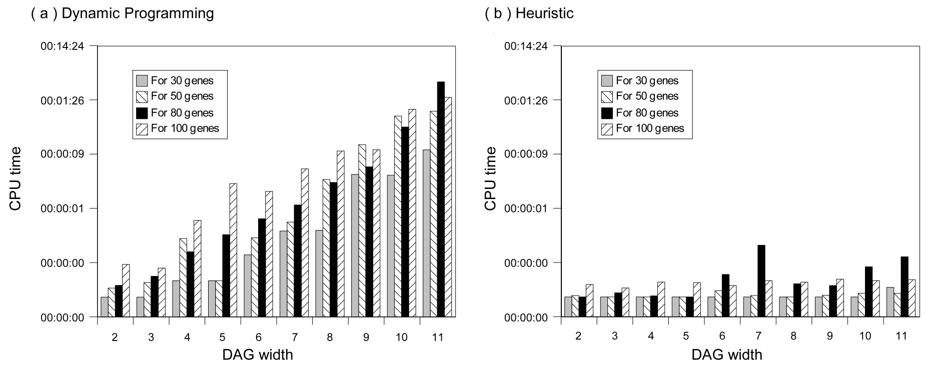


Figure 5: CPU time expended by (a) the dynamic programming algorithm and (b) the heuristic, for DAGs of a given size and width. Each result is obtained from 10 runs (10 different simulated DAGs). The Y axis is logarithmic.

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.

Both the heuristic and the dynamic programming algorithm were run on the dataset described above. We evaluated two criteria: the CPU time and the number of breakpoints (or similarly adjacencies) induced by the returned linearization.

Figure 5 shows that the running time of the dynamic programming algorithm grows exponentially with the width of the DAG (defined as the size of the DAG's largest border), while the heuristic is not affected by it (this was expected, as the time-complexity only depends on the DAG's size). Moreover, the greedy heuristic can easily handle partial orders consisting of thousands of genes, as illustrated in Figure 6.

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 greedy algorithm always returns a linearization containing at least 90% of the adjacencies of the optimal solution, and usually close to 100%.

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

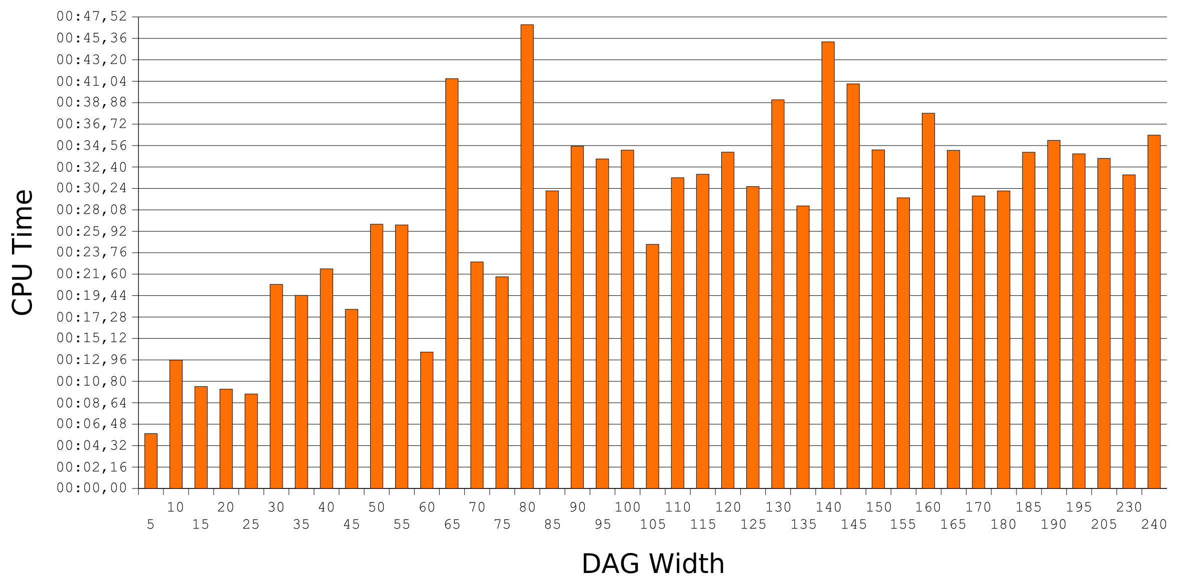


Figure 6: CPU time expended by the heuristic, for DAGs of 1000 vertices with different widths. Each result is obtained from 10 runs (10 different simulated DAGs).

		DAG Width									
		2	3	4	5	6	7	8	9	10	11
Genome size	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.

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" (Polacco and E., 2002) map for chromosomes 5 (Figure 1) and 1 (Figure 7) of maize as a reference, and compared it with the "Paterson 2003" (Bowers et al., 2003) and "Klein 2004" (Menz et al., 2002) 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 7.b).

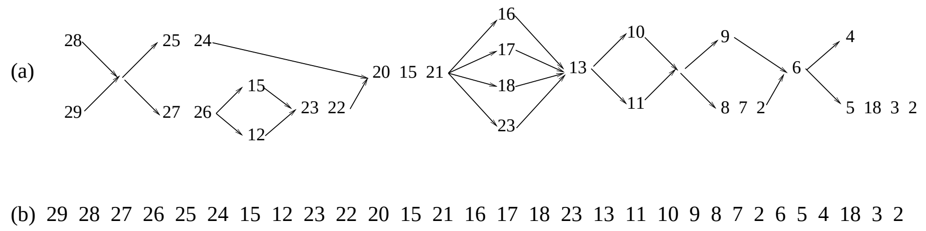

(a)
28 29 25 27 24 26 15 12 23 22 20 15 21 16 17 18 23 13 10 11 9 8 7 2 6 4 5 18 3 2
(b) 29 28 27 26 25 24 15 12 23 22 20 15 21 16 17 18 23 13 11 10 9 8 7 2 6 5 4 18 3 2

Figure 7: (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 and common intervals distances. 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 studied 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 suitable applications.

The next step of this work will be to generalize our approach to two (or more) partial orders, as previously considered in (Sankoff et al., 2005; Zheng et al., 2005) for the reversal distance. As conjectured by Sankoff, an NP-complete result for this problem should be proved. A dynamic programming approach may also be developed for this case.

An adjacency of two genes being 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. An efficient heuristic has to be found for the problem of linearizing a partial order considering the maximal number of common intervals as a criterion.

References

Bérard, S., Bergeron, A., and Chauve, C. (2004). Conservation of combinatorial structures in evolution scenarios. In *RECOMB 2004 Satellite meeting on Comparative Genomics*, volume 3388 of *LNCS*, pages 1 - 14. Springer.

Bergeron, A., Mixtacki, J., and Stoye, J. (2004). Reversal distance without hurdles and fortresses. In *15th Symposium on Combinatorial Pattern Matching*, volume 3109 of *LNCS*, pages 388 - 399. Springer.

Blin, G., Chateau, A., Chauve, C., and Gingras, Y. (2006). Inferring positional homologs with common intervals of sequences. In *Fourth Annual RECOMB Satellite meeting on Comparative Genomics*, volume 4205 of *LNCS/LNBI*, pages 24 - 38.

Blin, G. and Rizzi, R. (2005). Conserved interval distance computation between non-trivial genomes. In *Proc. 11th International Computing and Combinatorics Conference*

(COCOON'05), volume 3595 of *LNCS*, pages 22–31.

Bourque, G., Yacef, Y., and El-Mabrouk, N. (2005a). Maximizing synteny blocks to identify ancestral homologs. In *Third Annual RECOMB Satellite meeting on Comparative Genomics*, volume 3678 of *LNCS/LNBI*, pages 21 - 34.

Bourque, G., Zdobnov, E., Bork, P., Pavzner, P., and Tesler, G. (2005b). Comparative architectures of mammalian and chicken genomes reveal highly variable rates of genomic rearrangements across different lineages. *Genome Research*, 15:98- 110.

Bowers, J., Abbey, C., Anderson, A., Chang, C., Draye, X., Hoppe, A., Jessup, R., Lemke, C., Lennington, J., Li, Z., Lin, Y., Liu, S., Luo, L., Marler, B., Ming, R., Mitchell, S., Qiang, D., Reischmann, K., Schulze, S., Skinner, D., Wang, Y., Kresovich, S., Schertz, K., and Paterson., A. (2003). 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*, 165:367-386.

El-Mabrouk, N. (2000). Sorting signed permutations by reversals and insertions/deletions of contiguous segments. *Journal of Discrete Algorithms*, 1(1):105-122.

Figeac, M. and Varré, J. (2004). Sorting by reversals with common intervals. In *WABI*, volume 3240 of *LNBI*, pages 26 - 37. Springer-Verlag.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

Hannenhalli, S. and Pevzner, P. A. (1999). Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Journal of the ACM*, 48:1–27.

Jackson, B., Aluru, S., and Schnable, P. (2005). Consensus genetic maps: a graph theory approach. In *IEEE Computational Systems Bioinformatics Conference (CSB'05)*, pages 35- 43.

Lander, S., Green, P., Abrahamson, J., and amd M.J Daly *et al.*, A. B. (1987). MAPMAKER: an interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics*, 1:174 - 181.

Menz, M., Klein, R., Mullet, J., Obert, J., Unruh, N., and Klein, P. (2002). A High-Density Genetic Map of Sorghum Bicolor (L.) Moench Based on 2926 Aflp, Rflp and Ssr Markers. *Plant Molecular Biology*, 48:483–99.

Pevzner, P. and Tesler, G. (2003). Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *Proc. Natl. Acad. Sci. USA*, 100:7672 - 7677.

Polacco, M. and E., J. C. (2002). IBM neighbors: a consensus GeneticMap.

Sankoff, D., Zheng, C., and Lenert, A. (2005). Reversals of fortune. *Proceedings of the 3rd RECOMB Comparative Genomics Satellite Workshop*, 3678:131–141.

Tang, J. and Moret, B. (2003). Phylogenetic reconstruction from gene rearrangement data with unequal gene contents. In *Lecture Notes in Computer Science*, volume 2748 of *WADS'03*, pages 37- 46. Springer Verlag.

Yap, I., Schneider, D., Kleinberg, J., Matthews, D., Cartinhour, S., and McCouch, S. R. (2003). A graph-theoretic approach to comparing and integrating genetic, physical and sequence-based maps. *Genetics*, 165:2235- 2247.

Zheng, C., Lenert, A., and Sankoff, D. (2005). Reversal distance for partially ordered genomes. *Bioinformatics*, 21, Suppl 1:i502–i508.