Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS 2010)

Genome Rearrangement and Planning: Revisited

Tansel Uras and **Esra Erdem**
Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey

Abstract

Evolutionary trees of species can be reconstructed by pairwise comparison of their entire genomes. Such a comparison can be quantified by determining the number of events that change the order of genes in a genome. Earlier Erdem and Tillier formulated the pairwise comparison of entire genomes as the problem of planning rearrangement events that transform one genome to the other. We reformulate this problem as a planning problem to extend its applicability to genomes with multiple copies of genes and with unequal gene content, and illustrate its applicability and effectiveness on three real datasets: mitochondrial genomes of *Metazoa*, chloroplast genomes of *Campanulaceae*, chloroplast genomes of various land plants and green algae.

Introduction

In biology, evolutionary trees (or phylogenies) can be reconstructed from the comparison of genomes of species (Sankoff and Blanchette 1998). One metric of evolutionary distance for this purpose is the number of rearrangement events such as transpositions and inversions to convert one genome to the other where a smaller number of such events implies a closer lineage. A rearrangement event is a genome-wide mutation that changes the order, orientations and/or existence of genes in a genome. Finding the minimum number of these rearrangement events between genomes is called the genome rearrangement problem and it is conjectured to be NP-hard (Bylander 1994).

We consider the genome rearrangement problem as a planning problem as in (Erdem and Tillier 2005): one of the genomes is represented as the initial state and the other one as the goal state; the planner is prompted to find a sequence of at most k actions (rearrangement events) that leads the initial state to the goal state. As in (Erdem and Tillier 2005), we describe the genome rearrangement problem in ADL (Pednault 1989), and use TLPLAN (Bacchus and Kabanza 2000) to compute solutions. Our formulation of the genome rearrangement problem differs from that of (Erdem and Tillier 2005) in the following ways. First of all, it extends the descriptions of genomes to be able to handle duplicate genes—genes that occur multiple times in a single genome. Accordingly, it not only extends the descriptions of transpositions, inversions, inverted transpositions (transversions) but also introduces new operators for insertions and deletions. The temporal control information is described as preconditions of events. As observed in (Rintanen 2000; Gabaldon 2003), such a modification improves the computational efficiency. Also, the goal-check is done in a more computationally-efficient way by means of a biologically motivated measure, called the *breakpoint distance*, which does not require us to check the whole gene orders.

The main contribution of our work are as follows:

- We have introduced a computational method that can solve the genome rearrangement problem with duplicates and unequal gene content. Although the genomes of many species (in particular, the chloroplast genomes) contain duplicate genes, no existing genome rearrangement software (e.g., GRIMM (Tesler 2002), GRAPPA (Moret et al. 2001), DERANGE 2 (Blanchette, Kunisawa, and Sankoff 1996)) can handle them.
- One method to handle duplications without loss of information is to treat them as different genes and solve the problem for each possible relabeling of these genes (Cui et al. 2006); however, there are exponentially many possible relabelings in the number of occurrences of the duplicate genes. We have introduced a new method to handle duplicates without such enumeration of relabelings, by identifying the duplicates upfront and introducing a 0-cost auxiliary action.
- We have illustrated the applicability and the effectiveness of our planning-based approach to genome rearrangement on three sets of real data: mitochondrial genomes of *Metazoa* (animals with a nervous system, and muscles) (Blanchette, Kunisawa, and Sankoff 1999), chloroplast genomes of *Campanulaceae* (flowering plants) (Cosner et al. 2000), and chloroplast genomes of various land plants and green algae (Cui et al. 2006). Our results conform with the most recent and widely accepted results.

Genome Rearrangement Problem

The *genome* of a single-chromosome organism can be represented by circular configurations of numbers $1, \ldots, n$, with a sign $+$ or $-$ assigned to each of them. For instance, Figure 1(a) shows a genome for $n = 5$. Numbers $\pm 1, \ldots, \pm n$

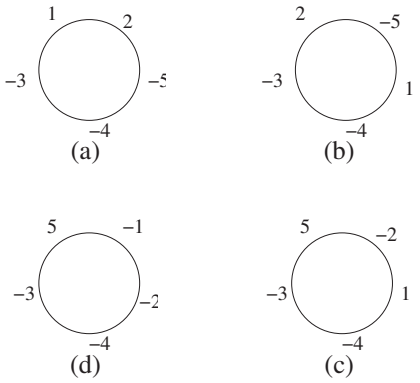

Figure 1: (a) A genome; (b) a transposition of (a); (c) an inversion of (b); (d) a transversion of (c).

will be called *labels*. Intuitively, a label corresponds to a gene, and its sign corresponds to the orientation of the gene. By $(l_1, \ldots, l_n)$ we denote the genome formed by the labels $l_1, \ldots, l_n$ ordered clockwise. For instance, each of the expressions $(1, 2, -5, -4, -3), (2, -5, -4, -3, 1), \ldots$ denotes the genome in Figure 1(a).

About genomes g, g' we say that g' is a *transposition* of g (or *can be obtained from g by a transposition*) if, for some labels $l_1, \ldots, l_n$ and numbers k, m $(0 < k, m \leq n)$,

$$g = (l_1, \ldots, l_n)$$
$$g' = (l_k, \ldots, l_m, l_1, \ldots, l_{k-1}, l_{m+1}, \ldots, l_n).$$

For instance, the genome in Figure 1(b) is a transposition of the genome in Figure 1(a).

Similarly, about genomes g, g', we say that g' can be obtained from g by a *deletion* (or g can be obtained from g' by an *insertion*) if, for some labels $l_1, \ldots, l_n$ and a number m $(0 < m \leq n)$,

$$g = (l_1, \ldots, l_n)$$
$$g' = (l_1, \ldots, l_{m-1}, l_{m+1}, \ldots, l_n).$$

Other events, inversions and transversions, can be defined as in (Erdem and Tillier 2005).

We say that there is a *breakpoint* between two genomes g and g' at the pair l, l' if g includes the pair l, l' and g' includes neither the pair l, l' nor the pair $-l', -l$. For instance, there are 3 breakpoints between $(1, 2, 3, 4, 5)$ and $(1, 2, -5, -4, 3)$. The number of breakpoints between two genomes is called their *breakpoint distance*.

The *genome rearrangement problem* can be defined as follows: given two genomes g and g', and a positive integer k, decide whether g' can be obtained from g by at most k successive events. We view the genome rearrangement problem as a planning problem:

> given two genomes g and g', and a nonnegative integer k, find a sequence of at most k events that reduces the number of breakpoints between g and g' to 0.

Note that this planning problem is different from the one described in (Erdem and Tillier 2005) in that both genomes are specified in the initial world, and that the goal is specified in terms of the number of breakpoints.

Representing the Planning Problem

We represent a genome by specifying the clockwise order of its labels, for instance, by a fluent of the form $cw(L, L1)$ which expresses that label $L1$ comes after label L in clockwise direction. However, such a representation alone is not sufficient to describe genomes with duplicate genes. For example, the genome $(1, 2, 3, 2, 4)$ can be represented by the atoms: $cw(1, 2)$, $cw(2, 3)$, $cw(3, 2)$, $cw(2, 4)$, $cw(4, 1)$. Here, $(1, 2, 4)$ can be erroneously considered as a subsequence of the genome. For this reason, we treat duplicate genes as different genes but also keep track of them. To identify which genes are duplicates, we introduce a predicate $duplicate$. For example, the genome $(1, 2, 3, 2, 4)$ can be represented by the atoms: $cw(1, 2)$, $cw(2, 3)$, $cw(3, 5)$, $cw(5, 4)$, $cw(4, 1)$, $duplicate(2, 5)$.

We view the genome rearrangement problem as a planning problem, as described in the previous section. Since both genomes are described in the initial state, we introduce two fluents to describe their gene orders: $cw(L, L1)$ and $other(L, L1)$. The rearrangement events are applied to the genome described by cw atoms. To describe the goal, we introduce a functional fluent $bpcount$ to denote the number of breakpoints: initially, it is counted; after that, at each step, it is decreased by the application of a rearrangement event. For instance, suppose that we are given two genomes, $(1, 2, 3, 2, 4)$ and $(1, 4, -3, -2)$. In the corresponding planning problem, the initial state is described as follows:

```
(define (initial0)
  (cw 1 2) (cw 2 3) (cw 3 5)
  (cw 5 4) (cw 4 1) (duplicate 2 5)
  (other 1 4) (other 4 -3)
  (other -3 -2) (other -2 1))
```

and the goal as follows:

```
(define (goal0) (= (bpcount) 0))
```

in the language of TLPLAN. The maximum plan length k is set to 3 by the fact

```
(set-initial-facts (= k 3)) .
```

With the description of this planning problem

```
(set-initial-world (initial0))
(set-goal (goal0))
```

TLPLAN computes the following 2-step plan

```
(transvert 2 3 4) (delete 5)
```

according to which the genome $(1, 2, 3, 2, 4)$ can be transformed to $(1, 4, -3, -2)$ as follows: first $2, 3$ is inverted and then inserted after 4, next the first appearance of 2 is deleted. Here, by default, the cost of each action is 1, and depth-best-first search strategy is applied.

Representing Rearrangement Events

We introduce five actions to describe transpositions, inversions, transversions, insertions and deletions, and represent them as ADL-style operators in the language of TLPLAN. These operators are applied to the genome described by cw.

Consider the action $transpose(X, Y, Z)$ ("the gene sequence starting with the gene X and ending at the gene Y is inserted after gene Z") that describes a transposition. We represent this action in the language of TLPLAN as follows. Let us start with its straightforward formulation in (Erdem and Tillier 2005):

```
(def-adl-operator (transpose ?x ?y ?z)
  ; preconditions
  (pre (?x) (label ?x)
       (?y) (label ?y)
       (?z) (label ?z)
       (cantranspose ?x ?y ?z))
  ; insertion of ?x ?y after ?z
  ; in (?x1,?x..?y,?y1..?z,?z1,...)
  ; is (?x1,?y1..?z,?x..?y,?z1,...)
  (exists (?x1) (cw ?x1 ?x)
    (?y1) (cw ?y ?y1)
    (?z1) (cw ?z ?z1)
    (and (add (cw ?x1 ?y1) (cw ?z ?x)
              (cw ?y ?z1))
         (del (cw ?x1 ?x) (cw ?y ?y1)
              (cw ?z ?z1))))).
```

where `(cantranspose ?x ?y ?z)` is defined as follows:

```
(def-defined-predicate (cantranspose ?x ?y ?z)
  (and (< (plan-length) (k))
    (not (= ?x ?z)) (not (= ?y ?z))
    (not (cw ?z ?x)) (notbetween ?z ?x ?y))).
```

In (Erdem and Tillier 2005) the breakpoint heuristic—the number of breakpoints should decrease at the next step—is described as part of temporal control. As mentioned in the introduction, we enforce this heuristic by defining the preconditions cleverly so as to reduce a breakpoint in each action. For example, we extend the preconditions of transposition with the following conditions:

```
(not (goodbefore ?x)) (not (goodafter ?y))
(not (goodafter ?z))
(or (goodlink ?z ?x)
  (exists (?t) (cw ?z ?t) (goodlink ?y ?t)))
```

The first three conditions prevent a transposition from breaking any existing links that are not breakpoints: if `?x1 ?x`, `?y ?y1`, or `?z ?z1` is a good link (i.e., it is a subsequence of the gene order of the other genome described by `other`) then a transposition is not applicable. The disjunction enforces a transposition to relieve at least one breakpoint by forming a good link. Accordingly, the effects of transposition are modified to take into account the change in the number of breakpoints:

```
(add (= (bpcount)
        (- (bpcount)
           (relievedbp-transpose ?x ?y ?z))))
```

where `relievedbp-transpose` calculates the number of breakpoints that will be eliminated by a transposition. We define `relievedbp-transpose` as follows:

```
(def-defined-function
  (relievedbp-transpose ?x ?y ?z)

  (local-vars ?count)
  (and (:= ?count 0)
     (implies (goodlink ?z ?x)
        (:= ?count (+ 1 ?count)))
     (exists (?t) (cw ?z ?t)
        (implies (goodlink ?y ?t)
          (:= ?count (+ 1 ?count))))
     (exists (?x1) (cw ?x1 ?x)
        (?y1) (cw ?y ?y1)
        (implies (goodlink ?x1 ?y1)
          (:= ?count (+ 1 ?count))))
     (:= relievedbp-transpose ?count)))
```

This function counts the number of new good links formed after a transposition. It is important to emphasize here that, with `relievedbp-transpose`, breakpoints are not counted from the scratch at each step: they are counted initially, and after that the number of breakpoints is decreased by each applied operator.

We describe the other events similarly. Their descriptions will be provided in an electronic appendix.

Improvements over the Representation

We have improved the representation of the planning domain even further to get more accurate solutions more efficiently (in terms of computation time and the memory used).

Discarding redundant labels: To improve the computational efficiency, we can remove redundant labels along the way. For instance, a label is redundant if it has formed the two good links as expected, i.e., it is in its goal position. Redundant labels can be discarded by modifying the delete effects of the operators. Discarding redundant labels reduce the search space since all operators are using the label information in their preconditions. With this method, the computation time has decreased drastically and some large instances (e.g., *Chlorella* and *Chlamydomonas*) that required too much memory could be solved.

Swapping duplicates: Simply renaming the duplicate genes is something we are trying to avoid since different renamings can lead to different length plans. To be able to find short plans, we introduce an auxiliary action, $swap(x, y)$, for swapping duplicates. The idea is to assign a cost of 1 to every action, and 0 cost to $swap$, and then to try to find a plan of minimum cost. In the formulation of $swap$, we enforce it to relieve a breakpoint, so there is no risk of getting stuck in an infinite loop.

Assigning priorities to events: To get more accurate results we incorporate biological information in search by assigning priorities to operators in our formulation. For instance, transpositions may occur more often in some species, then we can define its priority taking into account also the number of relieved breakpoints. Assigning higher priorities to events that are observed to occur more often has led to more accurate results (e.g., grouping of chordates with echinoderms) in mitochondrial genomes of *Metazoa*. Assigning priorities has a similar effect of guiding the search by assigning costs to events as in (Erdem and Tillier 2005), under the same depth-first search strategy.

Experimental Results

Before we solve the genome rearrangement problem instances, we apply some preprocessing to reduce their sizes. First, we apply "safe" deletions (resp. insertions) of genes in the genome g described by cw atoms: if a gene is present in g (resp. g') only, we delete (resp. insert) it. After that, we apply "condensing": we identify the common subsequences in the genomes and replace them by some new identifiers.

We have experimented with three sets of data using TLPLAN: *Metazoan* mitochondrial genomes (Blanchette, Kunisawa, and Sankoff 1999), *Campanulaceae* chloroplast genomes (Cosner et al. 2000), and chloroplast genomes of

various land plants and green algae (Cui et al. 2006). Only in the last data set, genomes are of unequal content with duplicate genes.

To analyze the accuracy of our approach, for each data set, first we have computed a small number of events for each pair of genomes and constructed a distance matrix, and then we have constructed a phylogeny using the distance matrix program NEIGHBOR (Felsenstein 2009).

In all experiments, TLPLAN is run with the depth-best-first search strategy. The cost of each action is 1 (except the 0-cost action of swapping duplicates); so the goal is to find a plan with a small cost (rather than a shortest plan). The priorities of insertions, deletions, and swaps are much higher than the other events.

Mitochondrial genomes of *Metazoa*: Each one of these 11 genomes consists of 36 genes. The priorities of transpositions, inversions, transversions are specified as 2, 1, 1 respectively. All 45 plans (each with 1–26 events) are computed in less than 3 minutes. The phylogeny constructed by NEIGHBOR groups chordates and echinoderms together, arthropods, molluscs and annelids together; nematodes are a sister to these two groupings. These results conform with the results of (Nielsen 2001) based on morphological data. Groupings of chordates and echinoderms, and molluscs and annelids also conform with the most widely accepted view of Metazoan Systematics and Tree of Life, based on the analysis of molecular data (18S rRNA sequences).

Chloroplast genomes of *Campanulaceae*: We consider 13 genomes, each with 105 genes. The priorities of transpositions, inversions, transversions are specified as 2, 3, 4 respectively (since inversions often occur in chloroplast genomes). All 66 plans (each with 1–12 events) are computed in less than 1 minute. According to the phylogeny constructed by NEIGHBOR, the groupings are identical to the ones in the consensus tree presented in Figure 4 of (Cosner et al. 2000). The major division between the grouping of *Codonopsis*, *Cyananthus* and *Platycodon*, and the others conform with the most recent results (Cosner, Raubeson, and Jansen 2004) based on the sequence analysis; also this division corresponds to the distribution of pollen morphology characteristics, unlike the previous results.

Chloroplast genomes of land plants and green algae: These 7 genomes share 85 genes; each genome is of length 87–97. The priorities of transpositions, inversions, transversions are specified as 2, 3, 4 respectively. All 21 plans (each with 6–47 events) are computed in less than an hour. (The computation of a phylogeny for these species takes almost 25 days in (Cui et al. 2006).) The phylogeny constructed by NEIGHBOR groups *Nicotiana* and *Marchantia* with *Chaetosphaeridium*, thus grouping the land plants and charophyte algae; it also groups *Chlorella* and *Chlamydomonas* with *Nephroselmis*, thus grouping the chlorophyte algae; *Mesostigma* is an outlier. These results conform with the biological evidence based on the analysis of 50 concatenated proteins (Cui et al. 2006).

Conclusion

We have introduced a new computational method, based on AI planning, to solve genome rearrangement problems with duplicate genes, involving transpositions, inversions, inverted transpositions, insertions, and deletions. No existing genome rearrangement software can handle such problems. We have shown the applicability and the effectiveness of our planning-based method on real data sets; we have observed that the results are similar to those widely accepted.

References

Bacchus, F., and Kabanza, F. 2000. Using temporal logic to express search control knowledge for planning. *Artificial Intelligence* 116(1–2):123–191.

Blanchette, M.; Kunisawa, T.; and Sankoff, D. 1996. Parametric genome rearrangement. *Gene-Combis* 172:11–17.

Blanchette, M.; Kunisawa, T.; and Sankoff, D. 1999. Gene order breakpoint evidence in animal mitochondrial phylogeny. *Journal of Molecular Evolution* 49:193–203.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1-2):165–204.

Cosner, M.; Jansen, R.; Moret, B.; Raubeson, L.; Wang, L.; Warnow, T.; and Wyman, S. 2000. An empirical comparison of phylogenetic methods on chloroplast gene order data in *Campanulaceae*. In *Comparative Genomics*. Kluwer. 99–122.

Cosner, M.; Raubeson, L.; and Jansen, R. 2004. Chloroplast DNA rearrangements in *Campanulaceae*: phylogenetic utility of highly rearranged genomes. *BMC Evolutionary Biology* 4(27).

Cui, L.; Leebens-Mack, J.; Wang, L.; Tang, J.; Rymarquis, L.; Stern, D.; and dePamphilis, C. 2006. Adaptive evolution of chloroplast genome structure inferred using a parametric bootstrap approach. *BMC Evol. Bio.* 6:13.

Erdem, E., and Tillier, E. 2005. Genome rearrangement and planning. In *Proc. of AAAI*, 1139–1144.

Felsenstein, J. 2009. PHYLIP (phylogeny inference package) version 3.6. Distributed by the author.

Gabaldon, A. 2003. Compiling control knowledge into preconditions for planning in the situation calculus. In *Proc. of IJCAI*, 1061–1066.

Moret, B.; Wyman, S.; Bader, D.; Warnow, T.; and Yan, M. 2001. A new implementation and detailed study of breakpoint analysis. In *Proc. of PSB*, 583–594.

Nielsen, C. 2001. *Animal Evolution: Interrelationships of the Living Phyla*. Oxford University Press.

Pednault, E. 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. of KR*, 324–332.

Rintanen, J. 2000. Incorporation of temporal logic control into plan operators. In *Proc. of ECAI*, 526–530.

Sankoff, D., and Blanchette, M. 1998. Multiple genome rearrangement and breakpoint phylogeny. *Journal of Computational Biology* 5:555–570.

Tesler, G. 2002. GRIMM: genome rearrangements web server. *Bioinformatics* 18(3):492–493.