5. Double-Cut-and-Join with Insertions and Deletions

Literature:

Braga, M. D. V., Willing, E., & Stoye, J. (2011). *Double cut and join with insertions and deletions.* Journal of Computational Biology, Vol. 18(9), pp. 1167–1184.

An *indel* operation adds or removes one or several consecutive genes in a genome and its unit cost is 1.

Problem 4. *Given two (circular) genomes A and B with possibly unequal gene content (but without duplications), find the* minimum *overall number of DCJ and indel operations that sort A into B, denoted $d_{DCJ}^{id}(A, B)$.*

Definition 19. *Given two genomes A and B over the sets of markers $\mathcal{G}_A$ and $\mathcal{G}_B$, respectively, let $\mathcal{G} := \mathcal{G}_A \cap \mathcal{G}_B$ be the set of markers common to both genomes. We denote by $\mathcal{A} = \mathcal{G}_A \setminus \mathcal{G}$ the markers that occur uniquely in A and the set of unique markers of B by $\mathcal{B} = \mathcal{G}_B \setminus \mathcal{G}$.*

Note that a *deletion* can only delete genes of $\mathcal{A}$. Likewise, an *insertion* can only insert markers of B.

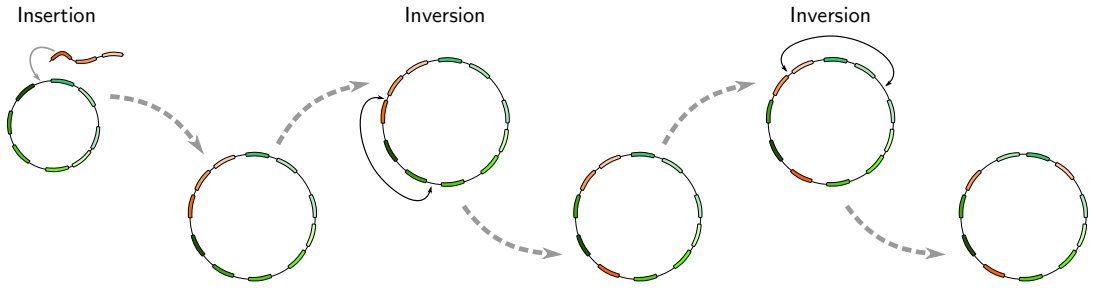


Figure 5.1.: Exemplary scenario of a segmental HGT event with that is followed by genome rearrangements. Three genes are introduced through HGT (left), and are diffused by two successive inversions (progressing towards the right).

An obvious upper bound can be constructed by considering that each unique (uncommon) gene is inserted/deleted separately from $\mathcal{A}$ and $\mathcal{B}$:

$$d^{id}_{\mathrm{DCJ}}(A,B) \leq d_{\mathrm{DCJ}}(A,B) + |\mathcal{A}| + |\mathcal{B}| \tag{5.1}$$

However, this bound is rather weak, as exemplified in Figure 5.1 when comparing the leftmost and rightmost circular chromosome. To sort the rightmost genome back to its original, leftmost form requires three DCJ operations and a single deletion of the initially inserted genes. The given upper bound proposes three independent deletions, in addition to the three DCJ operations.

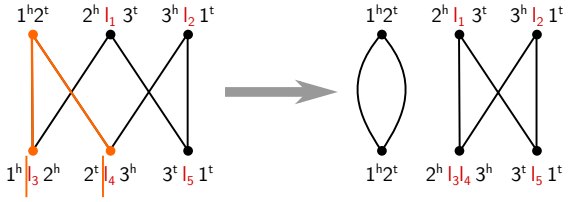
How can we determine the minimum number of DCJ+indel operations?

- For counting the DCJ operations, we have the adjacency graph.
- But unique markers have no match!
- Solution: Label points (breakpoints + adjacencies) of common genes with interleaving unique genes.

Definition 20. *A* label *is a set of unique genes between two common extremities. A $\mathcal{G}$-adjacency is a pair of common extremities and their label.*

The components in the adjacency graph are denoted as ε-, $\mathcal{A}$-, $\mathcal{B}$-, or $\mathcal{AB}$-cycles, according to whether they are entirely unlabeled, contain only labels from $\mathcal{A}$, or $\mathcal{B}$, or both.

Let us now consider for each individual cycle c the number of operations needed to sort it. Clearly, if c is a ε-cycle, then its DCJ distance is $d^{id}_{\mathrm{DCJ}}(c) = d_{\mathrm{DCJ}}(c)$. For all other cycles, we extract all *clean*, i.e. unlabelled, simple cycles. To this end, consider adjacency xy. If xy is a clean $\mathcal{G}$-adjacency in a cycle c of $AG(A,B)$, then it is always possible to extract a clean (simple) cycle from c with an optimal DCJ operation, as illustrated:



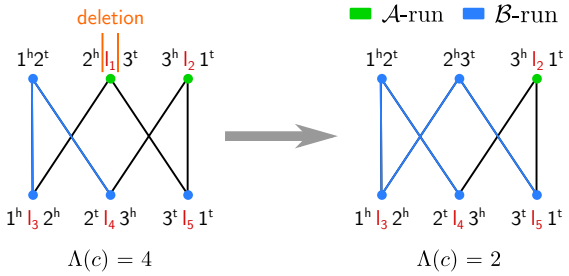
This operation, if applied on an $\mathcal{A}$- or $\mathcal{B}$-cycle, can accumulate all labels in a single adjacency. In other words, sorting an $\mathcal{A}$-/$\mathcal{B}$-cycle c takes $d_{\mathrm{DCJ}}(c)$ DCJ operations plus a single insertion/deletion.

An $\mathcal{AB}$-cycle is more challenging to sort as it requires a decomposition into *runs*: An $\mathcal{A}$-*run* in cycle c denotes a maximal path whose vertices are either unlabeled or labeled with genes of $\mathcal{A}$. $\mathcal{B}$-runs are analogously defined. Since the concept of runs can be applied not only to $\mathcal{AB}$-cycles, but any cycles, we can use this concept to obtain a new, much tighter bound on the DCJ+indel distance. To this end, let $\Lambda(c)$ denote the number

of runs of a cycle c, then the DCJ+indel distance of two genomes A and B is bounded by

$$d_{\mathrm{DCJ}}^{id}(A,B) \leq d_{\mathrm{DCJ}}(A,B) + \sum_{c \text{ cycle of } AG(A,B)} \Lambda(c)\,. \tag{5.2}$$

The true DCJ+indel distance can be lower that. Observe that an indel operation in an $\mathcal{AB}$-cycle can reduce the number of runs by 2:



Thus, the minimum number of indel operations to for a *labelled* cycle c is

$$\lambda(c) = \left\lceil \frac{\Lambda(c)+1}{2} \right\rceil , \tag{5.3}$$

and its DCJ+indel distance is $d_{\mathrm{DCJ}}^{id}(c) = d_{\mathrm{DCJ}}(c) + \lambda(c)$.

This leads to the following algorithm for sorting genome A into B (both consisting of circular chromosomes only) under the DCJ+indel model:

1. Decompose cycles into $\mathcal{A}$- and $\mathcal{B}$-runs.
2. Extract all simple ε-cycles.
3. Delete all $\mathcal{A}$ labels from genome A.
4. Extract newly formed simple ε-cycles.
5. Insert all $\mathcal{B}$ labels into genome A.

Note that this approach does not consider alternative (co-optimal) sorting scenarios, which could also involve cycle merges.