A Framework for Orthology Assignment from Gene Rearrangement Data

Krister M. Swenson, Nicholas D. Pattengale, and B.M.E. Moret

Department of Computer Science
University of New Mexico
Albuquerque, NM 87131, USA
{kswenson,nickp,moret}@cs.unm.edu

Abstract. Gene rearrangements have successfully been used in phylogenetic reconstruction and comparative genomics, but usually under the assumption that all genomes have the same gene content and that no gene is duplicated. While these assumptions allow one to work with organellar genomes, they are too restrictive when comparing nuclear genomes. The main challenge is how to deal with gene families, specifically, how to identify orthologs. While searching for orthologies is a common task in computational biology, it is usually done using sequence data. We approach that problem using gene rearrangement data, provide an optimization framework in which to phrase the problem, and present some preliminary theoretical results.

1 Introduction

Gene rearrangements have successfully been used in phylogenetic reconstruction and comparative genomics (see the survey of [13] and the monograph of [15]), but usually under the assumption that all genomes have the same gene content and that no gene is duplicated. While these assumptions allow one to work with organellar genomes [2–7, 11, 18], they are too restrictive when comparing nuclear genomes [8], where the main challenge is how to deal with gene families, specifically, how to identify orthologs. While searching for orthologies is a common task in computational biology, it is usually done using sequence data. We approach that problem using gene rearrangement data. Sankoff [14] first addressed this problem with his introduction of *exemplars*, in which he suggested identifying a single gene within each family (the exemplar) on the basis of a parsimonious criterion (using the fewest rearrangements) and discarding all others. Our group provided an alternate approach in which a correspondence is established between gene families on the basis of conserved segments [12, 17]; our results suggested that considering all members of a gene family yields better results than keeping only exemplars, but were limited in that the assignment of orthologs did not take into account any rearrangement structure beyond conserved segments. In this paper, we remedy this problem by providing an optimization framework derived from the breakpoint graph (the basic structure behind the last decade of work in gene rearrangements [10]) in which to phrase the problem; we give preliminary theoretical results in support of our framework. We are not suggesting that orthology assignment based on rearrangement data should replace that based on sequence data, but that the two complement each other—indeed, that we should aim for methods that will eventually take both types of data into account in the same analysis.

2 Preliminaries

The problem we consider can be phrased as follows: given a set of genes S and two genomes, G_1 and G_2, where each genome is represented as a (linear or circular) sequence of elements of S (an element may occur zero, one, or many times within the sequence), each with an associated sign (which basically denotes which strand the gene lies on), find the shortest *edit sequence*, that is, the shortest sequence of evolutionary events that transforms one genome into the other. Permitted evolutionary events are *inversions*, which take a subsequence of genes and reverse it in place (in both order and signs), deletions (each of which removes a consecutive subsequence of genes), and insertions (including duplications). A parsimony constraint is also imposed on any editing scenario: if G_1 has a family of k_1 genes and G_2 a corresponding family of k_2 genes, for $k_1 \geq k_2$, then none of the k_2 genes in G_2's family may be deleted in the edit sequence—instead, we must identify within G_1's family of k_1 genes a distinct ortholog for each of the k_2 genes in G_2's family. (In absence of this constraint, of course, the most parsimonious edit sequence is almost always that which deletes the entire genome G_1 as a single operation, then insert the entire genome G_2, an absurd scenario.) Once that identification has been made, the algorithms of El-Mabrouk [9] and of our group [8, 12, 17, 18] can complete the work of finding one ore more parsimonious edit sequences.

In this paper, we assume that genome G_2 contains no duplicate genes, although we illustrate how our framework describes the more general case. Moreover, we assume that gene families present in one genome but not the other have been removed—these families do not affect orthology assignment, which is simply a mapping between the elements of gene families present in both genomes. Finally, we assume that the remaining genes have been indexed from 1 to n so as to turn G_2 into the identity permutation $12\ldots n$; as is customary, we will prepend a marker gene 0 and append another marker gene $n+1$ to both genomes.

The edit distance can be affected in two ways by the orthology assignment. A good choice of orthologies can reduce the required number of deletions (or duplications)—this is the main focus of the cover-based method [12, 17]. It can also reduce the number of required inversions by grouping them properly: this is the focus of this paper. We rely on the fact that every gene in a one-member family of G_1 must be assigned to its corresponding gene in G_2 and that these singleton genes must all be sorted through inversions: because we know how to sort by inversions [1, 10], the presence of singleton genes creates a structural contex in which to study orthology assignment.

3 Background and Definitions

3.1 The Breakpoint Graph

The basic structure describing a pair of genomes with no duplicates and equal gene content is the *breakpoint graph* (really a multigraph)—for a careful and readable description of its construction, see [16]. In our case, however, gene families in G_1 need not be singletons, so we need to extend the construction. Let $B(G_1) = (V, E)$ denote the breakpoint graph for G_1 and G_2 (because of our conventions, G_2 is known once G_1 is). As in the regular breakpoint graph, each singleton gene g in G_1 becomes a pair of vertices, g^- and g^+ (the "negative" and "positive" terminals), joined by an edge; we leave out the gene families with multiple members, since only the singletons have a well-defined structure, but we now need to accommodate gaps left in the sequence where duplicate genes exist in G_1. We add a *desire* edge (in the charming terminology of [16]—also known elsewhere as a gray edge) (a_i^-, b_j^+), for each member i and j of gene families a and b, respectively, whenever a and b differ by one in the indexing (i.e., are neighbors in G_2). We add a *reality* edge (a^p, b^q) if a is the element to the left of b in G_1 and either $p = q$ if a and b have different parities (in G_1 naturally) or $p \neq q$ if a and b have the same parity. Figure 1 illustrates the construction. Note that the figure shows a circular ordering, in which the added sentinels are considered adjacent to each other.

a) $G_1 =$ 4 −3 2 3 1 6 9 8 −10 −7 9

b)

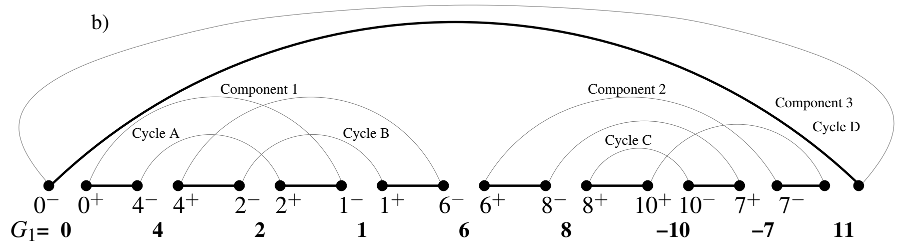

Fig. 1. (a) The original genome G_1. (b) Its associated graph $B(G_1)$ after removing gene families with duplicates (3 and 9). Desire edges are shown in gray, reality edges in black.

The inversion distance equals the number of genes minus the number of cycles in the breakpoint graph, plus some corrective factors (hurdles and a fortress) [10]. Researchers have found that hurdles are very rare in real data, so we focus on selecting an orthology assignment that maximizes the number of cycles.

3.2 The Consequences of An Assignment

We call each gene in a multigene family of G_1 a *candidate*, since it is one of the choices for an orthology assignment to the unique corresponding gene in G_2. For each candidate d, denote by $\beta^+(d)$ the positive terminal of the next smaller element in $B(G_1)$ and by $\beta^-(d)$ the negative terminal of the next larger element; we call these nodes the *bookends* of d. Choosing a candidate imposes an additional constraint on $B(G_1)$, which we now proceed to examine. The following simple lemma underlies many of our results.

Lemma 1. *When a candidate d is chosen, the only edges affected are the reality edge that spans it and the edge between its bookends.*

Proof. As shown in Figure 2, when d is added to the breakpoint graph, the reality edge that spans it gets split, creating two new endpoints d_{β^+} and d_{β^-}. The desire edge that links the bookends of d also gets split, to meet each of d_{β^+} and d_{β^-}.

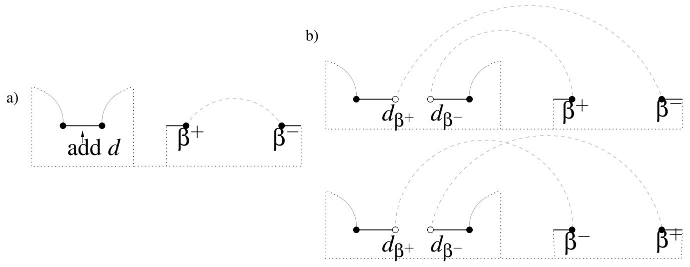

Fig. 2. (a) A subgraph of $B(G_1)$ before adding d. (b) The two choices after adding d. Dotted lines indicate a path (and therefore a cycle).

3.3 The Cycle Splitting Problem

As observed above, we can formulate the orthology assignment problem as an optimization problem within the context of the breakpoint graph $B(G_1)$: choose an assignment of orthologs (one from each multigene family in G_1) such that the number of cycles in the augmented breakpoint graph ($B(G_1)$ to which the chosen candidates have been added) has the largest possible number of cycles.

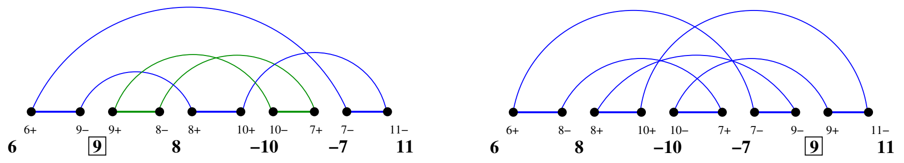

Fig. 3. The breakpoint graphs for the two candidates for gene 9.

Consider component 2 from Figure 1, namely $(6,9,8,-10,-7,9,11)$. There are two occurrences of gene 9 and we must choose which one to call orthologous with gene 9 in G_2. Figure 3 shows the two breakpoint graphs. Note that the graph on left, where the candidate lies between 6 and 8, has one more cycle than the graph on the right, where the candidate lies between 7 and 11; thus the first candidate is a better choice. The choice of candidate is advantageously viewed on breakpoint graph inscribed in a circle as shown in Figure 4. Now overlay the two choices into a single graph, as shown in Figure 4(b). Two curved lines meet on the perimeter between 10^- and 8^+, denoting the two choices. The solid line indicates that choosing the candidate between 6 and 8 gives rise to desire edges that do not cross in the inscribed representation. The dashed line indicates that the other candidate gives rise to crossing desire edges. Each line meets the perimeter at one end between the two terminals of the candidate and at the other end between its bookends. Figure 5(a) illustrates a more general instance with three multigene families, while Figure 5(b) shows how this representation can be used for the general many-to-many case.

Note that the solid and dashed lines we have introduced really represent orthology assignments, or *operations*; we will call an operation represented by a solid line a *straight* operation (because it does not introduce crossings) and one represented by a dashed line a *cross* operation. The collection of all lines

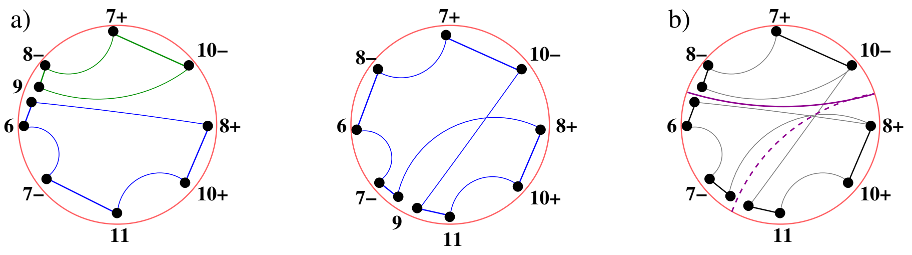

Fig. 4. (a) The graphs of Figure 3 inscribed in a circle. (b) The result of overlaying the two graphs from part (a).

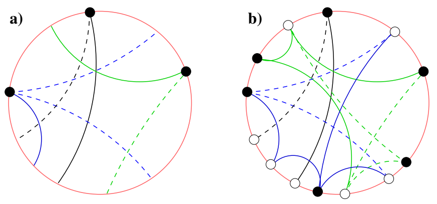

Fig. 5. (a) An instance of the many-to-one cycle splitting problem. (b) An instance of the many-to-many cycle splitting problem.

that share an endpoint represents all members of the gene family in G_1, so we also call it a family and call its common endpoint (between the bookends) the *family home*. We can now state the constraints for the optimization problem: (i) each family home is a distinct point on the circle; (ii) the family home is not the endpoint of any operation not in that family; and (iii) the other endpoint (on the circle) of each operation is unique to that operation.

The many-to-many variation gives rise to multiple homes per family. Each home in the same family must connect to all of the same endpoints. The problem thus becomes picking as many operations as there are homes per family such that the cycle count is maximized. The only additional complication is that applying an operation removes that operation from consideration in all other homes for its family. See Figure 5(b) for an example instance as well as the effect of applying an operation.

Straight and cross operations display a form of duality, one that allows us to restrict our attention to just straight operations.

Theorem 1. *Applying a cross operation c converts all operations that intersect c (call the set of such operations I) to their complement—crosses are replaced by straights and straights by crosses. Furthermore, for any two operations in I, if they intersected before applying c, then they no longer do after applying c, and vice versa.*

Figure 6 sketches the main elements of the omitted proof.

4 Theoretical Results

4.1 Buried Operations

An operation makes no contribution to the cycle count of a a complete assignment whenever the two new desire edges it creates lie on the same cycle. Because of their appearance in our inscribed graph representation, we call such operations *buried*—see Figure 7. Since the two desire edges occur on the same cycle, they "bury" the chosen operation in the cycle.

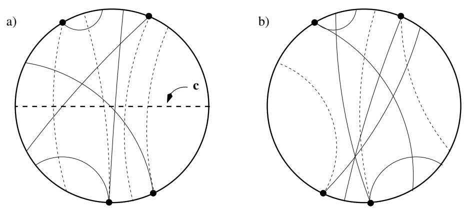

Fig. 6. The visual argument for Theorem 1.

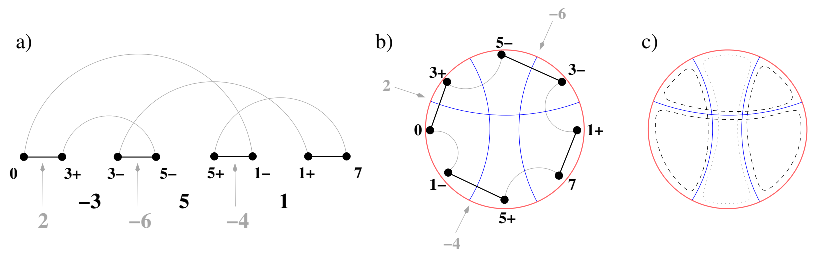

Fig. 7. (a) A new configuration; (b) a buried edge for that configuration; and (c) the visual appearance of the buried edge, inside two segments of the same cycle.

Theorem 2. *If an orthology assignment creates a total of k buried edges, then the number of cycles is bounded by* $n-b+1$.

Proof. The number of cycles cannot exceed $n+1$, since each operation can give rise to at most one new cycle. Consider the effect on the breakpoint graph of choosing a buried operation. A single desire edge d is replaced with two desire edges d'_1 and d'_2, and a single reality edge r is replaced with two reality edges r'_1 and r'_2. Without loss of generality assume that d'_1 connects to r'_1 and d'_2 connects to r'_2. d'_1 and d'_2 each inherit one of the original endpoints of d. Similarly, r'_1 and r'_2 each inherit one of the original endpoints of r. By assumption d'_1 and d'_2 lie on the same cycle, so that so do all of the original endpoints of d and r. Thus all of the newly created edges must lie on a cycle that already existed.

4.2 Chains and Stars

We have discovered two operation patterns that, while they need not contain buried operations, nevertherless impose sharp bounds on the number of cycles. A *k-chain* is an assignment in which k operations form a chain, that is, each chosen operation overlaps two of the other k, its predecessor and successor around the circle. Figure 8(a,b) illustrates a k-chain.

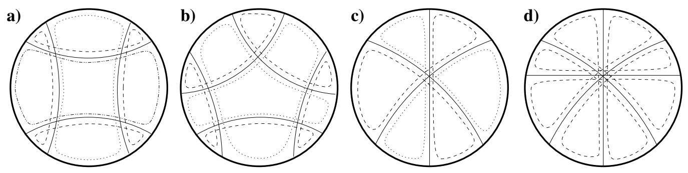

Fig. 8. (a) A 4-chain; (b) a 5-chain; (c) a 3-star; (d) a 4-star.

Proposition 1.

- *A k-chain has no buried operations.*
- *In a k-chain with k odd, the cycle count is* 2.
- *In a k-chain with k even, the cycle count is* 3.

A *k-star* is an assignment in which *k* operations form a clique (each overlaps every other). Figure 8(c,d) illustrates a *k*-star.

Proposition 2.

- *In a k-star with k even, every operation is buried and the cycle count is* 1.
- *In a k-star with k odd, no operation is buried and the cycle count is* 2.

We conjecture that these two patterns, along with buried operations, describe all operations that reduce the upper bounds on the number of cycles. Figure 9 shows examples of combining *k*-stars and *k*-chains.

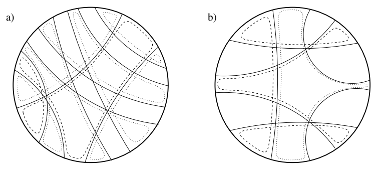

Fig. 9. (a) A 3-star and two 4-chains. (b) Four 3-stars.

4.3 Reduced Forms

Clearly, a successive assignment procedure could reach a state in which no operation remains that could split a cycle. We call such a state a *reduced form* of the instance. In a reduced form, an instance is composed of multiple cycles linked by the operations from the remaining families. This structure lends itself naturally to a graph representation; an analysis of this graph reveals conditions under which optimality can be verified.

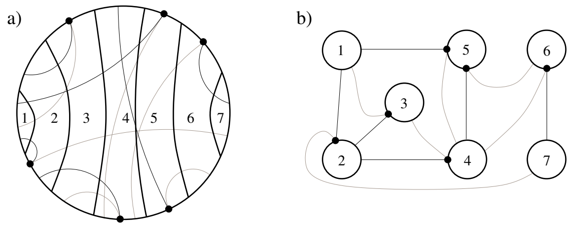

Fig. 10. (a) An instance of the cycle splitting problem. The operations without a family home (bold) are those that we apply to obtain a reduced form. (b) The resulting reduced instance. Black edges are those chosen to produce an optimal solution to the reduced instance.

Theorem 3. *After applying a maximal nonoverlapping set of operations M, remaining operations can only (by themselves) join two cycles.*

Proof. The application of a set of k nonoverlapping operations always yields k new cycles, each defined either by two contiguous operations or by a single operation and the original circle. Since M is maximal, every remaining operation from every family overlaps an element of M. Application of any $m \in M$, therefore, must span two of the new cycles, joining them into one.

Figure 10(b) shows the reduced instance induced by applying each of the operations shown in bold in Figure 10(a). We are left with a reduced form that can be viewed as a graph on the cycles created so far and in which each adjacency list is strictly ordered (because exactly one operation from a family will remain).

We can now take advantage of graph properties such as planarity, circuits, and connected components in our analysis. Because of the ordered nature of the edges on the vertices planarity is somewhat specialized in our case. Nonplanar edges can occur in simpler situations than in general graphs, as shown in Figure 11(b). Circuits play a vital role on these graphs. In particular, we are interested in circuits that are subcircuits of no other, i.e., *minimal* circuits. Provided

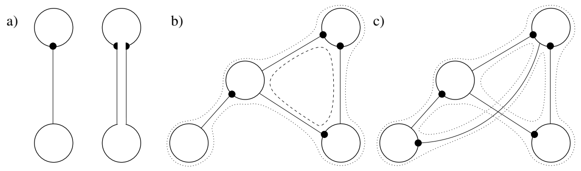

Fig. 11. (a) The effect of applying an operation on a reduced form. (b) A reduced form. The dotted and dashed lines trace the cycle created by applying the operations shown. (c) Adding a nonplanar edge to the reduced form from (b) joins the cycles.

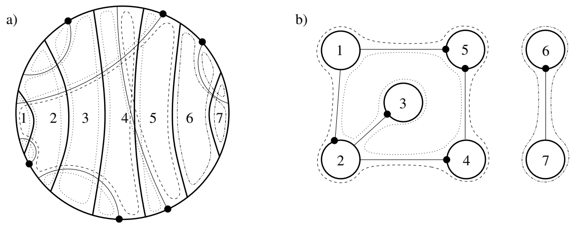

Fig. 12. An optimal solution to the reduced instance in Figure 10 embedded as (a) the original instance and (b) its reduced form.

there is no nonplanar edge in a reduced instance, the properties of a solution directly dictate the number of cycles produced. As shown in Figure 11, each connected component produces a cycle around its outer hull. Each minimal circuit yields another cycle to its inside. Figure 11(c) shows how nonplanar edges can join these two cycles.

Theorem 4. *The number of cycles produced by a solution S to a planar reduced instance with m minimal circuits and c connected components is $R(S) = m + c$.*

Proof. This certainly holds for a reduced instance with no operations. Assume $R(S) = m + c$ for a solution where $|S| = k$. We look at the effect of adding another edge.

1. If that edge links two previously disconnected components, then the cycles around the hulls of these components will get merged, removing a cycle and a connected component.
2. If that edge links two connected components, then a minimal circuit will be created. Since the edge added is planar, we know that the same cycle runs past both endpoints of the operations and thus the operation will split it.

It remains to relate results on reduced forms back to the original inscribed breakpoint graph formulation, a process illustrated in Figure 12.

5 Conclusion

We have described a graph-theoretical framework in which to represent and reason about orthology assignments and their effect on the number of cycles present in the resulting breakpoint graph. We have given some foundational results about this framework, including several that point us directly to to algorithmic strategies for optimizing this assignment. We believe that this framework will lead to a characterization of the orthology assignment problem as well as to the development of practical algorithm solutions.

References

1. D.A. Bader, B.M.E. Moret, and M. Yan. A fast linear-time algorithm for inversion distance with an experimental comparison. *J. Comput. Biol.*, 8(5):483–491, 2001.
2. M. Blanchette, T. Kunisawa, and D. Sankoff. Gene order breakpoint evidence in animal mitochondrial phylogeny. *J. Mol. Evol.*, 49:193–203, 1999.
3. J.L. Boore. Phylogenies derived from rearrangements of the mitochondrial genome. In N. Saitou, editor, *Proc. Int'l Inst. for Advanced Studies Symp. on Biodiversity*, pages 9–20, Kyoto, Japan, 1999.
4. J.L. Boore and W.M. Brown. Big trees from little genomes: Mitochondrial gene order as a phylogenetic tool. *Curr. Opinion Genet. Dev.*, 8(6):668–674, 1998.
5. J.L. Boore, T. Collins, D. Stanton, L. Daehler, and W.M. Brown. Deducing the pattern of arthropod phylogeny from mitochondrial DNA rearrangements. *Nature*, 376:163–165, 1995.
6. M.E. Cosner, R.K. Jansen, B.M.E. Moret, L.A. Raubeson, L. Wang, T. Warnow, and S.K. Wyman. An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae. In D. Sankoff and J.H. Nadeau, editors, *Comparative Genomics*, pages 99–122. Kluwer Academic Publishers, 2000.
7. S. R. Downie and J. D. Palmer. Use of chloroplast DNA rearrangements in reconstructing plant phylogeny. In D.E. Soltis, P.S. Soltis, and J.J. Doyle, editors, *Molecular Systematics of Plants*, pages 14–35. Chapman and Hall, New York, 1992.
8. J. Earnest-DeYoung, E. Lerat, and B.M.E. Moret. Reversing gene erosion: reconstructing ancestral bacterial genomes from gene-content and gene-order data. In *Proc. 4th Int'l Workshop Algs. in Bioinformatics (WABI'04)*, volume 3240 of *Lecture Notes in Computer Science*, pages 1–13. Springer Verlag, 2004.
9. N. El-Mabrouk. Genome rearrangement by reversals and insertions/deletions of contiguous segments. In *Proc. 11th Ann. Symp. Combin. Pattern Matching (CPM'00)*, volume 1848 of *Lecture Notes in Computer Science*, pages 222–234. Springer Verlag, 2000.
10. S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proc. 27th Ann. ACM Symp. Theory of Comput. (STOC'95)*, pages 178–189. ACM Press, New York, 1995.
11. B. Larget, D.L. Simon, and J.B. Kadane. Bayesian phylogenetic inference from animal mitochondrial genome arrangements. *J. Royal Stat. Soc. B*, 64(4):681–694, 2002.
12. M. Marron, K.M. Swenson, and B.M.E. Moret. Genomic distances under deletions and insertions. *Theor. Computer Science*, 325(3):347–360, 2004.
13. B.M.E. Moret, J. Tang, and T. Warnow. Reconstructing phylogenies from gene-content and gene-order data. In O. Gascuel, editor, *Mathematics of Evolution and Phylogeny*, pages 321–352. Oxford University Press, 2005.
14. D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):990–917, 1999.
15. D. Sankoff and J. Nadeau, editors. *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*. Kluwer Academic Pubs., Dordrecht, Netherlands, 2000.
16. J.C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishers, Boston, MA, 1997.
17. K.M. Swenson, M. Marron, J.V. Earnest-DeYoung, and B.M.E. Moret. Approximating the true evolutionary distance between two genomes. In *Proc. 7th SIAM Workshop on Algorithm Engineering & Experiments (ALENEX'05)*. SIAM Press, Philadelphia, 2005.
18. J. Tang, B.M.E. Moret, L. Cui, and C.W. dePamphilis. Phylogenetic reconstruction from arbitrary gene-order data. In *Proc. 4th IEEE Symp. on Bioinformatics and Bioengineering BIBE'04*, pages 592–599. IEEE Press, Piscataway, NJ, 2004.