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 been used successfully 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 for nuclear genomes. The main challenge in handling more realistic data 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. Sankoff first addressed the problem in 1999, introducing the notion of exemplar, but his approach uses an NP-hard optimization step to discard all but one member (the exemplar) of each gene family, losing much valuable information in the process. We approach the problem using all available data in the gene orders and gene families, provide an optimization framework in which to phrase the problem, and present some preliminary theoretical results.

1 Introduction

Gene rearrangements have been used in phylogenetic reconstruction and comparative genomics (see, e.g., [17, 23]), but usually under the assumption that all genomes have the same gene content and that no gene is duplicated. These assumptions allow one to work with organellar genomes [2–5, 9, 10, 15, 21, 26], but are too restrictive for nuclear genomes [11], 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 [19] first addressed this problem, proposing to identify within each gene family an *exemplar* (a single gene, presumably the "original" one within that family) and to discard all other homologs, thereby reducing the problem to one in which no gene is duplicated. He further proposed that, for a pair of genomes, the exemplars should be selected so as to minimize the distance (measured in terms of breakpoints or inversions) between the two reduced genomes. One problem with this approach is that identifying the exemplars is itself NP-hard, even when one genome contains no duplicate genes [6]; another is that, by discarding all homologs, much valuable information is lost. (The different numbers and arrangements of homologs need to be explained with a suitable sequence of duplications, losses, and inversions, none of which appears in the exemplar framework.) Nguyen et al. [18] proposed a divide-and-conquer approach to compute an exemplar-based distance between two genomes in reasonable time.

Sankoff *et al.* [22] also proposed a simple heuristic based on breakpoints [20] that adds new genes incrementally at random; that heuristic performed well on a small collection of mitochondrial genomes with widely divergent contents. However, that method cannot handle duplications, only deletions and nonduplicating insertions; it is thus well suited to organellar genomes, but not to nuclear genomes, where large gene families are common. El-Mabrouk later gave an exact solution for that problem (but with respect to inversion distances), as well as a bounded-ratio approximation when both deletions and non-duplicating insertions are allowed [12]. She also developed an approach, based on her earlier work with doubled genomes, that uses both inversions and duplications [13]. Our group provided an alternate approach in which a correspondence is established between gene families on the basis of conserved segments [16, 25] before completing the sequence using El-Mabrouk's algorithm; 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. Chen *et al.* [8] gave a first attempt at using rearrangements and keeping more than just exemplars.

In this paper, we extend these approaches by providing an optimization framework derived from the breakpoint graph (the structure behind the last decade of work in gene rearrangements [14]) in which to phrase the problem; we give preliminary theoretical results in support of our framework.

2 Preliminaries

We are given a set of gene families S (the set of "names" of the gene families) and two genomes, G_1 and G_2. 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 denotes which strand the gene lies on). In this formulation, each genome consists of a single chromosome; however, the unichromosomal version embodies the heart of the orthology assignment problem and, as shown by Tesler [27] in the context of equal gene contents, a multichromosomal version does not introduce insurmountable problems. The problem is to find the shortest *edit sequence*, that is, the shortest sequence of evolutionary events that transforms one genome into the other. Permitted evolutionary events in this setting are *inversions*, which take a subsequence of genes and reverse it in place (in both order and signs), *deletions*, and *insertions* (including *duplications*). These events all operate on consecutive subsequences of genes: that is, we assume that the cost of deleting, inserting, or making one duplicate of, one gene is the same as that of deleting or inserting (including duplicating insertions) a contiguous segment of k genes, for any $k \geq 1$.

In absence of other constraints, the edit distance between any two genomes is then bounded by 2: simply delete the entire genome in one operation of unit cost, then insert the entire new genome in another operation of unit cost. Since this scenario is patently absurd in biological terms, we impose a simple parsimony constraint on any editing scenario: if G_1 has a family of k_1 genes and G_2 a homologous family of k_2 genes, with $k_1 \geq k_2$, then none of the k_2 genes in G_2's family may be inserted in the edit sequence from G_1 to G_2: 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. The $k_1 - k_2$ unmatched homologs in G_1's family will then be deleted in the edit sequence. Once that orthology identification has been made, the algorithms of El-Mabrouk [12] and of our group [11, 26] can complete the work of finding one or more parsimonious edit sequences.

A good choice of orthologies can reduce the required number of deletions and insertions (or duplications) by inserting contiguous segments of many genes rather that one gene at a time—although care must be taken not to do so at the expense of proper placement within the ordering, lest many additional inversions be required to move individual genes to their final destination [16]. It can also reduce the number of required inversions by grouping genes properly: this is the focus of the cover-based methods [8, 16, 25].

We shall rely on the fact that every gene that appears as a singleton in both genomes has a direct assignment and that these singleton genes must all be sorted through inversions: because we know how to sort by inversions [1, 14], the presence of singleton genes creates a structural context in which to study orthology assignment [19].

We assume, without loss of generality, that gene families present in one genome but not the other have been removed—these families do not affect orthology assignment and the insertion of the unique genes can easily be handled by El Mabrouk's algorithm. We describe the framework for the general case, but, for the sake of clarity in presentation, we shall frequently restrict genome G_2 to contain no duplicate genes, in which case our framework becomes a special case of the exemplar problem. Finally, when using G_2 with no duplicate genes, 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 0^+ and append another marker $n+1^-$ to both genomes.)

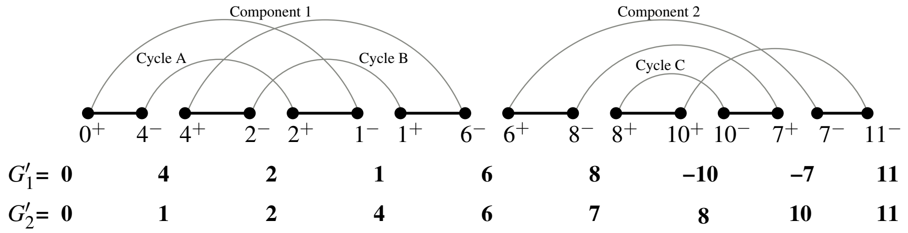
3 Background and Definitions

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 readable description of its construction, see [24]. In our case, however, gene families need not be singletons, so we modify the construction to include *only* singleton gene families. Let $BG_{1,2}$ denote the breakpoint graph for G_1 and G_2; As in the normal breakpoint graph, each singleton gene g becomes a pair of vertices, g^- and g^+ (the “negative” and “positive” terminals); however, we leave out the gene families with multiple members, since only the singletons have a readily usable structure. We need to accommodate gaps left in the sequence where duplicate genes exist in G_1. Call the versions of G_1 and G_2 without multigene families G_1' and G_2' respectively. We add an edge (a *desire* edge, in the charming terminology of [24], but also known elsewhere as a gray edge) (a^-, b^+) for each singleton a and b, whenever a occurs immediately to the left of b in G_2'. We add a *reality* edge (also known elsewhere as a black edge) (a^p, b^q) if a is the element to the left of b in G_1' and we have 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. Thus desire edges trace the (re-)ordering of G_1 that we need to achieve to match G_2, while reality edges trace the given ordering of G_1. Figure 1 illustrates the construction.

G_1 = 4 -3 2 3 1 6 9 3 8 -10 -7 9

(a) the genome G_1



(b) the breakpoint graph $BG_{1,2}$

Fig. 1. A genome G_1 and its associated breakpoint graph $BG_{1,2}$ (with respect to the identity permutation G_2) after removing gene families with duplicates (3 and 9); desire edges are shown in gray, reality edges in black.

Hannenhalli and Pevzner proved that 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). Researchers have found that hurdles are very rare in real data (a finding confirmed in a theorem under some restrictive assumptions [7]), so we focus on selecting an orthology assignment that maximizes the number of cycles.

3.2 The Consequences of An Assignment

Our job of assigning orthologs may be compared to that of reshelving books in a library with unlabelled shelves. Each book has a proper location on a shelf and multiple copies of a book must be shelved together. A librarian can proceed by first removing misshelved books and then identifying the appropriate location of each book based on the context of the books that remain in their correct spot.

In our problem each multigene family has been removed from the ordering, leaving a structure of cycles defined by singleton genes. We call each gene in a multigene family of G_2 a *candidate*, since it is one of the choices for an orthology assignment to a corresponding gene in G_2. Like each book in the library, each candidate has a location between two remaining elements in G_2'; each family, like each group of book copies, contains candidates that all share the same location between elements of G_2'. For each candidate d, denote by $\beta^+(d)$ the positive terminal of the next smaller (in value) element in $BG_{1,2}$ and by $\beta^-(d)$ the negative terminal of the next larger element. We call these nodes the *bookends* of d and the cycle on which they reside the *shelf* of d. For instance, in Figure 1, the bookends for the family of gene 3 (a family of 3 members) are 2^+ and 4^- and therefore the shelf for the family of 3s is cycle A. Although the definition of bookends applies equally well to singletons, we are only interested in bookends for candidates: bookends are part of the breakpoint graph, but candidates are not, since multigene families do not appear in the breakpoint graph.

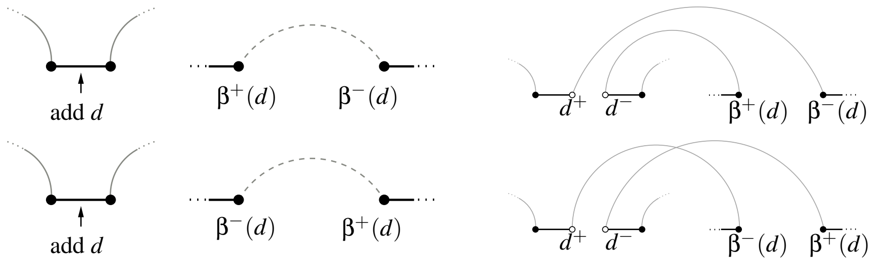
Once we have chosen a candidate, the candidate and its matching gene in G_2 effectively form a singleton gene family, so we can add the candidate to the breakpoint graph of G_1. The consequences of that choice are summarized in the following easy lemma, which underlies many of our results.

Lemma 1. *When a candidate d is chosen, exactly two edges are affected: the reality edge that spans the location where d is added and the edge between its bookends.*

Proof. Refer to Figure 2. Adding d to $BG_{1,2}$ splits the reality edge that spans the location where d is added, creating two new endpoints d^+ and d^-, as well as splitting the desire edge that links $\beta^+(d)$ and $\beta^-(d)$ to meet each of d^+ and d^-.

We say that a candidate d is added *on-cycle* if, once added, it lies on its own shelf; otherwise it is added *off-cycle*. The following is an immediate consequence of Lemma 1.

Lemma 2. *When a candidate is added off-cycle, two cycles get joined.*



(a) a subgraph of $BG_{1,2}$ before adding d; the dashed line is the desire edge that will be split

(b) the two possibilities after adding d

Fig. 2. Adding an element d to a breakpoint graph.

3.3 The Cycle Splitting Problem

We can formulate orthology assignment as an optimization problem: choose an assignment of orthologs that maximizes the number of cycles in the resulting breakpoint graph (i.e. $BG_{1,2}$, to which the chosen candidates have been added). Note that the order in which the chosen candidates are added does not affect the structure of the resulting breakpoint graph.

Consider cycle C in Figure 1. This cycle is associated with the gene segment $(6,9,8,-10,-7,9,11)$, which contains two occurrences of gene 9; thus we must choose which of these two occurrences to call the ortholog of gene 9 in G_2. Figure 3 shows the augmented breakpoint graphs resulting from each choice of candidate. The graph on the left, where we chose the candidate between 6 and 8, has one more cycle than the graph on the right, where we chose the candidate between -7 and 11, and is thus the better choice.

The choice of a candidate is advantageously viewed on a breakpoint graph inscribed in a series of circles, one for each cycle in the graph. We embed each cycle of $BG_{1,2}$ in a circle by choosing any start vertex and then following the cycle. Figure 4 shows three of the four cycles of Figure 1 inscribed in three circles. Returning to the two possible orthology assignments shown in Figure 3, we can look at the inscribed versions of these graphs, as illustrated in Figure 5(a).

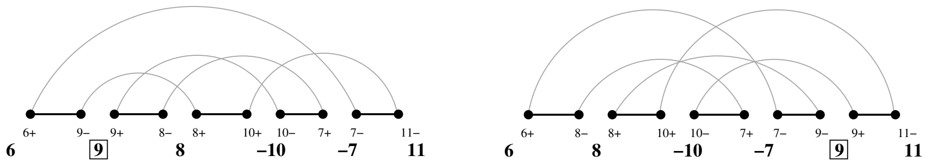


Fig. 3. The breakpoint graphs for the two candidates for gene 9 on cycle C.

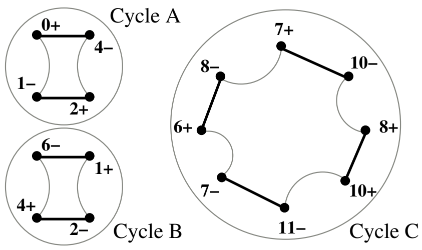
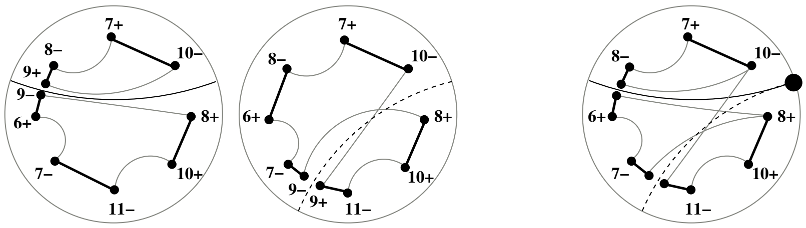


Fig. 4. The breakpoint graph of Figure 1 inscribed in three circles (cycle D is not shown).

Choosing candidates adds edges across the circle, edges that may cross each other, depending on the parity of the candidates and the locations of their bookends. The effects on the graph can be represented in just one graphical representation, as shown in Figure 5(b). In this representation, we denote the two choices by drawing two curved line segments, both originating on the perimeter between the bookends 10^- and 8^+ and each ending between the two terminals of the corresponding candidate. Choosing the candidate between 6^+ and 8^- gives rise to desire edges that do not cross in the inscribed representation; we represent such choices with solid lines. The other candidate, between 7^- and 11^-, does give rise to crossing desire edges; we represent such choices with dashed lines.



(a) the graphs of Figure 3 inscribed in circles

(b) the two choices of part (a) superimposed

Fig. 5. How the cycle splitting problem can be inscribed in a single circle.

These curved lines represent assignment *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 operations that share an endpoint represents all members of a gene family from G_1, so we also call it a *family* and call its common endpoint (between the bookends and represented by a large disk in the figures) the *family home*. We can now state the three constraints for our optimization problem:

1: Each family home is a distinct point on the circle.
2: The family home is not the endpoint of any operation not in that family.
3: The other endpoint of each operation is unique to that operation.

The objective to be maximized is the number of cycles. Figure 6 shows the operations for each of the gene families from our running example. Operations that cross cycles are off-cycle and therefore will join cycles.

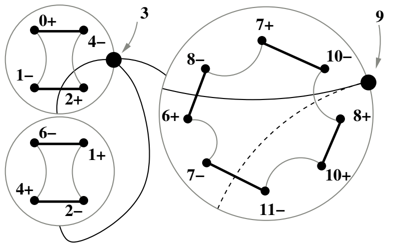


Fig. 6. The operations that represent the gene families for our running example.

Figure 7 shows a single cycle and its operations for the simplified ("many-to-one") case where G_2 has only singletons and for the general ("many-to-many") case where both G_1 and G_2 have multigene families. (The case where two multigene families have the same bookends can be handled because the relative location of the bookends does not change.) In the general case we have multiple homes per family, with one additional constraint:

4: 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 (as required by the fourth constraint).

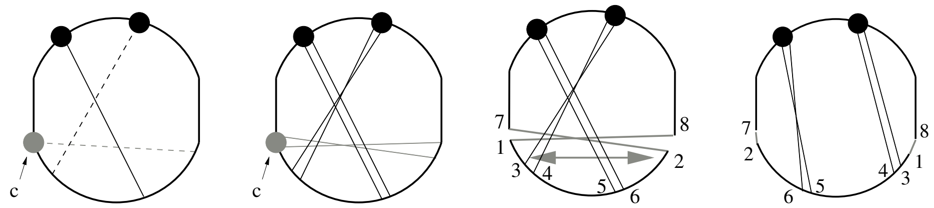
(a) a many-to-one instance *(b) a many-to-many instance*

Fig. 7. A single cycle for the simplified case (left) and the general one (right).

Straight and cross operations display a form of duality that allows us to focus on straight operations alone.

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.*

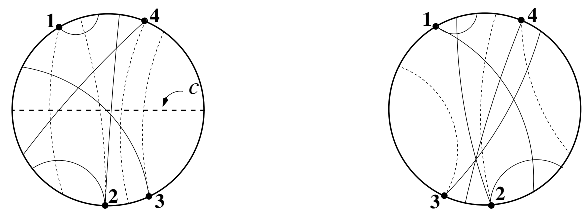
Proof. We sketch the proof graphically, using Figure 8, a typical situation where three operations, two of which are crosses and one a straight, overlap each other. The cross operation shown in parts (a) and (b) twists, but does not break the cycle, as shown in part (c). If we redraw the cycle inscribed neatly in a circle, we find we must reverse the indices on half of the cycle; Figure 8(d) shows the result after reversing indices on the bottom half of the cycle. Previously intersecting operations no longer intersect and the identities of the operations have been inverted.



(a) drawn on the circle as usual *(b) zooming in on each operation* *(c) after applying cross operation c* *(d) result redrawn on the circle*

Fig. 8. Illustration for Theorem 1. Labels for the points along the circle are numbered.

Figure 9 shows the implications of Theorem 1 in a more complicated setting.



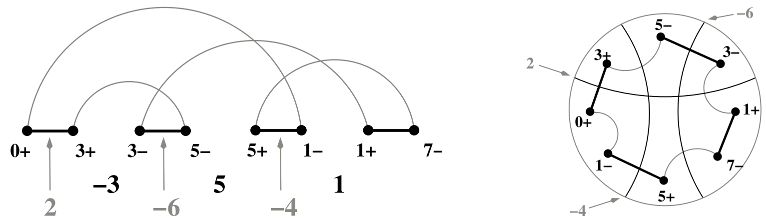
(a) before application *(b) after application*

Fig. 9. Applying cross operation c.

4 Theoretical Results

4.1 Buried Operations

An operation makes no contribution to the cycle count of a complete assignment if the two new desire edges it creates lie on the same cycle. In Figure 10, the choices of candidates for the gene families are indicated in the breakpoint graph on the left and shown as operations in the inscribed representation on the right.



(a) the breakpoint graph *(b) the inscribed version of $BG_{1,2}$*

Fig. 10. An example with $G_1 = 2, -3, 4, -6, 5, -4, -2, 6, 1$. Chosen duplicates are shown in grey.

In Figure 11, we show again the three operations depicted in Figure 10(b), but this time only the three operations and the resulting two cycles are shown. Note the operation corresponding to gene family 2 (shown as a heavy curve): the curved edge is bounded on each side by the same cycle; we say that such an operation is *buried* (for the given choice of candidates). Since the two desire edges created by this operation lie on the same cycle, the operation does not increase the number of cycles (in fact, in this particular example, it reduces the number of cycles, which stood at 3 after operations -6 and -4).

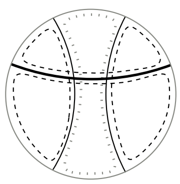


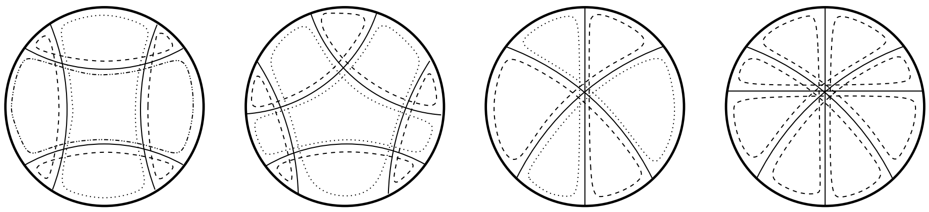
Fig. 11. The cycle and the operations; operation "2" (the heavy curve) is buried.

Theorem 2. *If an orthology assignment creates a total of b buried edges, then the number of cycles is bounded by $a - b + 1$, where a is the number of cycles present in the breakpoint graph induced by the shared singleton genes plus the total number of orthology assignments to be made.*

Proof. The number of cycles cannot exceed $a+1$, since each orthology assignment can give rise to at most one new cycle. Consider the effect on the breakpoint graph of choosing an 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. By construction, 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, the chosen edge is buried, so that d'_1 and d'_2 lie on the same cycle; therefore 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. Since this is true of any buried operation, every one of the buried operations decreases by one the maximum number of attainable cycles.

4.2 Chains and Stars

We have discovered two operation patterns that, while they need not contain buried operations, nevertheless impose sharp bounds on the number of cycles. A *k-chain* (for $k \geq 3$) 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 12(a,b) illustrates k-chains. A *k-star* (for $k \geq 1$) is an assignment in which k operations form a clique (each overlaps every other). Figure 12(c,d) illustrates k-stars.



(a) a 4-chain *(b) a 5-chain* *(c) a 3-star* *(d) a 4-star*

Fig. 12. Some examples of stars and chains.

Proposition 1. *For any integer $k \geq 1$ (but recall that k-chains are only defined for $k \geq 3$), we have:*

1. *a k-chain has no buried operations;*
2. *in a k-chain with k odd, the cycle count is* 2*;*
3. *in a k-chain with k even, the cycle count is* 3*;*
4. *in a k-star with k even, every operation is buried and the cycle count is* 1*;*
5. *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.

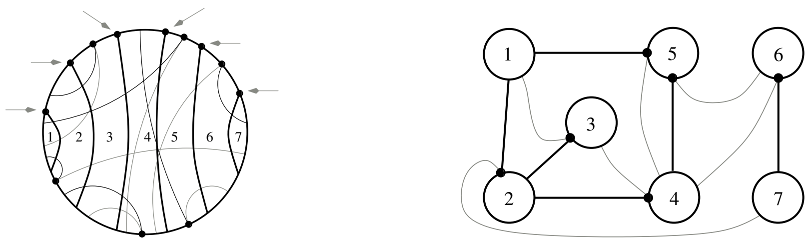
4.3 Reduced Forms

A serial 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 characterized.

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

Proof. Applying a set of k nonoverlapping operations yields k new cycles, each separated from the others by two adjacent operations or, in the case of an outermost cycle, by one operation from all others. 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 13(b) shows the reduced instance induced by applying each of the (straight) operations chosen in Figure 13(a). We are left with a reduced form that can be viewed as a graph on the cycles so far; however, because that graph is embedded in the plane, the edges incident on a vertex are strictly ordered.

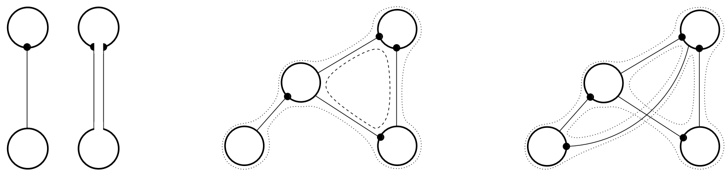


(a) operations indicated by heavy lines (and arrows) are those chosen to produce the reduced form of part (b)

(b) the resulting reduced instance; heavy edges will produce an optimal solution to the reduced instance

Fig. 13. Creating a reduced instance and solving it.

We can now take advantage of graph properties such as planarity, cycles, and connected components. Because of the ordered nature of the edges incident upon a given vertex, planarity is somewhat specialized in our case: nonplanar edges can occur in simpler situations than in general graphs, as shown in Figure 14(c). Cycles again play a vital role in these new graphs. If we restrict our attention to planar graphs, we can look at the elementary cycles (those that delimit an inside face of the planar embedding) and obtain directly the value of an



(a) the effect of applying an operation between two circles

(b) a reduced form: lines trace the cycles created by the operations

(c) adding a nonplanar operation to the reduced form from (b) joins the cycles

Fig. 14. The effect of choosing operations on a reduced form.

optimal solution. As shown in Figure 14, each connected component produces a cycle around its outer hull (one of the cycles for the outer face of the planar graph). Each elementary cycle yields another cycle to its inside. Figure 14(c) shows how nonplanar edges can join these two cycles.

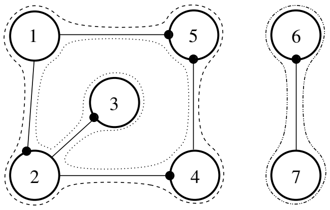
Theorem 4. *The number of cycles in a solution S to a planar reduced instance with m elementary cycles and c connected components is $R(S) = m + c$.*

Proof. This certainly holds for a reduced instance with no operations. Assume $R(S) = m + c$ for a current solution and look at the effect of adding another edge. 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. If that edge links two connected components, then an elementary cycle 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.

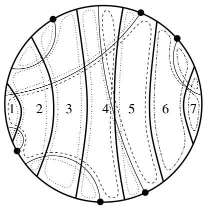
It remains to relate results on reduced forms back to the original inscribed breakpoint graph formulation; we illustrate the process in Figure 15, where the left part shows the solution obtained on a reduced form and the right part shows the corresponding solution inscribed in the circle.

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 algorithmic solutions.



(a) the solution embedded through a reduced form



(b) the solution embedded on the circle

Fig. 15. An optimal solution to the reduced instance in Figure 13

Note that research in orthology assignment based on rearrangement data does not aim to replace assignment based on sequence data: instead, the two approaches complement each other. Biologists are already routinely using the notion of contiguous gene blocks in their determination of orthology assignments: an assignment based on rearrangement data simply formalizes that insight. How to use both sequence data and gene-rearrangement data within the same framework remains a tantalizing, but for now elusive goal.

Acknowledgments

This work is supported by the US National Science Foundation under grants EF 03-31654, IIS 01-13095, IIS 01-21377, and DEB 01-20709, and by the US National Institutes of Health under grant 2R01GM056120-05A1.

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. D. Bryant. The complexity of calculating exemplar distances. In D. Sankoff and J. Nadeau, editors, *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pages 207–212. Kluwer Academic Publishers, Dordrecht, NL, 2000.
7. A. Caprara. On the tightness of the alternating-cycle lower bound for sorting by reversals. *J. Combin. Optimization*, 3:149–182, 1999.
8. X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Computing the assignment of orthologous genes via genome rearrangement. In *Proc. 3rd Asia Pacific Bioinformatics Conf. (APBC'05)*, pages 363–378. Imperial College Press, London, 2005.

9. 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, Dordrecht, NL, 2000.
10. 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.
11. 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, Berlin, 2004.
12. 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, Berlin, 2000.
13. N. El-Mabrouk. Reconstructing an ancestral genome using minimum segments duplications and reversals. *J. Comput. Syst. Sci.*, 65:442–464, 2002.
14. 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.
15. 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.
16. M. Marron, K.M. Swenson, and B.M.E. Moret. Genomic distances under deletions and insertions. *Theor. Computer Science*, 325(3):347–360, 2004.
17. 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, UK, 2005.
18. C. Thach Nguyen, Y.C. Tay, and L. Zhang. Divide-and-conquer approach for the exemplar breakpoint distance. *Bioinformatics*, 21(10):2171–2176, 2005.
19. D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):990–917, 1999.
20. D. Sankoff and M. Blanchette. The median problem for breakpoints in comparative genomics. In *Proc. 3rd Int'l Conf. Computing and Combinatorics (COCOON'97)*, volume 1276 of *Lecture Notes in Computer Science*, pages 251–264. Springer Verlag, Berlin, 1997.
21. D. Sankoff and M. Blanchette. Multiple genome rearrangement and breakpoint phylogeny. *J. Comput. Biol.*, 5:555–570, 1998.
22. D. Sankoff, D. Bryant, M. Deneault, B.F. Lang, and G. Burger. Early Eukaryote evolution based on mitochondrial gene order breakpoints. *J. Comput. Biol.*, 7(3):521–536, 2000.
23. 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 Publishers, Dordrecht, NL, 2000.
24. J.C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishers, Boston, MA, 1997.
25. 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.
26. 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.
27. G. Tesler. Efficient algorithms for multichromosomal genome rearrangements. *J. Comput. Syst. Sci.*, 65(3):587–609, 2002.