Visualizing Measures of Genetic Distance

Eric M. Jordan[1]

Abstract

Recent research has investigated the effects of specific types of evolutionary events on a genome, including transpositions [BP95], inversions [BP93] (signed [KS94, HP95] or unsigned [KS95]), and translocations [KR95]. Many of these results derive lower bounds on the minimum number of operations required to transform one gene sequence into another by examining alternating cycles in certain "dart graphs" that can be constructed from a permutation. The lower bounds can then be used to construct approximation algorithms and branch and bound algorithms. We will show that there is a correspondence between dart graphs and what we will call "embedded graphs," such that the alternating cycles correspond to the faces of the embedded graph. Furthermore, the lower bound on the number of transpositions derived in [BP95] is the genus of the embedded graph, and the lower bound on the number of signed reversals derived in [BP93] is the crosscap number. The topological interpretation comes about because the correspondence between a dart graph and the associated embedded graph is a consequence of the fact that both are representations of a combinatorial representation of a closed surface, known as a rotation system [GT87]. For the implementation challenge, we wrote a program that takes a permutation, and displays the associated embedded graph on an appropriate surface, to help illustrate the structure that these quantities are measuring. We hope that this understanding of what alternating cycles represent will reveal intuition that can be used to further the research on the genome rearrangement problem. Rather than trying to automate the process of reconstructing the sequence of evolutionary events that transformed one gene sequence into another, we feel that we have created a tool that researchers can use to visualize the relationship between two specific sequences. We start by defining dart graphs and embedded graphs, and show how they correspond. We then give a straightforward algorithm for realizing an embedding of these graphs in an appropriate surface. Finally, we provide simpler proofs of the lower bounds on transpositions and signed reversals.

[1]Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. emjordan@theory.lcs.mit.edu. Research supported by a graduate fellowship from the Office of Naval Research.

1 Introduction

We start by showing that the graphs used to derive lower bounds on the signed inversion distance and the transposition distance of a permutation are in a one to one correspondence with embedded graphs.

A surface can be represented by what is called a rotation system ([GT87]). A rotation system is equivalent to a graph, with two exceptions. First, there is a circular ordering of edges associated with each vertex, rather than just a set of edges. Second, in a rotation system, there is a "twist bit" associated with each edge, that specifies whether the edge is flat (type 0) or twisted (type 1) [Tut79, GT87].

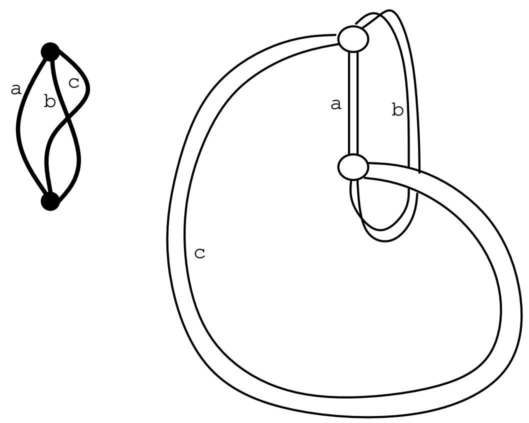

Figure 1: An embedded graph (with only flat edges), and one way to construct the corresponding surface (A torus). A band decomposition replaces each vertex with a deformation of a disk (0-cells), and each edge with a strip (1-cells), to get a surface with boundary. The boundaries (faces) can then be filled in with deformations of a disk (2-cells) to get a closed surface.

A straightforward way to represent a rotation system in a computer is to associate a circular linked list (an "adjacency list") of edges and their twist bits with each vertex. This representation of a surface is what we will call an "embedded graph." Figure 1 shows how to construct a surface from an embedded graph.

[Tut79] indirectly suggests an alternative way to represent a rotation system in a computer. In the original paper, Tutte only considers rotation systems in which all of the edges are flat. He represents each undirected edge in a rotation system with two directed edges, known as "darts." The rotation system is then determined by two permutations, P and θ, on the set of darts. θ maps a dart $d = (v_1, v_2)$ onto its inverse $d^{-1} = (v_2, v_1)$ (the dart pointing the other way), and P maps d onto the one that follows it in the circular ordering associated with v_1. With Tutte's method, the edges in a rotation system are no longer distributed among representations of the vertices; rather, the relationships among the edges are represented directly by the permutations, and the vertices are represented implicitly as the orbits of P. One way to incorporate twisted edges into this representation of a rotation system is to associate a twist bit with each pair of darts $(d, \theta d)$.

The idea of representing darts and the relationships between them directly leads to the concept of a "dart graph", illustrated in Figure 2. Using a dart graph to represent a rotation system may not be as natural as using an embedded graph, but we introduce them here because, as we will see, the graphs that have been used to prove lower bounds on signed reversal distance and on transposition distance are dart graphs.

A dart graph is a directed, edge-colored graph that has one vertex for each pair of darts $(d, \theta d)$, and one directed edge for each dart. Each vertex $(d, \theta d)$ has two incoming edges d and θd, and two outgoing edges Pd and $P\theta d$. The colors in a dart graph are used to specify which incoming edge is associated with which outgoing edge. If none of the edges in the rotation system are loops, then it suffices to choose one color for each vertex in the rotation system, and assign that color to every dart that has that vertex as its tail. The embedded graphs that we will be looking at have no loops, so we will use this convention.

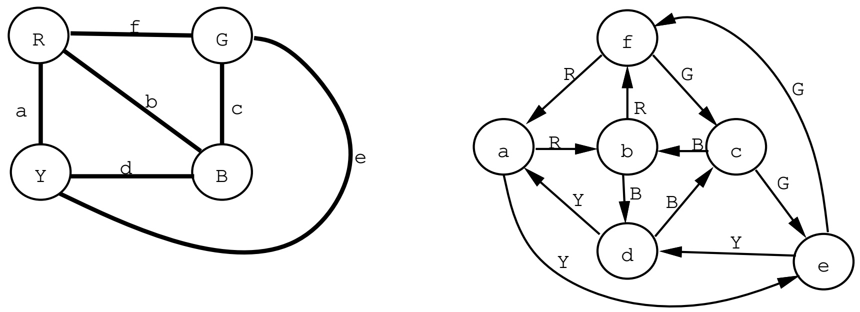

Figure 2: An embedded graph, and the corresponding dart graph. All edges are flat. We adopt the convention that the adjacency lists represent the counter-clockwise ordering of the edges about the vertices. The order of the edges about any given vertex is significant in the embedded graph, but not the dart graph. R, G, B, and Y stand for red, green, blue, and yellow, respectively.

To summarize, the edges of a rotation system are represented by edges in an embedded graph, and by vertices in a dart graph. The circular ordering of edges about each vertex in a rotation system are represented by adjacency lists in an embedded graph, and by colored, directed cycles of edges in a dart graph. There is one other piece of information that can be extracted from a rotation system that we have not discussed yet: the faces. The face tracing algorithm for a rotation system is described in detail in [GT87]. The faces are important, because they provide the combinatorial structure that is used to characterize the surface. Tutte showed that when there are no twisted edges, the faces are traced out by the permutation $P\theta$. In an embedded graph, this captures the intuition that the "angles" formed by the edges as the leave the vertices can be used to trace out the faces. With our convention that P designates the counter-clockwise ordering of darts emerging from each vertex, $P\theta$ will consistently trace out the counter-clockwise angle (going from b to f to c, for example, in figure 2). In a dart graph, Tutte's observation makes it easy to trace out the faces. The edge that represents d points to a vertex whose outgoing edges represent Pd and $P\theta d$. Pd will have the same color as d, so to trace out the face, we follow the edge that leaves the vertex that d points to, that is colored differently from d. (See figure 3.)

Note that the permutation θP^{-1} traces out the same faces, but in the opposite direction. The face tracing algorithm for a graph that contains twisted edges switches between using $P\theta$ and θP^{-1} to trace out the faces. We start out using one of these two permutations, and every time we traverse a twisted edge, we switch to the other permutation. For an embedded graph, this means that a twisted edge makes us switch from tracing out counter-clockwise angles, to tracing out clockwise angles. For a dart graph, this means that the first time we traverse a twisted edge in the face tracing algorithm, we start going against the direction on the edges.

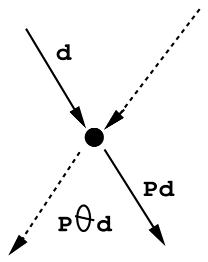

Figure 3: Tutte's notation, applied to a dart d in a dart graph.

Now, given a signed permutation π on the integers from 1 to n, we can construct a dart graph as follows: Extend the domain of π, so that $\pi_0 = +0$ and $\pi_{n+1} = +(n+1)$. (Note that by arbitrarily designating these two elements as being positive, we are destroying the initial equivalence of the two signs.) Create $n+2$ vertices, and add directed grey edges from vertex i to vertex $i+1 \bmod n+2$, and directed black edges from vertex π_i to vertex $\pi_{((i-1) \bmod n+2)}$, for all i from 0 to $n+1$. Designate a vertex as flat if the sign on the associated marker is $+$, and twisted if $-$. The resulting dart graph only has two colors, so the cycles traced out by the face tracing algorithm are alternating cycles. Figure 4 illustrates the relationship between a permutation, its dart graph, and its embedded graph.

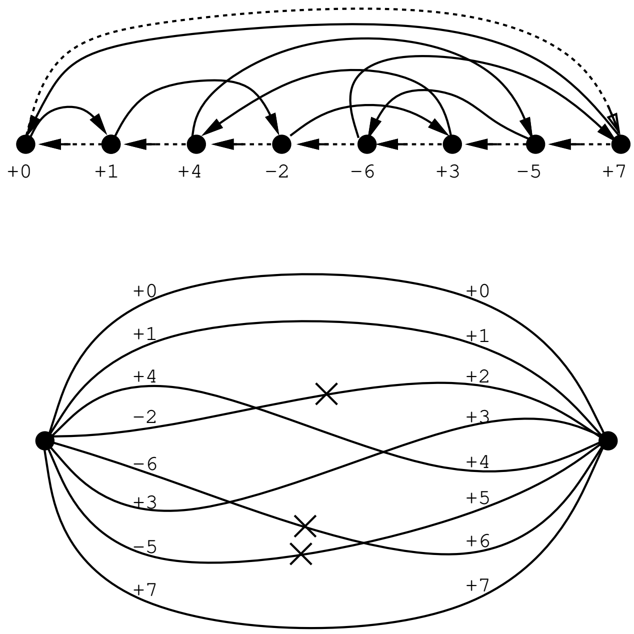

Figure 4: The dart graph and the embedded graph for the permutation +1 +4 -2 -6 +3 -5. An X indicates a twisted edge. The alternating cycles of the dart graph are the same as the faces of the embedded graph: 0 1, 0 7, and 2 1 4 3 5 6 3 2 4 5 7 6. (When tracing out faces, remember to switch directions when you traverse a twisted edge. If you forget to do this and act as if all of the edges are flat, then you will get the faces 1 2 4 5 3 4 and 2 3 6 7 5 6.)

If only transpositions occur, then the resulting embedded graph has only flat edges. The dart graph is then the same as the cycle graph described in [BP95] with the addition of edges from 0 to $n+1$ and back. Let c be one less than the number of faces in the embedded graph (which equals the number of alternating cycles in the dart graph after removing the edges going from 0 to $n+1$ and back). Applying Euler's formula [GT87, pages 122–124], the genus g of the embedded graph satisfies $2-(n+2)+(c+1) = 2-2g \Rightarrow g = (n+1-c)/2$. But, $(n+1-c)/2$ is the first lower bound on the transposition distance in [BP95]. A similar but slightly more involved argument shows that the dart graph is equivalent to the breakpoint graph introduced in [BP93] and used for signed inversions in [HP95], and the cycle graph used to prove the same lower bound in [KS94]. Here, Euler's formula can be used to show that the crosscap number of the embedded graph provides the same lower bound. We will make some comments about what the proofs of these lower bounds look like when applied directly to the embedded graphs, using topological terminology, in a later section.

2 Constructing the Surface

The classification theorem tells us that the closed surface associated with any given embedded graph is either homeomorphic to the connected sum of a sphere with handles, or homeomorphic to the connected sum of a sphere with crosscaps. (The connected sum of a surface with a crosscap is realized by poking a hole in the surface, and identifying the boundary of the hole with the boundary of a Mobius band. The result is a nonorientable surface, and therefore cannot be embedded in $\Re^3$. Our program will overcome this by drawing the surface with a hole in it.) If the surface is nonorientable, then the crosscap number $\overline{\gamma}$ is the number of crosscaps needed to realize the surface. If it is orientable, then the genus is the number of handles. A surface that has c crosscaps and h handles, where $c \geq 3$, is homeomorphic to a surface with $c-2$ crosscaps and $h+1$ handles. We will use this theorem to construct surfaces that are composed of both handles and crosscaps. The crosscap number of a surface with c crosscaps and h handles is clearly $2h+c$.

We can construct a surface from a rotation system by attaching strips to the boundaries of disks, using the rotation system to determine the order of the ends of the strips around the disk and whether the strips are flat or twisted. (See figure 1). In general, this construction does not give a decomposition of the surface in terms of handles and crosscaps. This is unfortunate in general, because it does not tell us when two surfaces are homeomorphic. It is also unfortunate for us in particular, because we are trying to visualize the genus and the crosscap number. Fortunately, we can manipulate the rotations system until it is in a standard form, that does describe the surface in terms of crosscaps and handles. It is essential, however, when doing this manipulation, to realize that once we start changing the way that we are going to construct the surface, there is no longer a one to one correspondence between the edges that are going to be embedded in the surface, and the strips that are used to construct the surface. In this section, we show how to derive a description of the surface in terms of handles and crosscaps, along with a specification of how the edges behave when the graph is embedded in the surface. Our algorithm is based on a close examination of the proof of the classification theorem in [FG82].

First, we contract edges in the embedded graph until there is only one vertex. For the embedded graph corresponding to a permutation π, we can contract both of the two edges that we added when we extended π to 0 and $n+1$. The adjacency list for the resulting vertex can then be written as a cyclic "word" ${\pi_1}^{e_1}{\pi_2}^{e_2}...{\pi_n}^{e_n}n^{-1}(n-1)^{-1}...1^{-1}$, where e_i is 1 if edge i is flat, and -1 if edge i is twisted. A word represents a rotation system with a single vertex. In general, the elements of a word are chosen to be letters, not numbers, because we are only interested in the pairing and sequence of the elements in the word. For example, the word for the permutation in figure 4 is: $adb^{-1}f^{-1}ce^{-1}f^{-1}e^{-1}d^{-1}c^{-1}b^{-1}a^{-1}$.

The important realization that enables us to convert the proof of the classification theorem to an algorithm for embedding graphs in surfaces is that there is a subtle distinction between the word that is used to construct the surface (as illustrated in figure 1), and the word that represents the edges embedded in the surface. We are going to perform "substitution moves" on the original word to derive words that represent the same surface, constructed in different ways. As we do so, we must keep track of the substitutions that we make. Then, we can substitute back to determine the list of edges that go down each strip.

A substitution move is represented by an equation $l' = s_1 l s_2$, where l' is a new letter, that will be used to eliminate l. This equation tells us to replace all occurrences of l with ${s_1}^{-1} l' {s_2}^{-1}$, and all occurrences of l^{-1} with $s_2 l'^{-1} s_1$. Finally, we simplify the word by cancelling adjacent inverses. As noted above, however, we must keep track of the equation representing this substitution.

Substitution moves are justified in theory because they correspond to a series of Tietze moves. They work because we are maintaining the invariant that if we substitute letters in the word back until we only have letters that were in the original word, then we can cancel adjacent inverses to get the original word back.

By applying substitution moves, we can work the description of the surface into a form that we recognize and know how to create (by adding handles and crosscaps to a sphere, for example), without losing track of how the original edges are embedded in the surface. If the surface is created using the procedure illustrated in figure 1, then back substitution can be used to tell you the edges that go down each strip; where the strips come together, the edges should match up as well.

For example, if there are elements x and y in the original edge word, such that the elements x, y, x^{-1}, and y^{-1} appear in the word in that order, then we say that x and y are a separated pair. We can now use substitution moves to work this separated pair into a handle $xyx^{-1}y^{-1}$:

1. The original word is $AxByCx^{-1}Dy^{-1}E$.
2. Let $p = D^{-1}xC^{-1}$.
3. The word is now $ADpCByp^{-1}y^{-1}E$.
4. Let $q = CBy$.
5. The word is now $ADpqp^{-1}q^{-1}CBE$.

The result is that we can embed the edges in a handle $pqp^{-1}q^{-1}$, by having the edges $p = D^{-1}xC^{-1}$ go under the handle, and the edges $q = CBy$ go down the handle. The reason that the equation tells us to have the edges bundled in B, C, and D go in and then out again is because the equations are based on the idea that the edges are traveling on strips that leave from and return to a single vertex.

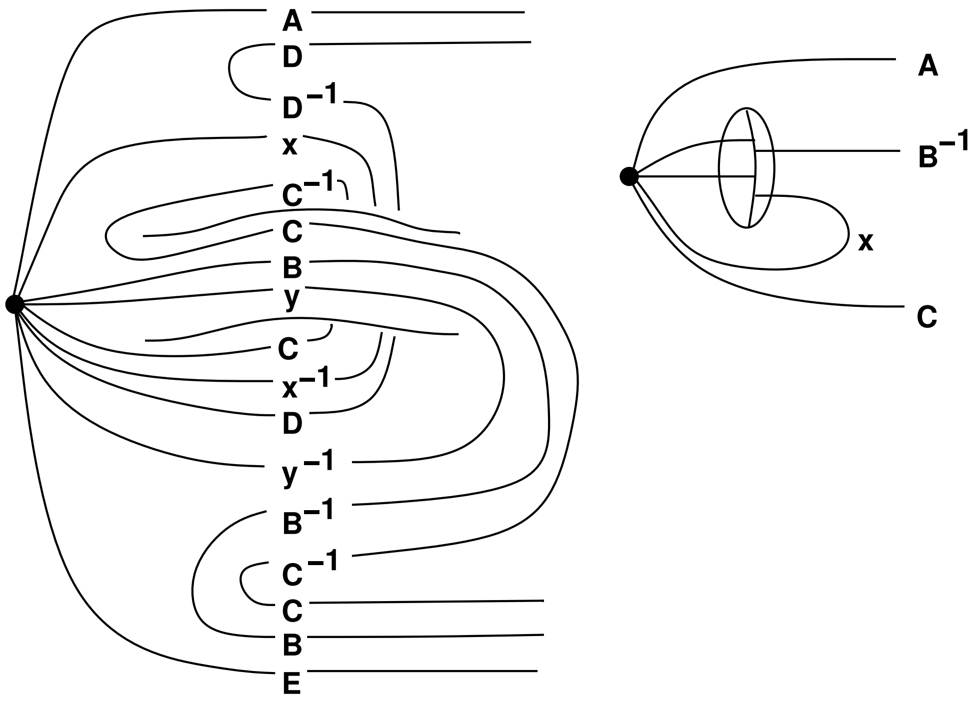

Figure 5: A handle can be used to reduce the problem of embedding the word $AxByCx^{-1}Dy^{-1}E$ to the problem of embedding the word $ADCBE$. A crosscap can be used to reduce $AxBxC$ to $AB^{-1}C$.

Based on this, figure 5 illustrates a procedure that can be used to eliminate a separated pair by introducing a handle. Figure 5 also shows that a crosscap can be used to simplify a word that contains a twisted edge: $AxBxC$ can be turned into $AyyB^{-1}C$ with the substitution $y = xB$. In theory, any surface that contains a single crosscap is homeomorphic to a surface that contains only crosscaps and no handles. We can derive such a surface using substitution moves, or we can stop after deriving a surface that contains both crosscaps and handles. The latter has the advantage that it gives a simpler embedding, in that it has fewer substitution moves.

In a word that contains only flat edges, it is always possible to find a separated pair $AxByCx^{-1}Dy^{-1}E$ in which D is empty. First, simplify the word by eliminating any substrings of the form "xx^{-1}". Then, assume the claim is false. This means that each adjacent pair of letters must be a non-separating pair, so the word must look like (starting from the last adjacent pair, and moving left one by one): $...a^{-1}...b^{-1}...c^{-1}...d^{-1}...dcba$. As we move left, we get closer and closer to the matching inverse, until eventually, we will find a substring of the form "$x^{-1}x$".

Our algorithm constructs a surface by looking for a twisted edge and adding a crosscap and simplifying the word accordingly if it finds one, and otherwise looking for a flat separated pair of the form $...x...y...x^{-1}y^{-1}....$ Future versions will be more general, and will allow arbitrary series of substitution moves to be performed, and will try to find sequences of substitution moves that result in less complicated embeddings.

3 The Lower Bounds

An interesting variant on the embedded graph associated with a permutation can be constructed by expanding the vertices as shown in figure 6 for our sample permutation. The resulting graph represents the same surface, because it is possible to contract edges until we get back to the original graph. There is one other operation besides contracting and expanding edges that can be performed on an embedded graph that gives a representation of the same surface: You are allowed to "flip" the disk that is represented by one of the vertices. Specifically, you can reverse the adjacency list associated with any vertex, as long as the twist bit of each edge in the adjacency list is switched once each time it appears in the list. By doing this operation, it is possible to derive a graph like the one on the right of figure 6. Note that in this form, twisted edges represent a *change* in sign.

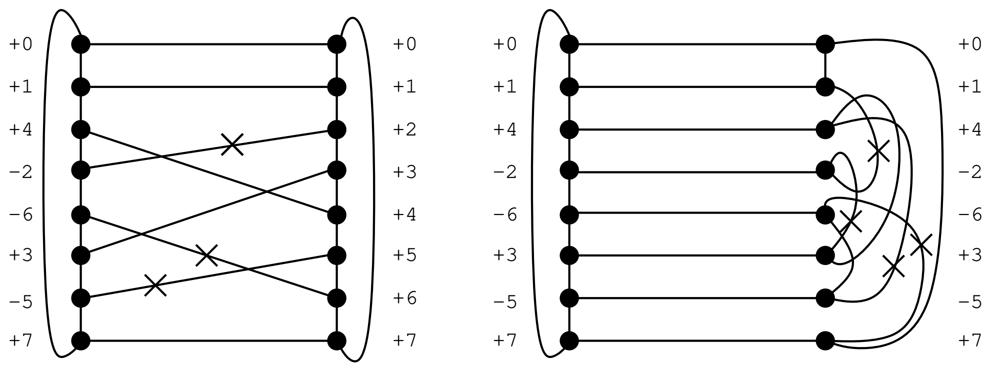

Figure 6: Both of these embedded graphs are equivalent to the one in figure 4.

Figure 7 uses this form to illustrate the local effect that transpositions and signed reversals have on the embedded graph. It is possible to prove the lower bounds on signed reversals and transpositions directly at this point by showing that a signed reversal changes the number of faces in the embedded graph by -1, 0, or 1, (causing the crosscap number to change by 1, 0, or -1), while a transposition changes the number of faces by -2, 0, or 2 (causing the genus to change by 1, 0, or -1). The details of these proofs go through several cases, describing the different effects that an operation can have on the number of faces. The case analysis is not discussed in detail, because it is fairly straightforward, and is equivalent to the ones presented in the original papers.

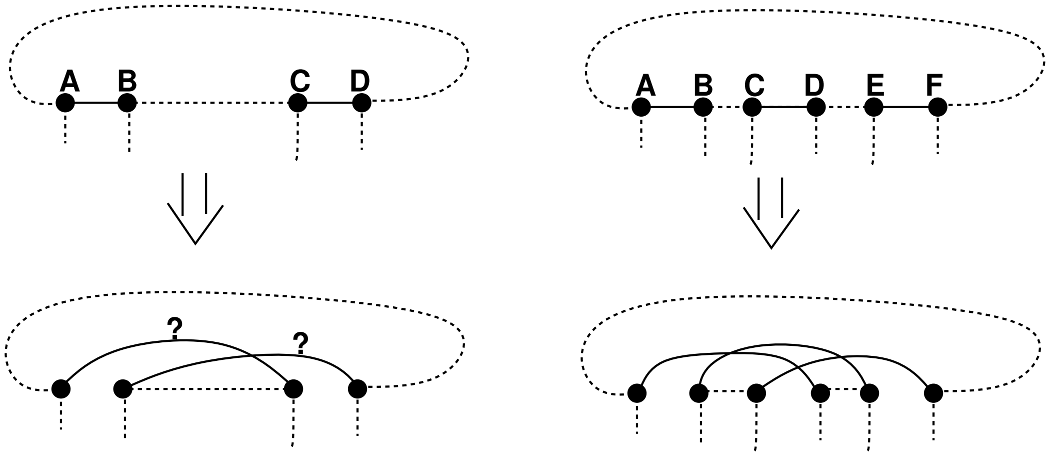

Figure 7: The effect of a signed reversal, and of a transposition. Dotted lines are unspecified parts of the graph; solid lines are specific edges. The question marks indicate that edge AC is twisted if and only if there are an even number of twisted edges in any path from A to C in the original graph (and similarly for BD).

Slightly different proofs can be given by appealing to the classification theorem. For a given sequence of m signed reversals, there is an embedding of the corresponding graph G in a surface S with m crosscaps. Specifically, put G in the form shown on the left of figure 6. Then, for each reversal, use a crosscap to reverse the order of the edges in the center section of G that correspond to the elements that the reversal acts on. The result is an embedding of G in a surface with m crosscaps. Now, "it can be shown as a generalization of the Schoenfliess theorem that for any imbedding $G \to S$, the graph G is contained in the 1-skeleton of a triangulation of the surface S" [GT87, page 100]. The 1-skeleton of a triangulation of S is itself an embedded graph T, whose crosscap number is m. Since G is a subgraph of T, and adding edges can only increase the crosscap number, the crosscap number of G is less than or equal to m. The argument is similar for the genus number and transpositions, with handles "transposing" the order of edges as they go from one side of G to the other.

However, the proof of Euler's formula (which is used to compute the crosscap number of G) depends on the use of the number of faces as a potential function to show that when the word representing a 1-vertex embedded graph is reduced to a normal form, you always end up with the same number of edges. Therefore, appealing to the classification theorem still gives you a proof that is based on the use of the number of faces as a potential function.

It seems very intuitive that handles correspond to transpositions, because they take a set of edges from one place to another, and that crosscaps correspond to reversals, because they reverse a sequence of edges. The "cheating" that makes these quantities only lower bounds is due to the fact that *both* ends of each edge emanate from the same vertex, and one by one, the ends of the edges are tied together. This makes it possible, for example, for a handle to be traversed by both ends of the same edge. What we really want is for the edges to travel from a "source" vertex, through the handles and crosscaps in a consistent direction, to a "destination" vertex.

4 Plans and Future Work

We will use our program to visualize simulated biological data, random permutations, the permutation described by the mouse-man data provided by the organizers of the workshop, and theoretically interesting permutations, like the Gollan permutation.

The surface associated with a given embedded graph can be realized in many ways, with different sets of crosscaps and handles, and different sets of edges going down each. Yet, all realizations of an embedded graph are homeomorphic. How does changing the order in which separating pairs and twisted edges are chosen affect the appearance of the resulting surface? We will address this question by generalizing our program to work with any sequence of substitution moves.

Different embedded graphs may represent homeomorphic surfaces, but the classification theorem makes it easy to determine when this is the case. Recall, however, that even though two embedded graphs may be equivalent in the sense that they represent homeomorphic surfaces, it is not necessarily possible to continuously deform one into the other; being homeomorphic only requires that there be a mapping that preserves neighborhoods [Sti93]. Intuitively, this means that properties of a surface that are invariant under a homeomorphism are properties of the surface, and not properties of the embedding or of the space that the surface is being embedded in. The disadvantage, however, is that there is structure in the embedded graphs that is missed when you only look at properties like the Euler characteristic. Can the mathematics used to investigate the structure of knots, braids, and embeddings of surfaces be applied to the genome rearrangement problem?

5 Acknowledgements

I am grateful to Bonnie Berger, Daniel Kleitman, and David Wilson for their feedback and support.

References

[BP93] Vineet Bafna and Pavel Pevzner. Genome rearrangements and sorting by reversals. In *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, pages 148–157. IEEE, 1993.

[BP95] Vineet Bafna and Pavel Pevzner. Sorting permutations by transpositions. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 614–623, 1995.

[FG82] P. A. Firby and C. F. Gardiner. *Surface Topology*. Ellis Horwood Limited, 1982.

[GT87] Johnathan Gross and Thomas Tucker. *Topological Graph Theory*. Wiley interscience series in discrete mathematics and optimization. Wiley-Interscience Publication, 1987.

[HP95] Sridhar Hannenhalli and Pavel Pevzner. Transforming cabbage into turnip. In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computation*, 1995.

[KR95] John Kececioglu and R. Ravi. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 604–613, 1995.

[KS94] John Kececioglu and David Sankoff. Efficient bounds for oriented chromosome inversion distance. In *Proceedings of the Fifth Annual Symposium on Combinatorial Pattern Matching*, volume 807 of *Lecture Notes in Computer Science*, pages 307–325. Springer-Verlag, 1994.

[KS95] John Kececioglu and David Sankoff. Exact and approximation algorithms for sorting by reversals, with applications to genome rearrangement. *Algorithmica*, 13:180–210, 1995.

[Sti93] John Stillwell. *Classical Topology and Combinatorial Group Theory*. Number 72 in Graduate Texts in Mathematics. Springer-Verlag, second edition, 1993.

[Tut79] W. T. Tutte. Combinatorial oriented maps. *Can. J. Math.*, XXXI(5):986–1004, 1979.