Simulating Mitochondrial Network Dynamics

Raphie Mostov

December, 2023

Abstract

In this primer for the public showcase of my work, I discuss the mathematical representation of mitochondrial morphology as graphs and how they are simulated in a computer program. An overview of some interesting properties of mitochondria graphs is also provided. Please not that everything discussed is still actively being worked on.

1 Mitochondria Networks are Graphs

The formalism of graphs presents a clear choice for describing mitochondrial network morphology. As shown in Figure 1 we can assign a node to each network junction, and to the terminal ends of network branches, to create a graph representation. Since every graph that embeds in a sphere also embeds in a plane [Wes01], it follows that graphs created this way are plane geometric graphs. That is, graphs that have an assigned planar embedding, and whose edges have a physical length.

For our purposes, we will not incorporate the physical distances into our description. This is perhaps the largest simplification our model will make. Since the mitochondria of budding yeast appear free to move around on the cell surface and adjust the lengths of their branches, we will assume that edge length places a negligible constraint on our random walker.

The vast majority of nodes observed in mitochondria graphs either have a degree of one or three. Degree four nodes have been observed, however at a statistically insignificant amount [Suk+10]. The X-shaped junctions of a degree four node appear to be mechanically unstable and seem to always resolve into pairs of degree three nodes with Y-shaped junctions. This leaves us with a working description of mitochondria networks as graphs belonging to the following family:

Definition 1.0.1. A *mitochondria graph*, M, is a plane graph made up of exclusively degree three and degree one nodes, (where we consider self-loops to contribute 2 to a node's degree). The planar embedding is given by its dual graph, M^D. It cannot have more degree three nodes than n_{max}.

The last constraint mentioned, N_{max}, is provided to make the number of mitochondria graphs, and the size of the state space, finite. There must be

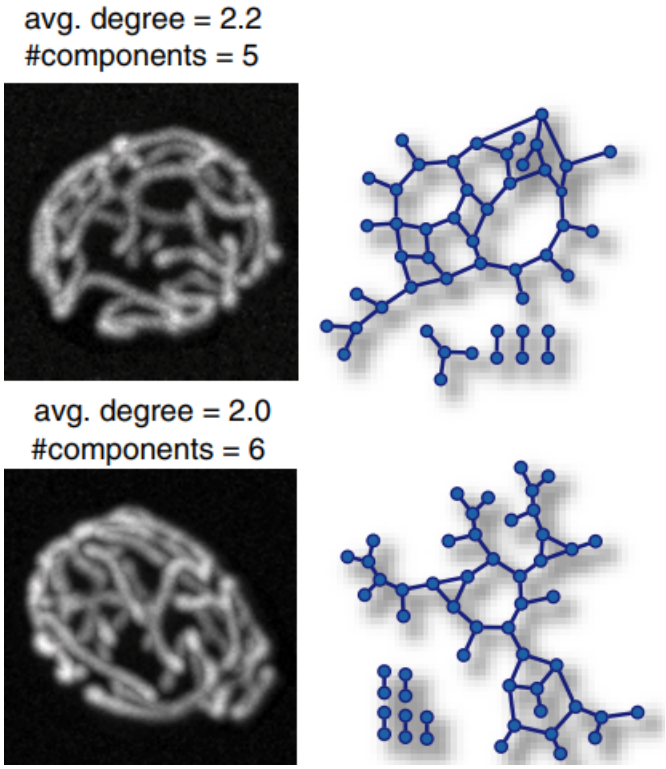

Figure 1: [Via+20] Confocal microscopy images of mitochondrial networks in budding yeast cells (left) and their respective mitochondria graphs (right).

some physical constraint on the number of nodes as they are correlated with the volume of a mitochondrial network, which is itself constrained by the size of a cell. This cutoff N_{max} should be chosen so that it is large enough to include all observed mitochondria graphs, but small enough to be physically realistic and computationally practical.

Notation 1.0.2. We use the following set of notation to denote features of a mitochondria graph.

- N: set of nodes for M, set of faces for M^D.
- F: set of faces for M, set of nodes for M^D
- E: set of edges for M, likewise E^D set of edges for M^D.
- $\deg(i)$: degree of node i.
- n: number of degree 3 nodes of M. And, p: number of degree 1 nodes of M.
- n_{max}: maximum number of degree 3 nodes

The formalism of a mitochondria graph allows us to also clearly define the morphological transformations as transitions within the state space. As it turns out, the precise definition in terms of graphs is rather involved, and these transitions are best understood with the image provided in Figure 2.

1.1 Encoding Plane Graphs

A key feature of mitochondria graphs is that they are plane, not planar, graphs. This means that the specific embedding matters and this is why mitochondria graphs always include a dual graph M^D to describe how they embed in the plane. For instance, it is only possible to tip-to-tip fuze two nodes if they are embedded within the same face, otherwise the fusion would break planarity. For this reason, we need a way of detecting if two edges are embedded in the same face, and so we define the following map

Definition 1.1.1. Given an edge $ij \in E$ with $i, j \in N$, the *boundary map* $\partial : E \to E^D$ gives us

$$\partial(ij) = kl$$

the corresponding edge in the dual graph with $k, l \in F$. If $k = l$, then the edge ij is said to be *embedded* within face k. Otherwise, ij is on the *boundary* of faces k and l. Given two edges $ij, pq \in E$, if

$$\partial(ij) \cap \partial(pq) \neq \emptyset$$

then ij and pq are said to be *embedded in the same face.*

With this, boundary map, we can then define the operations that make up our state space transitions,

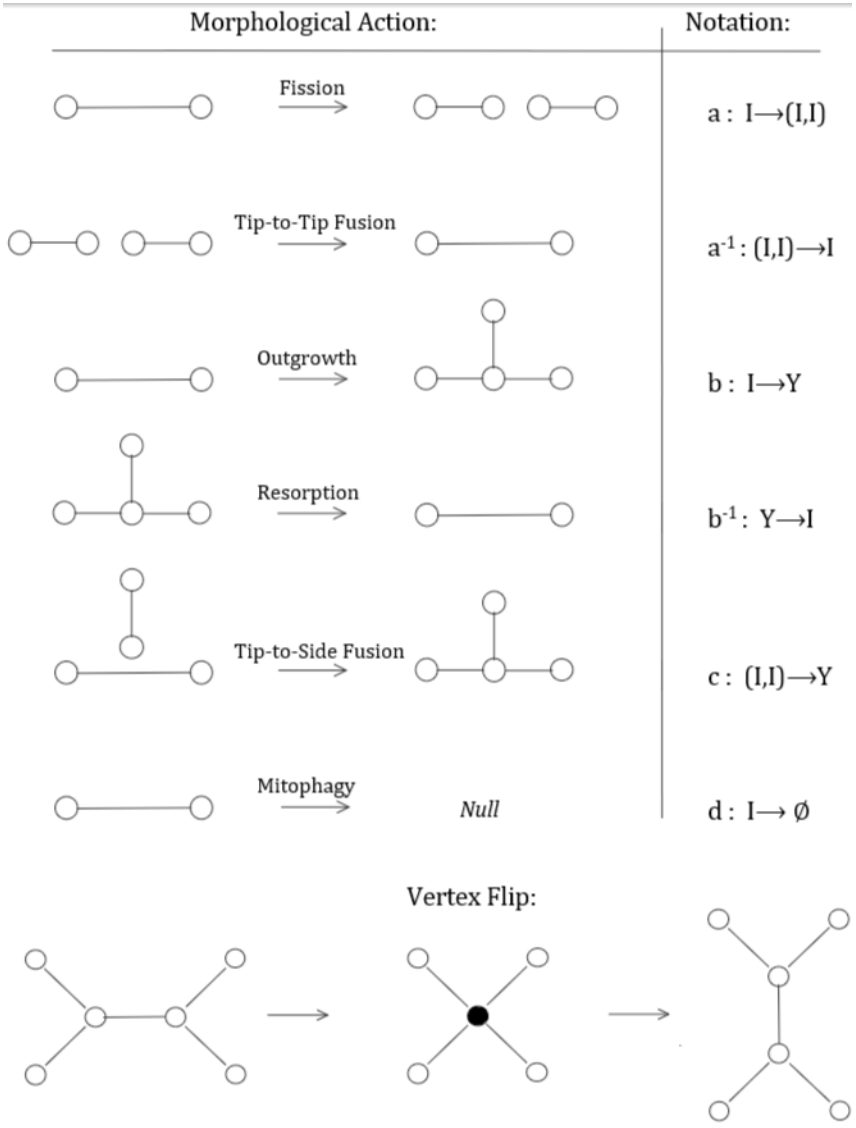

Figure 2: A visual description of the 7 types of state space transitions, each referred to here as a "Morphological Action," that can take place on a mitochondria graph. This is a slightly modified version of a figure from [LM-7]

.

Definition 1.1.2. For a mitochondria graph M, the operation **delete edge** (ij) removes the edge ij from M, and, since the dual edge is always linked, it also removes the edge $\partial(ij)$ from M^D. Likewise, **add edge**(ij, kl) adds the edges ij to M and kl to M^D and assigns the mapping $\partial(ij) = kl$

Finally, we can start defining the state space transitions and where they are possible

2 Simulating State Space Transitions

2.1 Mitophagy

This is the simplest transformation, so we will define it first. We can perform mitophagy, $d(M, ij) = M'$, when the following conditions are both met:

- $\deg(i) = \deg(j) = 1$
- $|N| > 2$

And this transition is achieved by executing a **delete edge** (ij) operation.

2.2 Fission

We can perform fission, $a(M, ij) = M'$, on any edge of the mitochondria graph, provided that $|N| \leq N_{max} - 2$, with the following sequence of operations:

1. **add edges** $(ii_1, \partial(ij))$ and $(ij_1, \partial(ij))$, where i_1 and j_1 are newly created nodes.
2. **delete edge** (ij)
3. If M has one fewer face than before, then we remove the face l from M^D and the boundary map.

We will provide more details on detecting if a face was removed, and how we remove it from the boundary map in the next section.

2.3 Tip-to-Tip Fusion

We can perform tip-to-tip fusion, $a^{-1}(M, ii_1, jj_1) = M'$, when the following conditions are met:

- ii_1 and jj_1 are embedded in the same face
- Without loss of generality (WLOG), $\deg(i) = \deg(j) = 1$

We perform this transition by executing the following sequence of operations:

1. **add edge** $(i_1j_1, \partial(ii_1))$.
2. **delete edges** (ii_1) and (jj_1).

3. If M has one more face than before, we add a new face to M^D and the boundary map.

We will cover in detail how we add a new face to M^D and the boundary map in the next section.

2.4 Outgrowth

We can perform outgrowth, $b(M, ij, kk) = M'$, when the following conditions are met

- $|N| \leq N_{max} - 2$
- ij is on the boundary of or embedded in the face k.

We perform this transition by executing the following sequence of operations:

1. **add edges** $(iq, \partial(ij))$, $(jq, \partial(ij))$, and (qq_1, kk), where q and q_1 are newly added nodes.
2. **delete edge** (ij)

2.5 Resorption

We can perform resorption, $b^{-1}(ij, jj_1, jj_2)$, so long as WLOG $\deg(i) = 1$, by executing the following sequence of operations:

1. **add edge** $(j_1j_2, \partial(jj_1))$.
2. **delete edges** (ij), (jj_1), and (jj_2).

2.6 Tip-to-Side Fusion

We can perform tip-to-side fusion, $c(M, ii_1, jj_1) = M'$, when the following conditions are met

- Edges ii_1 and jj_1 are embedded in the same face.
- WLOG $\deg(j) = 1$.

We perform this transition with the following sequence of operations:

1. **add edges** $(ij, \partial(ii_1))$ and $(i_1j, \partial(ii_1))$.
2. **delete edge** (ii_1).
3. If M has one more face than before, we add a new face to M^D and the boundary map.

2.7 Vertex Flip

We can perform a vertex flip, $e(M, ij, ii_1, ii_2, jj_1, jj_2) = M'$, when the following conditions are met

- There is only one edge between nodes i and j
- WLOG ii_1 and jj_1 are embedded in the same face.
- If any triplet of edges chosen from $\{ii_1, ii_2, jj_1, jj_2\}$ are embedded in the same face, then the whole set must be embedded in the same face.

The last condition exists so that vertex flips will always maintain planarity of the mitochondria graph – we cannot "move" a pendant edge into a different face without breaking planarity. We perform a vertex flip with the following set of operations:

1. Update the boundary map so that ij gets mapped to a newly added edge in the dual graph $[\partial(ii_1) \cap \partial(ii_2)][\partial(jj_1) \cap \partial(jj_2)]$.
2. **add edges** $(i_1 j, \partial(ii_1))$ and $(ij_1, \partial(jj_1))$.
3. **delete edges** (ii_1) and (jj_2).

We should note that there can be cases in which the intersections $\partial(ii_1) \cap \partial(ii_2)$ and $\partial(jj_1) \cap \partial(jj_2)$ return more than one face. More details on how these cases are handled are provided in Section **??**

2.8 Adding and Removing Faces from the Boundary Map

Detecting when a new face was created or removed by a transition is a fairly straightforward application of Euler's formula for planar graphs, $|N|-|E|+|F| = 2$. Removing a face from the boundary map, as is the case with fission, is not much more difficult. We simply replace all instances of the face l with the face k.

But perhaps the most complicated part of simulating the state space transitions on a computer is adding a new face to the boundary map and dual graph. It is here where I will admit that my computer program for simulating a random walker still sometimes produces errors. These errors most likely are a result of bugs in the implementation of this algorithm or the complex process of implementing a vertex flip.

To begin our exposition, notice that the two state space transitions which can create a new face, tip-to-tip and tip-to-side fusion, do so by partitioning a preexisting face into two faces with the addition of a new edge.

Let $f \in F$ be the face that gets partitioned by the newly added edge ij, and let g be the newly added face. Then, the algorithm for finding the edges that are on the boundary of and embedded in g is as follows:

1. Find G_f, the induced subgraph of M given by all the edges that are embedded in, or on the boundary of, f.

2. Find the second shortest path in G_f between nodes i and j. This path, along with the edge ij forms the initial cycle C_0 in the set of cycles C_g that compose the boundary of our new face g.

3. Calculate the minimum cycle basis, B_{min}, of G_f. This is the smallest set of cycles such that any cycle in G_f can be expressed as the formal sum of cycles (symmetric difference in this case) in B_{min}.

4. Separate the cycles of B_{min} into the disjoint cycles $D = \{C \in B_{min} : C \cap C_0 = \emptyset\}$, and joint cycles $J = \{C \in B_{min} : C \cap C_0 \neq \emptyset\}$

5. Remove all the edges from G_f that are in the joint cycles J but not in the initial boundary cycle C_0.

6. For each node k in G_f, if there does not exist a path between k and i, then remove that node from G_f.

7. For each disjoint cycle $C \in D$, if $C \subset G_f$, then add it to the set of boundary cycles C_g for the new face.

8. The edges in C_g are on the boundary of g, and the remaining edges in G_f are embedded in g. Update ∂ and M^D accordingly.

One thing to note here is that a cycle basis is only technically defined for simple graphs, however, mitochondria graphs can have multiple edges between the same pair of nodes. As such, the available tools for computing a cycle basis require us to input a version of G_f that is simple. Adapting this algorithm to account for multigraphs has been the source of many bugs.

3 Mathematical properties of mitochondria graphs

3.1 We Don't Know How Large the State Space Is

One question that we must address is *how many mitochondria graphs are there*? This is closely related to an open problem in mathematics, which asks how many unlabeled planar graphs there are for a given amount of nodes. Even though it is difficult to know the exact number of mitochondria graphs for a given amount of nodes, upper and lower bounds can be established, as shown in [LM-7] (and some of the results we will provide may be able to help improve these bounds).

This means that we will not be able to find a presumed stationary distribution of the state space directly, and this is why we will choose to instead simulate many random walkers in our experiment.

3.2 The State Space is Irreducible

Even though it is currently beyond our capabilities to know how many mitochondria graphs there are, we can provide a rather simple argument to show

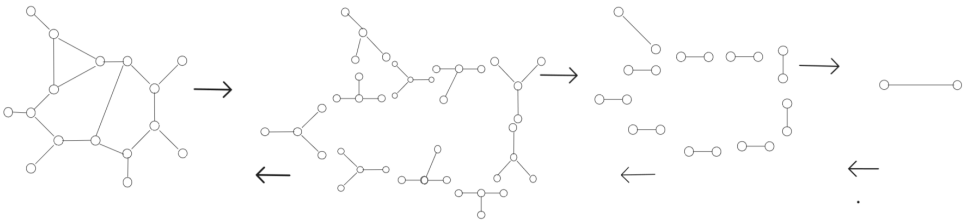

Figure 3: A visual explanation of why it is possible to interconvert between any two mitochondria graphs. We can perform a sequence of transitions that take any mitochondria graph to an I-shaped graph, and we can reverse these transitions to reach any other mitochondria graph.

that the state space is irreducible. So, at least in theory, the state space can have a steady state distribution.

As shown in Figure 3, given any mitochondria graph, we can always perform a sequence of fissions to break the graph down into a bunch of Y-shaped and I-shaped connected components. We can resorb all of the Y-shaped components, and then fuze the remaining connected components into a single I-shaped graph. Because these transitions have inverses, we can reverse this process to get to any other possible mitochondria graph.

Although tip-to-side fusion does not directly have an inverse, we see that we can produce an effective inverse for the transformation by performing resorption followed by fission. The fact that all of the transformation shave inverses, either directly, or through some composition of other inverses provides opportunity to apply Group Theory, and more broadly, Abstract Algebra to the study of mitochondrial dynamics.

3.3 Degree Ratio as a Test Statistic

Oftentimes, the simpler a formula is, the better. Our choice of test statistic, the average ratio of degree three nodes to total nodes, is no exception. For a data set with S observations, we have

$$T = \frac{1}{S} \sum_{i=1}^{S} \frac{n_i}{|N_i|}$$

Choosing the degree ratio as a test statistic offers a unique advantage, as it holds the potential to provide us with unexpected insights. For example, if we observe a significant difference in the ratios between our null model and data sets, it prompts us to consider the possibility of an undiscovered mechanism governing this ratio.

Before delving into such inquiries, it becomes crucial to establish the values of T that should be considered extreme. To accomplish this, understanding

both the theoretical and realistic upper and lower bounds of T is essential. The following result aids in deriving these bounds:

Lemma 3.3.1. *Let M be a mitochondria graph with n degree 3 nodes and a single connected component, then*

1. *The maximum amount of degree one nodes p that M can have is $n+2$.*
2. *If $n > 0$, then the minimum amount of degree one nodes p is 0 if n is even and 1 if n is odd. (The minimum is 2 when $n = 0$).*

Proof. 1. Consider the induced subgraph $Y \subseteq M$, where the nodes of Y are given by $N_Y = \{y \in N_M : \deg(y) = 3\}$ and $n = |N_Y|$. Since there are p degree one nodes and all are adjacent to n degree three nodes, it follows by definition of M and Y that

$$p + \sum_n \deg(y_n) = \sum_n 3 = 3n \quad \text{where } y_n \in N_Y, \text{ and } \deg(y_n) \in \{1, 2, 3\}$$

By the Handshaking Lemma, we have $p = 3n - 2|E_Y|$. Hence to maximize p, we must minimize the number of edges in Y. From Proposition 1.3.13 in [Wes01], the minimum number of edges in Y is $n-1$. Thus, the maximum of p is $p = n + 2$.

2. Consider the cases in which $n = 1$ and $n + 1 = 2$. For the $n = 1$ case, it follows from Proposition 1.3.5 in [Wes01], that we must have an even number of odd-degree nodes, and hence we must have another odd-degree node that is not a degree-three node. Hence the minimum amount of degree-one nodes for the $n = 1$ case is $p = 1$ (this graph would have a single self-loop and be shaped like a "P").

   For the second base case, $n + 1 = 2$, we can perform outgrowth on the graph given by $n = 1$, and as a consequence, we must produce a new degree-one node. We can then fuze the two degree-one nodes together into a single edge resulting in a graph with $p = 0$ degree-one nodes for the $n + 1$ case. We can inductively continue this process of performing outgrowth, and then performing tip-to-tip fusion as soon as two degree one nodes are available. .

□

It follows that the theoretical upper bound is $T = 1$, in which case the mitochondria graphs would all be completely triangulated. Additionally, the theoretical lower bound is $T = 0$, in which case the mitochondria graphs would just all be copies of a graph with one edge connecting two nodes (Lemma 3.3.1 applies only to graphs with a single connected component). However, it is not unreasonable to expect that the vast majority of mitochondria graphs have at least one degree three node, otherwise, the mitochondria would simply not form a "network". Thus, we should realistically never expect to see a value of $T = 0$.

Work done by [Suk+10] showed that mitochondria networks tend to be at a critical percolation threshold, with one large connected component joined by a few far smaller components. Thus, to a good approximation, we can apply our result from Lemma 3.3.1 to get a realistic lower bound of T that is slightly less than 0.5.

We have reason to believe that this realistic lower bound also applies to our null model. Because mitophagy is only possible on small connected components, and it does not have an inverse, we can expect small connected components to disappear before they grow big, leaving behind only a few large components. Thus, we can expect to rarely see $T < 0.5$ in both the data and the null model.

Indeed, in the dataset provided by [Via+20] which we are using, our value for T_0 is about 0.55 for the wild-type strain, which is very close to our realistic lower bound on T. Because the variation in T will occur almost entirely between 0.5 and 1, it is clear that we should reject a T-value when $T > T_0$.

3.4 Determining Initial Walk Distance

As stated previously, mitochondrial dynamics do not have a clearly defined initial condition. Because we are assuming the mitochondria graphs we observe are at a steady state, our ultimate goal will be to simulate random walkers starting from the mitochondria graphs in our dataset.

However, our first preliminary experiment will be simulating every random walk starting from the simplest possible mitograph – this is a mitochondria graph, I, with a single edge connecting two nodes (shaped like the letter I).

Assuming the null hypothesis, how many transitions should we expect the random walker to take before it has reached a steady state? How far should we let our random walker travel before we start sampling the states that it visits? I have provided the following proposition to help decide on this initial distance:

Proposition 3.4.1. *Let M be a mitochondria graph with n degree 3 nodes, and let C be the number of connected components in M. Furthermore, let I be the mitochondria graph consisting of a single edge joining two nodes. Then the distance (shortest path in the state space) from I to M is $n + C - 1$.*

Proof. This proof rests on the following claims,

1. If $C = 1$, then the distance from I to M is n.
2. If $n = 0$, then the distance from I to M is $C - 1$.

Because there are no transitions that can increase n and C at the same time, Proposition 3.4.1 follows from claims 1 and 2.

Proof of claim 1: In the case that the number of degree 1 nodes is $p = n + 2$, the direct pathway is given by $b^n(I)$ (See Lemma 3.3.1, this is the maximum value of p). Otherwise when $p < n + 2$ we have:

$$c^x b^y(I) = M$$

where $x = \frac{n-p+2}{2}$ and $y = \frac{n+p-2}{2}$. It follows that $x + y = n$. Because there are no transitions that increase n by more than one, and every increase in n uses up a single transition, this is the shortest path.

PROOF OF CLAIM 2: If $n = 0$, then M consists of $C-1$ copies of I. It follows that $a^{C-1}(I) = M$ is the shortest path, as there is no other transition that produces more than one additional connected component in a single step. □

Thanks to Proposition 3.4.1, we can tell how far any mitochondria graph is from our initial state I. Certainly, we should let the random walker travel at least the average distance before we sample the states it visits. However, this would be assuming that the walker would take a "direct path" to a steady state. To be safe, our implementation uses an initial walk distance of N_{max}, as the number of nodes in a mitochondria graph is always larger than its distance to I.

References

[Wes01] Douglas B. West. *Introduction to Graph Theory*. Pearson Education Inc., 2001. ISBN: 8178088304.

[Via+20] Matheus P. Viana et al. "Mitochondrial Fission and Fusion Dynamics Generate Efficient, Robust, and Evenly Distributed Network Topologies in Budding Yeast Cells". In: *Cell Systems* 10.e1-e5 (2020), pp. 287–297. DOI: https://doi.org/10.1016/j.cels.2020.02.002.

[LM-7] Greyson Lewis and Wallace Marshall. "Mitochondria Networks Through the Lens of Mathematics". In: *Physical Biology* 20.5 (2023-7), p. 051001. DOI: https://dx.doi.org/10.1088/1478-3975/acdcdb.

[Suk+10] Valerii M. Sukhorukov et al. "Emergence of the Mitochondrial Reticulum from Fission and Fusion Dynamics". In: *PLOS Computational Biology* 8.10 (2012-10), pp. 1–13. DOI: https://doi.org/10.1371/journal.pcbi.1002745.