Modeling Horizontal Gene Transfer via Class II Transposons and Deciphering Genomic Composition Changes in Fungal Populations

Chaohui Li
chl221@ucsd.edu

Laurel Li
sil089@ucsd.edu

Mason Kellogg
m1kellogg@ucsd.edu

1. Introduction

Transposition refers to the movement of a particular genetic sequence, called a transposon, from one part of the genome to another. Transposons, specifically Class II (cut-and-paste) transposons, play a pivotal role in the adaptive evolution of organisms by facilitating the rearranging of genetic material within the genome of one organism and transmission (and stable integration) of genetic material between genomes of separate organisms. These elements significantly contribute to horizontal gene transfer (HGT) - the process by which genetic material is passed from one organism to another non-progeny organism. This mechanism is particularly critical in fungi, organisms that thrive in often stressful and dynamically changing environments. The mobility of Class II transposons within fungal genomes allows for the rapid acquisition of novel traits, thereby enhancing genetic diversity and resilience far more swiftly than traditional vertical gene transfer methods. A study on the Penicillium genus identified 60 HGT events involving 190 genes acquired from bacteria and indicates that gene transfer plays a significant role in shaping the genomes of Penicillium fungi [1]. A novel group of Class II-like transposons called Starships were also identified that are linked to many recent horizontal gene transfer (HGT) events in fungi [2]. This work suggests HGT driven by transposons is an ancient and evolutionarily mechanism for moving adaptive traits within and between species, highlighting the complexity and long-term impacts of transposons in shaping fungal genomes [3].

Transposons (TRA) consist of a transposase gene flanked by reverse complementary terminal inverted repeats (TIR). These TIRs function as transposase binding sites for excision. In the case of Class II transposons, the TRA is cut from the double-stranded genome, able to insert elsewhere in the genome or in another genome. Likely, transposons have an affinity for non-deleterious insertion sites [4], but in the current study, we will assume an equal probability of insertion in any portion of the genome. Upon insertion of a TRA into a DNA sequence, identical target site duplications (TSD) are added to either side of the TRA. The length of the TSD is unique to the TRA. The sequence of nucleotides is roughly consistent, but some single-nucleotide differences sometimes occur per insertion [4], with the percentage of mutation a property of the TRA. Note that during excision, the TSD sequences remain and represent a good "footprint" for tracing TRA movement through a population of genomes [4]. The structure of a transposon and mechanism of excision and insertion is simplified in the below figure from Addgene [5].

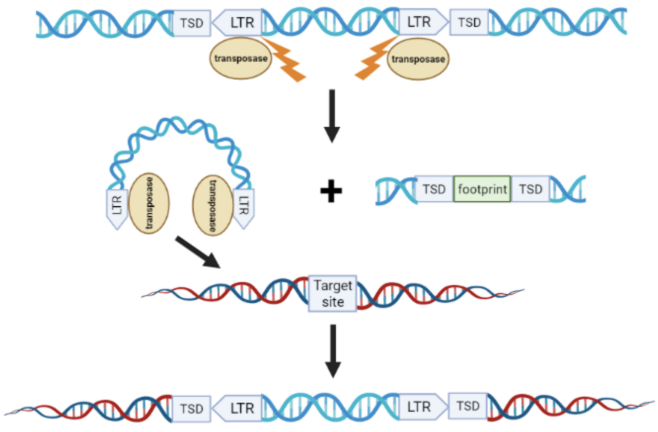

Figure 1. Overview of Class II transposon transposition. Note that LTR is synonymous with TIR.

Understanding the excision-insertion behavior of Class II transposons is essential for comprehending fungal evolutionary biology and has profound implications for agriculture (both in terms of crop diseases and biocontrol agents), and the development of gene therapies. In fact, many fungal genomes consist of 30% transposable elements [6]. Our biological inquiry focuses on the modeling of horizontal gene transfer within a fungal population, specifically in regard to Class II transposons. Given genomic data from a fungal population, we aim to reconstruct the most likely path of a Class II transposon through the population.

2. Problem Statement

2.1. Definitions

DNA, TRA, and TSD are strings with the alphabet {A, T, C, G}, with length ranges $[10^5, 10^6]$, $[10^2, 10^3]$, and $[20, 30]$, respectively. A *transposon* TRA consists of an encoding region (transposase gene) flanked by two equal-length (from 20 to 30 nucleotides) reverse complementary

regions (*terminal inverted repeats* TIR). Each character of the first terminal inverted repeat maps to each character of the reversed second repeat following the rule that {A, T, C, G} are mapped to {T, A, G, C} respectively.

Below we define a few actions involving these strings:

- *Transposon insertion event*: a process in which a TRA inserts at randomly selected position p (0-indexed) of a DNA, then the DNA is transformed to $DNA[:p] + TSD + TRA + TSD + DNA[p:]$, where "+" indicates string concatenation.
- *Transposon movement event*: for a *transposon* TRA moving from DNA_1 to DNA_2, delete TRA from DNA_1 and insert it into DNA_2 (via a *transposon insertion event*). TSD_1 pairs remain in DNA_1. The TSD_2 created in DNA_2 has the same length t as TSD_1 in DNA_1, and their Hamming distance is at most 1. The score between TSD_1 and TSD_2 is $score(TSD_1, TSD_2) = \sum_{i=0}^{t-1} S[TSD_1[i]][TSD_2[i]]$, where S is a 2D matrix as follows. (Table. 1)

TRA: TTACGACGTGATCGACTCGACGTAA
DNA_1: ACTGGACTTGATCG
DNA_2: TATACGGACTACGACCATGCGC

Transposon insertion event (TRA inserts into DNA_1 at position $p_1 = 8$):
DNA_1: ACTGGACTACTTACGACGTGATCGACTCGACGTAAACTTGATCG
DNA_2: TATACGGACTACGACCATGCGC

Transposon movement event (TRA moves from DNA_1 to DNA_2 at position $p_2 = 16$):
DNA_1: ACTGGACTACTACTTGATCG
DNA_2: TATACGGACTACGACACGTTACGACGTGATCGACTCGACGTAAACGCATGCGC

Figure 2. Example of a transposon insertion event and transposon movement event in a DNA population of size 2. The TRA is composed of the transposase gene (red) flanked by terminal inverted repeats (blue). TSDs are shown in green.

Table 1 depicts the TSD transition score matrix. The rows represent "from" and the columns "to". So, a transition from TSD_1 to TSD_2 where $TSD_1 = ACT$ and $TSD_2 = ACG$ has a score of 5, $score(TSD_1, TSD_2) = 5$.

	A	C	G	T
A	0	4	3	1
C	4	0	2	3
G	3	2	0	5
T	1	3	5	0

TABLE 1. SCORE MATRIX S.

A *multipartite graph* with n parts is an undirected graph (since the score matrix S is symmetric), where the set of nodes can be partitioned into n parts in such a way that there are no edges between nodes in the same part. (Fig. 3) Given a TRA with a collection of DNAs and TSDs, we can construct a weighted multipartite graph, where each part corresponds to each DNA, and each node corresponds to a potential TSD in this DNA. The weight between two potential TSDs (Hamming distance ≤ 1) from different DNAs is $score(TSD_i, TSD_j) + 2|p_i - p_j|/(p_i + p_j)$. The first term defines the difficulty of mutation from one nucleotide to another in a TSD, and we assume that a transposon tends to generate TSDs with lower scores. The second term defines a normalized genomic distance between two TSDs, and we also assume that a transposon tends to move between TSDs that are genomically close. Therefore, the smaller weight of two TSDs indicates that the TSDs are more likely to be involved in the transposition event.

A *traversing path* in a multipartite graph is a path that visits every part exactly once. The score of a path is the sum of the weight of all edges in the path. (Fig. 3) In terms of transposition, the traversing path with the minimal score represents an ordered path of TSDs that the transposon created when moving throughout all DNAs.

Lists of TSDs = [("ACT", 1, 20), ("CCT", 2, 30), ("ACC", 2, 80), ("AGT", 2, 180), ("CTT", 3, 70), ("ACT", 3, 80)]

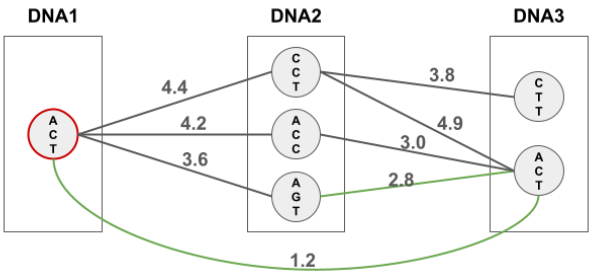

Figure 3. A multipartite graph with the optimal traversing path. Each TSD is a tuple like ("ACT", 1, 20), which indicates that the TSD "ACT" is at position 20 of DNA 1. The red node is the last TSD created by the TRA. The green edges compose a path of TSDs that the transposon created when visiting DNA2, DNA3, and DNA1 in order.

2.2. Computational Problem

Based on the definitions, we can formulate the **Finding Traversing Paths in a Multipartite Graph** problem.

Input: A weighted multipartite graph with a node v corresponds to TSD in DNA where the process ended.

Output: A traversing path in this graph that has a minimal score among all traversing paths starting at v.

3. Methods

3.1. Data simulation

We generated our own synthetic data [7] based on experimental data of transposons from The Transposon Registry,

as well as fungal genome literature [1] [3] [6] [8]. Nucleotides were drawn from a uniform random distribution for any DNA sequence. Note that for a fungal DNA population, we are simulating movement of a single transposon (TRA). We also assume the TRA inserts into each DNA in the population exactly once, and resides in the final DNA visited. Inputs into our synthetic data generator were as follows:

- Number of DNA sequences in the population
- DNA length range
- TRA length range
- TSD length range
- Probability of 1 Hamming distance edit of TSD on each insertion event

The output consisted of: original TSD on first insertion, TRA, original DNA population before transposon insertion or movement, final DNA population after transposon movement, insertion path (list of tuples of (TSD, index of DNA inserted into, index of transposon insertion)).

3.2. Finding TRA and TSDs

Given a list of DNAs, we find the transposon (TRA) and all possible TSDs based on the TSD flanking the TRA in the population of DNA sequences. Those TSDs will be used to conudct the multipartite graph for optimal traversing path finding.

3.2.1. Finding TRA and the TSD flanking it. Based on the special structure of transposon, the TRA and the TSD flanking it can be found by a two-stage judgment, including the finding of TIR and TSD.

The number of certain-length reverse complementary regions could be limited in population with 0.25^{length} probability statistically, which is helpful for finding TIR as well as TRA and its position.

Based on the length range of TSD, the TSD along with its position and length, can be identified by searching for two duplicated sequences flanking the already discovered TRA.

3.2.2. Finding all possible TSDs. Coupled with distinctive principles of transposon insertion and movement, all possible TSDs and their positions can be identified by searching for two consecutive duplications of a certain length which decided by TSD found in previous step.

Because the TRA inserts into each DNA in the population exactly once and resides in the final DNA visited, all DNAs need to be searched except for the DNA where TRA was found.

3.3. Finding an optimal traversing path

Given a list of TSDs on n DNAs, we construct a multipartite graph with n parts, where a node represents a TSD, and an edge connects two potential TSDs with a weight $score(TSD_i, TSD_j) + 2|p_i - p_j|/(p_i + p_j)$. Note the starting node v is given as an input.

We implemented two methods for solving the Finding Traversing Paths in a Multipartite Graph problem.

3.3.1. Depth-First Search. To find the optimal traversing path, we can use a brute-force algorithm like Depth-First Search (DFS) to traverse the multipartite graph. Starting at v, we try to find a path that visits each of the n parts exactly once, which is a traversing path. During traversal, we can recursively find all traversing paths and calculate their scores. Finally, the traversing path with the minimum score is output.

3.3.2. Integer Linear Programming. The other way to solve this problem is Integer Linear Programming (ILP). We can regard it as an optimization problem. According to the definitions, a traversing path is a Hamiltonian path, and we can reduce the problem to a Hamiltonian cycle problem by creating a dummy node d. Besides, we add virtual edges with weight 0 between d and every other node in the graph. (Fig. 4) With the edge set E including virtual edges, we have the objective function

$$\min \sum_{(i,j)\in E} w_{ij}x_{ij} \tag{1}$$

where w_{ij} is the weight of edge (i, j), and $x_{ij} = 0$ or 1 is a variable that indicates whether the edge from i to j is selected in the solution. Note that edge (i, j) and edge (j, i) represent the same edge with different directions.

To find the optimal traversing path, we can simply find the optimal cycle that visits each part exactly once. Hence, we have the constraint that each part P has 1 in-degree and 1 out-degree.

$$\sum_{i\in P, j\notin P} x_{ij} = 1, \sum_{i\in P, j\notin P} x_{ji} = 1 \tag{2}$$

Besides, each part P has exactly one node with 1 in-degree and 1 out-degree, while others have no degrees.

$$\sum_{k\notin P} x_{ik} + x_{kj} \leq 1, \forall i, j \in P \tag{3}$$

Moreover, each edge has at most one direction selected, and there must be an edge between v and d in the solution.

$$x_{ij} + x_{ji} \leq 1, \forall (i, j) \in E \tag{4}$$

$$x_{vd} + x_{dv} = 1 \tag{5}$$

Finally, for any subset S of all vertices V, there must be at least two edges between S and $V \setminus S$.

$$\sum_{i\in S, j\notin S} x_{ij} + x_{ji} \geq 2, \forall S \subset V \tag{6}$$

With the formulas above, we can efficiently solve the optimal cycle with an ILP solver like PuLP [9]. Since there is a dummy node d, we can simply delete d and convert the cycle into an optimal traversing path that starts with v.

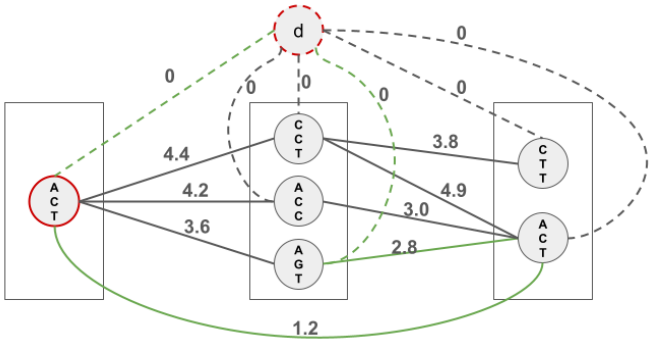

Figure 4. A multipartite graph with a dummy node d. The red node "ACT" is the node v.

4. Results

We benchmarked our DFS and ILP methods with 100 simulated samples, where each sample has a traversing path from the simulation process. Since there is randomness during data simulation, the simulated traversing path is not necessarily the optimal solution. We ran DFS and ILP on the 100 samples and calculated the scores of the optimal paths from both methods. As a result, both DFS and ILP reconstructed the optimal paths for all samples. As Fig. 5 shows, DFS and ILP output the same score for each of these samples, and the score is even smaller than the score from the simulation in some samples.

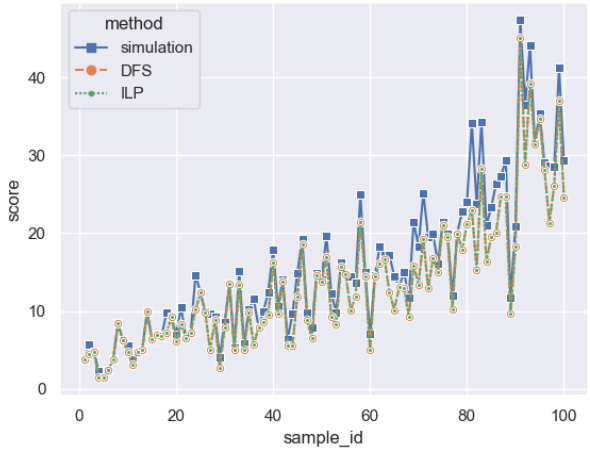

Figure 5. Scores of the optimal paths of 100 samples.

We also measured the running time in seconds of DFS and ILP methods. In terms of the varying sample size, we compared the running time over the number of parts (DNA), nodes (TSD), and edges in the multipartite graph. (Fig. 6-8)

As Fig. 6 illustrates, both DFS and ILP efficiently solved the problem within 1 second when the number of DNAs was less than 10, while ILP performed much worse than DFS when the number of DNAs was greater than 10.

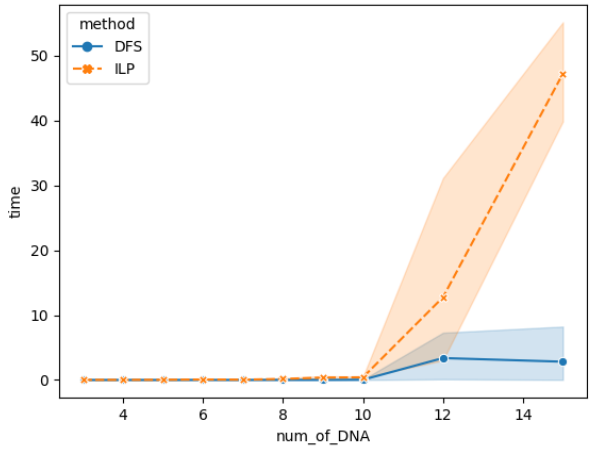

Figure 6. Running time with varying number of DNAs, i.e., the number of parts in the multipartite graph.

The running time over the number of TSDs showed a similar trend. (Fig. 7) However, there was a special case when the number of TSDs was 25, ILP had a shorter running time than DFS. When the number of edges was greater than 70, there were also a few cases, where ILP resolved the problem faster than DFS. (Fig. 8)

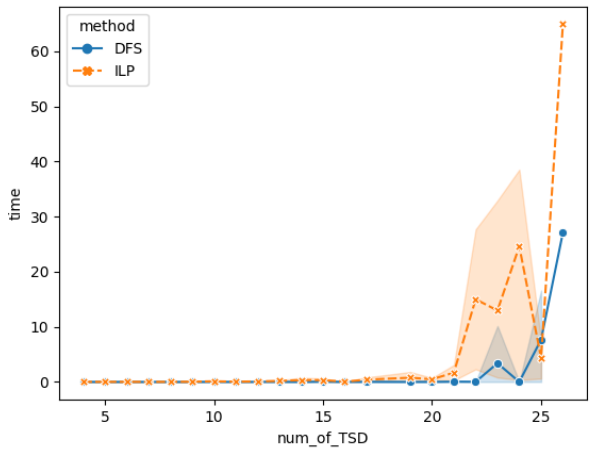

Figure 7. Running time versus the number of TSDs, i.e., the number of nodes in the multipartite graph.

Overall, both DFS and ILP can efficiently reconstruct the optimal traversing path when the number of parts is small ($\leq$ 10), while DFS has a better performance than ILP in terms of running time when the sample size is large. There are some special cases, i.e. the number of nodes is less than

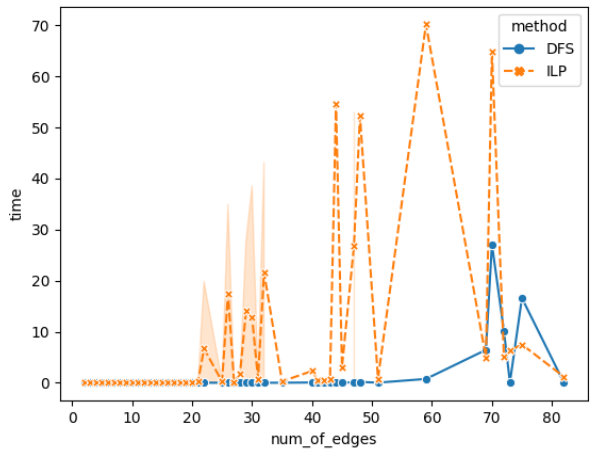

Figure 8. Running time versus the number of edges in the multipartite graph.

25 and the number of edges is greater than 70, where ILP has a better time efficiency.

5. Conclusion

Understanding the movement patterns of Class II transposons is essential for comprehending fungal evolutionary biology. In this paper, we formulated the biological problem of tracing the transposition history of a transposon, as a graph-based computational problem. Based on the biological properties of Class II transposons, we proposed a multipartite graph model (with DNA sequences as parts and potential TSDs as nodes) that creatively represents a fungal DNA population. To decipher the historical movement of a transposon in a population, we designed two methods based on DFS and ILP to reconstruct the optimal traversing path in the multipartite graph.

To benchmark our methods, we also simulated synthetic data based on experimental data of transposons from previous work. Both DFS and ILP methods successfully returned the optimal traversing path, which represents the most probable transposition of the transposon in a population. Moreover, DFS has a better performance than ILP in terms of running time. All codes and data are available at https://github.com/Eric0627/TransTracker.

6. Future Work

In this section, we address a few areas for improvement on this computational problem. First, if the TSD length is short and the DNA length is long, there are more possibilities for erroneous "TSD-like" sequences being detected in our DNA population to multipartite graph construction. For a transposon with a characteristically short TSD, more advanced biological models may have to be introduced to detect potential TSDs (nodes in each part of the graph). Luckily, the typical length of TSDs is long enough (oftentimes longer than 20 nucleotides) to have an extremely low occurrence of mistaken TSDs.

Next, in this study, we only model a single transposon moving through a population. In practice, there are often many transposable elements moving through a fungal population at ay given time. We believe that our proposed method will generalize well for tracking many transposons, however, issues may arise when the TSD of one transposon is the same length as another transposon. This may lead to confusion as to which TSD signatures belong to which transposon, which will increase the size of the search space (number of edges in multiparte graph) considerably, and may lead to non-optimal paths.

Additionally, the TSD score matrix can be made asymmetric. This would require directed edges in the multipartite graph (potential 2x increase in number of edges), but the same algorithms would apply.

Moreover, we assume that each position in the fungal DNA sequence has an equal likelihood of transposon insertion. In practice, we know that this is an oversimplification. There is a large body of literature on where Class II transposons tend to insert (often repetitive, non-deleterious regions). Expanding our model for transposon insertion site likelihood will improve our TRA search algorithm as well as improve the chance that the TSDs we select for nodes in the multipartite graph are accurate.

References

[1] Wang, M. & Ruan, R. *Genome-wide identification and functional analysis of the horizontally transferred genes in Penicillium.* Genomics, 112(6), 5037–5043 (2020).

[2] Bucknell, A. H. & McDonald, M. C. (2023). That's no moon, it's a Starship: Giant transposons driving fungal horizontal gene transfer. Molecular Microbiology, 120, 555–563. https://doi.org/10.1111/mmi.15118.

[3] Richards, T. A., Soanes, D. M., Foster, P. G., Leonard, G., Thornton, C. R., & Talbot, N. J. (2009). Phylogenomic analysis demonstrates a pattern of rare and ancient horizontal gene transfer between plants and fungi. The Plant cell, 21(7), 1897–1911. https://doi.org/10.1105/tpc.109.065805.

[4] Muñoz-López, Martín, and José L. García-Pérez. "DNA Transposons: Nature and Applications in Genomics." Current Genomics, vol. 11, no. 2, 1 Apr. 2010, pp. 115–128, https://doi.org/10.2174/138920210790886871.

[5] Kenkel, Beth. "Plasmids 101: Using Transposons in the Lab." Blog.addgene.org, blog.addgene.org/plasmids-101-using-transposons-in-the-lab.

[6] Castanera, Raúl, et al. "Transposable Elements versus the Fungal Genome: Impact on Whole-Genome Architecture and Transcriptional Profiles." PLOS Genetics, vol. 12, no. 6, 13 June 2016, p. e1006108, https://doi.org/10.1371/journal.pgen.1006108.

[7] https://colab.research.google.com/drive/1aqo6DfksndiZlnR9gkCtQjoKOGoBsAuG?usp=sharing

[8] Tansirichaiya, Supathep, et al. "The Transposon Registry." Mobile DNA, vol. 10, no. 1, 9 Oct. 2019, https://doi.org/10.1186/s13100-019-0182-3. Accessed 4 May 2023.

[9] Mitchell, S., O'Sullivan, M., & Dunning, I. *PuLP : A Linear Programming Toolkit for Python.* Department of Engineering Science The University of Auckland (2011).