Revisiting the Complexity of and Algorithms for the Graph Traversal Edit Distance and Its Variants

Yutong Qiu[1]
Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA, USA

Yihang Shen[1]
Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA, USA

Carl Kingsford[2] ✉
Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA, USA

───── **Abstract** ─────

The graph traversal edit distance (GTED), introduced by Ebrahimpour Boroojeny et al. (2018), is an elegant distance measure defined as the minimum edit distance between strings reconstructed from Eulerian trails in two edge-labeled graphs. GTED can be used to infer evolutionary relationships between species by comparing de Bruijn graphs directly without the computationally costly and error-prone process of genome assembly. Ebrahimpour Boroojeny et al. (2018) propose two ILP formulations for GTED and claim that GTED is polynomially solvable because the linear programming relaxation of one of the ILPs will always yield optimal integer solutions. The claim that GTED is polynomially solvable is contradictory to the complexity of existing string-to-graph matching problems.

We resolve this conflict in complexity results by proving that GTED is NP-complete and showing that the ILPs proposed by Ebrahimpour Boroojeny et al. do not solve GTED but instead solve for a lower bound of GTED and are not solvable in polynomial time. In addition, we provide the first two, correct ILP formulations of GTED and evaluate their empirical efficiency. These results provide solid algorithmic foundations for comparing genome graphs and point to the direction of heuristics that estimate GTED efficiently.

1 Introduction

Graph traversal edit distance (GTED) [6] is an elegant measure of the similarity between the strings represented by edge-labeled Eulerian graphs. For example, given two de Bruijn assembly graphs [19], computing GTED between them measures the similarity between two genomes without the computationally intensive and possibly error-prone process of

---

[1] These authors contributed equally to this work.
[2] Corresponding Author

assembling the genomes. Using an estimation of GTED between assembly graphs of Hepatitis B viruses, Ebrahimpour Boroojeny et al. [6] group the viruses into clusters consistent with their taxonomy. This can be extended to inferring phylogeny relationships in metagenomic communities or comparing heterogeneous disease samples such as cancer. There are several other methods to compute a similarity measure between strings encoded by two assembly graphs [20, 15, 13, 8]. GTED has the advantage that it does not require prior knowledge on the type of the genome graph or the complete sequence of the input genomes. The input to the GTED problem is two unidirectional, edge-labeled Eulerian graphs, which are defined as:

▶ **Definition 1** (Unidirectional, edge-labeled Eulerian Graph). *A unidirectional, edge-labeled Eulerian graph is a connected directed graph $G = (V, E, \ell, \Sigma)$, with node set V, edge multi-set E, constant-size alphabet Σ, and single-character edge labels $\ell : E \to \Sigma$, such that G contains an Eulerian trail that traverses every edge $e \in E$ exactly once. The unidirectional condition means that all edges between the same pair of nodes are in the same direction.*

Such graphs arise in genome assembly problems (e.g. the de Bruijn subgraphs). Computing GTED is the problem of computing the minimum edit distance between the two most similar strings represented by Eulerian trails in each input graph. A trail in a graph is a walk that contains distinct edges and may contain repeated nodes.

▶ **Problem 1** (Graph Traversal Edit Distance (GTED) [6]). *Given two unidirectional, edge-labeled Eulerian graphs G_1 and G_2, compute*

$$GTED(G_1, G_2) \triangleq \min_{\substack{t_1 \in trails(G_1) \\ t_2 \in trails(G_2)}} edit(str(t_1), str(t_2)). \tag{1}$$

Here, $trails(G)$ is the collection of all Eulerian trails in graph G, $str(t)$ is a string constructed by concatenating labels on the Eulerian trail $t = (e_0, e_1, \ldots, e_n)$, and $edit(s_1, s_2)$ is the edit distance between strings s_1 and s_2.

Ebrahimpour Boroojeny et al. [6] claim that GTED is polynomially solvable by proposing an integer linear programming (ILP) formulation of GTED and arguing that the constraints of the ILP make it polynomially solvable. This result, however, conflicts with several complexity results on string-to-graph matching problems. Kupferman and Vardi [10] show that it is NP-complete to determine if a string exactly matches an Eulerian tour in an edge-labeled Eulerian graph. Additionally, Jain et al. [9] show that it is NP-complete to compute an edit distance between a string and strings represented by a labeled graph if edit operations are allowed on the graph. On the other hand, polynomial-time algorithms exist to solve string-to-string alignment [17] and string-to-graph alignment [9] when edit operations on graphs are not allowed.

We resolve the conflict among the results on complexity of graph comparisons by revisiting the complexity of and the proposed solutions to GTED. We prove that computing GTED is NP-complete by reducing from the HAMILTONIAN PATH problem, reaching an agreement with other related results on complexity. Further, we point out with a counter-example that the optimal solution of the ILP formulation proposed by Ebrahimpour Boroojeny et al. [6] does not solve GTED.

We give two ILP formulations for GTED. The first ILP has an exponential number of constraints and can be solved by subtour elimination iteratively [3, 5]. The second ILP has a polynomial number of constraints and shares a similar high-level idea of the global ordering approach [5] in solving the TRAVELING SALESMAN problem [14].

In Qiu and Kingsford [21], Flow-GTED (FGTED), a variant of GTED is proposed to compare two sets of strings (instead of two strings) encoded by graphs. FGTED is equal to the edit distance between the most similar sets of strings spelled by the decomposition of flows between a pair of predetermined source and sink nodes. The similarity between the sets of strings reconstructed from the flow decomposition is measured by the Earth Mover's Edit Distance [23, 21]. FGTED is used to compare pan-genomes, where both the frequency and content of strings are essential to represent the population of organisms. Qiu and Kingsford [21] reduce FGTED to GTED, and via the claimed polynomial-time algorithm of GTED, argued that FGTED is also polynomially solvable. We show that this claim is false by proving that FGTED is also NP-complete.

While the optimal solution to ILP proposed in Ebrahimpour Boroojeny et al. [6] does not solve GTED, it does compute a lower bound to GTED that we call Closed-trail Cover Traversal Edit Distance (CCTED). We characterize the cases when GTED is equal to CCTED. In addition, we point out that solving this ILP formulation finds a minimum-cost matching between closed-trail decompositions in the input graphs, which may be used to compute the similarity between repeats in the genomes. Ebrahimpour Boroojeny et al. [6] claim their proposed ILP formulation is solvable in polynomial time by arguing that the constraint matrix of the linear relaxation of the ILP is always totally unimodular. We show that this claim is false by proving that the constraint matrix is not always totally unimodular and showing that there exists optimal fractional solutions to its linear relaxation.

We evaluate the efficiency of solving ILP formulations for GTED and CCTED on simulated genomic strings and show that it is impractical to compute GTED on larger genomes.

In summary, we revisit two important problems in genome graph comparisons: Graph Traversal Edit Distance (GTED) and its variant FGTED. We show that both GTED and FGTED are NP-complete, and provide the first correct ILP formulations for GTED. We also show that the ILP formulation proposed by Ebrahimpour Boroojeny et al. [6], i.e. CCTED, is a lower bound to GTED. We evaluate the efficiency of the ILPs for GTED and CCTED on genomic sequences. These results provide solid algorithmic foundations for continued algorithmic innovation on the task of comparing genome graphs and point to the direction of heuristics that estimate GTED efficiently.

2 GTED and FGTED are NP-complete

2.1 Conflicting results on computational complexity of GTED and string-to-graph matching

The natural decision versions of all of the computational problems described above and below are clearly in NP. Under the assumption that P $\neq$ NP, the results on the computational complexity of GTED and string-to-graph matching claimed in Ebrahimpour Boroojeny et al. [6] and Kupferman and Vardi [10], respectively, cannot be both true.

Kupferman and Vardi [10] show that the problem of determining whether a given string can be spelled by concatenating edge labels in an Eulerian trail in an input graph is NP-complete. We call this problem EULERIAN TRAIL EQUALING WORD. We show in Theorem 2 that we can reduce ETEW to GTED, and therefore if GTED is polynomially solvable, then ETEW is polynomially solvable. The complete proof is in Appendix A.1.

▶ **Problem 2** (Eulerian Trail Equaling Word [10]). *Given a string $s \in \Sigma^*$, an edge-labaled Eulerian graph G, find an Eulerian trail t of G such that $str(t) = s$.*

▶ **Theorem 2.** *If $GTED \in P$ then $ETEW \in P$.*

Proof sketch. We first convert an input instance $\langle s, G \rangle$ of ETEW into an input instance $\langle G_1, G_2 \rangle$ of GTED by (a) creating graph G_1 that only contains edges that reconstruct string s and (b) modifying G into G_2 by extending the anti-parallel edges so that G_2 is unidirectional. We show that if $GTED(G_1, G_2) = 0$, there must be an Eulerian trail in G that spells s, and if $GTED(G_1, G_2) > 0$, G must not contain an Eulerian trail that spells s.   ◀

Hence, an (assumed) polynomial-time algorithm for GTED solves ETEW in polynomial time. This contradicts Theorem 6 of Kupferman and Vardi [10] of the NP-completeness of ETEW (under $P \neq NP$).

2.2   Reduction from Hamiltonian Path to GTED and FGTED

We resolve the contradiction by showing that GTED is NP-complete. The details of the proof are in Appendix A.2.

▶ **Theorem 3.** *GTED is NP-complete.*

Proof sketch. We reduce from the HAMILTONIAN PATH problem, which asks whether a directed, simple graph G contains a path that visits every vertex exactly once. Here, simple means no self-loops or parallel edges.

Let $\langle G = (V, E) \rangle$ be an instance of HAMILTONIAN PATH, with $n = |V|$ vertices. We first create the Eulerian closure of G, which is defined as $G' = (V', E')$ where

$$V' = \{v^{in}, v^{out} : v \in V\} \cup \{w\}. \tag{2}$$

Here, each vertex in V is split into v^{in} and v^{out}, and w is a newly added vertex. E' is the union of the following sets of edges and their labels:
- $E_1 = \{(v^{in}, v^{out}) : v \in V\}$, labeled `a`,
- $E_2 = \{(u^{out}, v^{in}) : (u, v) \in E\}$, labeled `b`,
- $E_3 = \{(v^{out}, v^{in}) : v \in V\}$, labeled `c`,
- $E_4 = \{(v^{in}, u^{out}) : (u, v) \in E\}$, labeled `c`,
- $E_5 = \{(u^{in}, w) : u \in V\}$, labeled `c`,
- $E_6 = \{(w, u^{in}) : u \in V\}$, labeled `b`.

G' is an Eulerian graph by construction but contains anti-parallel edges. We further create G'' from G' by adding dummy nodes so that each pair of antiparallel edges is split into two parallel, length-2 paths with labels `x#`, where `x` is the original label.

We also create a graph C that has the same number of edges as G'' and spells out a string

$$q = \texttt{a\#}(\texttt{b\#a\#})^{n-1}(\texttt{c\#})^{2n-1}(\texttt{c\#b\#})^{|E|+1}. \tag{3}$$

We then argue that G has a Hamiltonian path if and only if G'' spells out the string q, which uses the same line of arguments and graph traversals as in Kupferman and Vardi [10]. We then show that $GTED(G'', C) = 0$ if and only if G'' spells q.   ◀

Following a similar argument, we show that FGTED is also NP-complete, and its proof is in Appendix A.3.

▶ **Theorem 4.** *FGTED is NP-complete.*
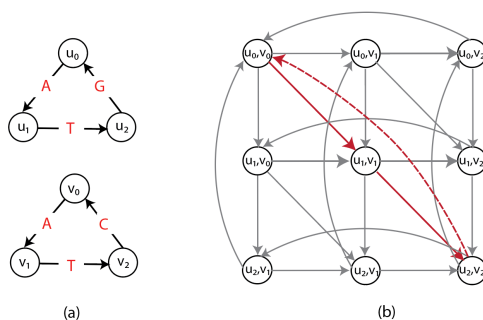
Figure 1 (a) Examples of two edge-labeled Eulerian graphs G_1 (top) and G_2 (bottom). (b) The alignment graph $\mathcal{A}(G_1, G_2)$. The cycle with red edges is the path corresponding to $\text{GTED}(G_1, G_2)$. Red solid edges are matches with cost 0 and red dashed-line edge is mismatch with cost 1.

3 Revisiting the correctness of the proposed ILP solutions to GTED

We revisit two proposed ILP solutions to GTED by Ebrahimpour Boroojeny et al. [6] and show that the optimal solution to these ILP is not always equal to GTED.

3.1 Alignment graph

The previously proposed ILP formulations for GTED are based on the alignment graph constructed from input graphs. The high-level concept of an alignment graph is similar to the dynamic programming matrix for the string-to-string alignment problem [17].

▶ **Definition 5** (Alignment graph). *Let G_1, G_2 be two unidirectional, edge-labeled Eulerian graphs. The* alignment graph $\mathcal{A}(G_1, G_2) = (V, E, \delta)$ *is a directed graph that has vertex set $V = V_1 \times V_2$ and edge multi-set E that equals the union of the following:*
Vertical edges $[(u_1, u_2), (v_1, u_2)]$ *for $(u_1, v_1) \in E_1$ and $u_2 \in V_2$,*
Horizontal edges $[(u_1, u_2), (u_1, v_2)]$ *for $u_1 \in V_1$ and $(u_2, v_2) \in E_2$,*
Diagonal edges $[(u_1, u_2), (v_1, v_2)]$ *for $(u_1, v_1) \in E_1$ and $(u_2, v_2) \in E_2$.*
Each edge is associated with a cost by the cost function $\delta : E \to \mathbb{R}$.

Each diagonal edge $e = [(u_1, u_2), (v_1, v_2)]$ in an alignment graph can be projected to (u_1, v_1) and (u_2, v_2) in G_1 and G_2, respectively. Similarly, each vertical edge can be projected to one edge in G_1, and each horizontal edge can be projected to one edge in G_2.

We define the edge projection function π_i that projects an edge from the alignment graph to an edge in the input graph G_i. If the alignment edge is a vertical or horizontal edge, it is projected to one edge in only one input graph. We also define the path projection function Π_i that projects a trail in the alignment graph to a trail in the input graph G_i. For example, let a trail in the alignment graph be $p = (e_1, e_2, \ldots, e_m)$, and $\Pi_i(p) = (\pi_i(e_1), \pi_i(e_2), \ldots, \pi_i(e_m))$ is a trail in G_i.

An example of an alignment graph is shown in Figure 1(b). The horizontal edges correspond to gaps in strings represented by G_1, vertical edges correspond to gaps in strings represented by G_2, and diagonal edges correspond to the matching between edge labels from the two graphs. In the rest of this paper, we assume that the costs for horizontal and vertical edges are 1, and the costs for the diagonal edges are 1 if the diagonal edge represents a mismatch and 0 if it is a match. The cost function δ can be defined to capture the cost of matching between edge labels or inserting gaps. This definition of alignment graph is also a generalization of the alignment graph used in string-to-graph alignment [9].

3.2   The first previously proposed ILP for GTED

Lemma 1 in Ebrahimpour Boroojeny et al. [6] provides a model for computing GTED by finding the minimum-cost trail in the alignment graph. We reiterate it here for completeness.

▶ **Lemma 6** ([6]). *For any two edge-labeled Eulerian graphs G_1 and G_2,*

$$
\begin{aligned}
GTED(G_1, G_2) = \text{minimize}_c \quad & \delta(c) \\
\text{subject to} \quad & c \text{ is a trail in } \mathcal{A}(G_1, G_2), \\
& \Pi_i(c) \text{ is an Eulerian trail in } G_i \text{ for } i = 1, 2,
\end{aligned}
\tag{4}
$$

where $\delta(c)$ is the total edge cost of c, and $\Pi_i(c)$ is the projection from c to G_i.

An example of such a minimum-cost trail is shown in Figure 1(b). Ebrahimpour Boroojeny et al. [6] provide the following ILP formulation and claim that it is a direct translation of Lemma 6:

$$
\underset{x \in \mathbb{N}^{|E|}}{\text{minimize}} \quad \sum_{e \in E} x_e \delta(e)
\tag{5}
$$

$$
\text{subject to} \quad Ax = 0
\tag{6}
$$

$$
\sum_{e \in E} x_e I_i(e, f) = 1 \quad \text{for } i = 1, 2 \text{ and for all } f \in E_i,
\tag{7}
$$

where

$$
A_{ue} = \begin{cases} -1 & \text{if } e = (u, v) \in E \text{ for some vertex } v \in V \\ 1 & \text{if } e = (v, u) \in E \text{ for some } u \in V \\ 0 & \text{otherwise.} \end{cases}
\tag{8}
$$

Here, E is the edge set of $\mathcal{A}(G_1, G_2)$. A is the negative incidence matrix of size $|V| \times |E|$, and $I_i(e, f)$ is an indicator function that is 1 if edge e in E projects to edge f in the input graph G_i (and 0 otherwise). We define the domain of each x_e to include all non-negative integers. However, due to constraints (7), the values of x_e are limited to either 0 or 1. We describe this ILP formulation with the assumption that both input graphs have closed Eulerian trails, which means that each node has equal numbers of incoming and outgoing edges. We discuss the cases when input graphs contain open Eulerian trails in Section 4.

The ILP in (5)–(8) allows the solutions to select disjoint cycles in the alignment graph, and the projection of edges in these disjoint cycles need not correspond to a single string represented by either of the input graphs. We show that the ILP in (5)–(8) does not solve GTED by giving an example where the objective value of the optimal solution to the ILP in (5)–(8) is not equal to GTED.

Construct two input graphs as shown in Figure 2(a). Specifically, G_1 spells circular permutations of `TTTGAA` and G_2 spells circular permutations of `TTTAGA`. It is clear that $GTED(G_1, G_2) = 2$ under Levenshtein edit distance. On the other hand, as shown in Figure 2(a), an optimal solution in $\mathcal{A}(G_1, G_2)$ contains two disjoint cycles with nonzero x_e values that have a total edge cost equal to 0. This solution is a feasible solution to the ILP in (5)–(8). It is also an optimal solution because the objective value is zero, which is the lower bound on the ILP in (5)–(8). This optimal objective value, however, is smaller than $GTED(G_1, G_2)$. Therefore, the ILP in (5)–(8) does not solve GTED since it allows the solution to be a set of disjoint components.
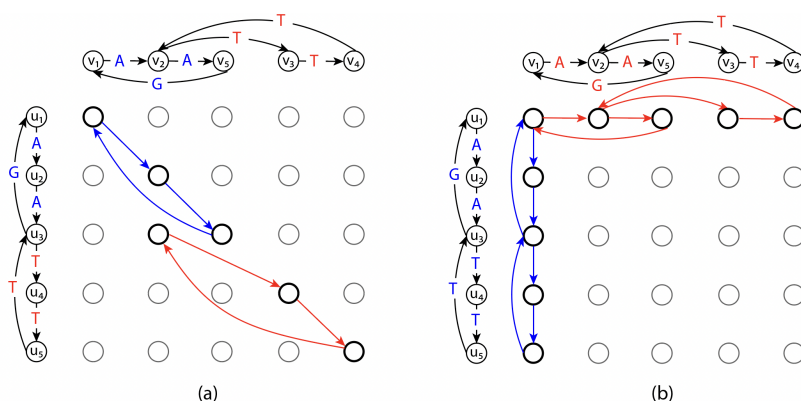
Figure 2 (a) The subgraph in the alignment graph induced by an optimal solution to the ILP in (5)–(8) and the ILP in (11)–(12) with input graphs on the left and top. The red and blue edges in the alignment graph are edges matching labels in red and blue font, respectively, and are part of the optimal solution to the ILP in (5)–(8). The cost of the red and blue edges are zero. (b) The subgraph induced by x^{init} with $s_1 = u_1$ and $s_2 = v_1$ according to the ILP in (11)–(12). The rest of the edges in the alignment graph are omitted for simplicity.
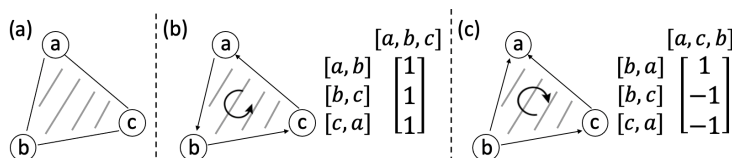
Figure 3 (a) A graph that contains an unoriented 2-simplex with three unoriented 1-simplices. (b), (c) The same graph with two different ways of orienting the simplices and the corresponding boundary matrices.

3.3 The second previously proposed ILP formulation of GTED

We describe the second proposed ILP formulation of GTED by Ebrahimpour Boroojeny et al. [6]. Following Ebrahimpour Boroojeny et al. [6], we use simplices, a notion from geometry, to generalize the notion of an edge to higher dimensions. A k-simplex is a k-dimensional polytope which is the convex hull of its $k + 1$ vertices. For example, a 1-simplex is an undirected edge, and a 2-simplex is a triangle. We use the orientation of a simplex, which is given by the ordering of the vertex set of a simplex up to an even permutation, to generalize the notion of the edge direction [16, p. 26]. We use square brackets $[\cdot]$ to denote an oriented simplex. For example, $[v_0, v_1]$ denotes a 1-simplex with orientation $v_0 \rightarrow v_1$, which is a directed edge from v_0 to v_1, and $[v_0, v_1, v_2]$ denotes a 2-simplex with orientation corresponding to the vertex ordering $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_0$. Each k-simplex has two possible unique orientations, and we use the signed coefficient to connect their forms together, e.g. $[v_0, v_1] = -[v_1, v_0]$.

For each pair of graphs G_1 and G_2 and their alignment graph $\mathcal{A}(G_1, G_2)$, we define an oriented 2-simplex set $T(G_1, G_2)$ which is the union of:

- $[(u_1, u_2), (v_1, u_2), (v_1, v_2)]$ for all $(u_1, v_1) \in E_1$ and $(u_2, v_2) \in E_2$, or

- $[(u_1, u_2), (u_1, v_2), (v_1, v_2)]$ for all $(u_1, v_1) \in E_1$ and $(u_2, v_2) \in E_2$,

We use the boundary operator [16, p. 28], denoted by ∂, to map an oriented k-simplex to a sum of oriented $(k-1)$-simplices with signed coefficients.

$$\partial[v_0, v_1, \ldots, v_k] = \sum_{i=0}^{p} (-1)^i [v_0, \ldots, \hat{v}_i, \ldots, v_k], \tag{9}$$

where $\hat{v}_i$ denotes the vertex v_i is to be deleted. Intuitively, the boundary operator maps the oriented k-simplex to a sum of oriented $(k-1)$-simplices such that their vertices are in the k-simplex and their orientations are consistent with the orientation of the k-simplex. For example, when $k = 2$, we have:

$$\partial[v_0, v_1, v_2] = [v_1, v_2] - [v_0, v_2] + [v_0, v_1] = [v_1, v_2] + [v_2, v_0] + [v_0, v_1]. \tag{10}$$

We reiterate the second ILP formulation proposed in Ebrahimpour Boroojeny et al. [6]. Given an alignment graph $\mathcal{A}(G_1, G_2) = (V, E, \delta)$ and the oriented 2-simplex set $T(G_1, G_2)$,

$$\underset{x \in \mathbb{N}^{|E|}, y \in \mathbb{Z}^{|T(G_1, G_2)|}}{\text{minimize}} \quad \sum_{e \in E} x_e \delta(e) \tag{11}$$
$$\text{subject to} \quad x = x^{init} + [\partial]y$$

Entries in x and y correspond to 1-simplices and 2-simplices in E and $T(G_1, G_2)$, respectively. $[\partial]$ is a $|E| \times |T(G_1, G_2)|$ boundary matrix where each entry $[\partial]_{i,j}$ is the signed coefficient of the oriented 1-simplex (the directed edge) in E corresponding to x_i in the boundary of the oriented 2-simplex in $T(G_1, G_2)$ corresponding to y_j. The index i, j for each 1-simplex or 2-simplex is assigned based on an arbitrary ordering of the 1-simplices in E or the 2-simplices in $T(G_1, G_2)$. An example of the boundary matrix is shown in Figure 3. $\delta(e)$ is the cost of each edge. $x^{init} \in \mathbb{R}^{|E|}$ is a vector where each entry corresponds to a 1-simplex in E with $|E_1| + |E_2|$ nonzero entries that represent one Eulerian trail in each input graph. x^{init} is a feasible solution to the ILP. Let s_1 be the source of the Eulerian trail in G_1, and s_2 be the sink of the Eulerian trail in G_2. Each entry in x^{init} is defined by

$$x_e^{init} = \begin{cases} 1 & \text{if } e = [(u_1, s_2), (v_1, s_2)] \text{ or } e = [(s_1, u_2), (s_1, v_2)], \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

If the Eulerian trail is closed in G_i, s_i can be any vertex in V_i. An example of x^{init} is shown in Figure 2(b).

We provide a complete proof in the extended version of this paper [22] that the ILP in (5)–(8) is equivalent to the ILP in (11)–(12). Therefore, the example we provided in Section 3.2 is also an optimal solution to the ILP in (11)–(12) but not a solution to GTED. Thus, the ILP in (11)–(12) does not always solve GTED.

Ebrahimpour Boroojeny et al. [6] argue that the linear programming relaxation of the ILP in (11)–(12) always yields integer optimal solutions by claiming that the constraint matrix of the LP relaxation of the ILP in (11)–(12) is always totally unimodular, and therefore the ILP in (11)–(12) can be solved in polynomial time. We provide a counterexample where the ILP in (11)–(12) yields fractional optimal solutions with fractional variable values. Additionally, we show that the constraint matrix of the LP relaxation of the ILP in (11)–(12) is not totally unimodular given most non-trivial input graphs. The details of the proofs and the counterexample are in the extended version of this paper [22] (Appendix F).
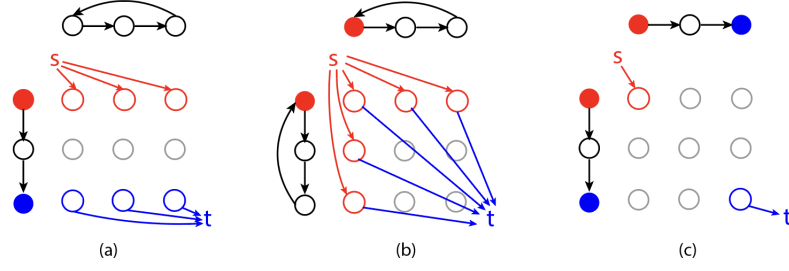
■ **Figure 4** Modified alignment graphs based on input types. (a) G_1 has open Eulerian trails while G_2 has closed Eulerian trails. (b) Both G_1 and G_2 have closed Eulerian trails. (c) Both G_1 and G_2 have open Eulerian trails. Solid red and blue nodes are the source and sink nodes of the graphs with open Eulerian trails. "s" and "t" are the added source and sink nodes. Colored edges are added alignment edges directing from and to source and sink nodes, respectively.

4 New ILP solutions to GTED

To ensure that our new ILP formulations are applicable to input graphs regardless of whether they contain an open or closed Eulerian trail, we add a source node s and a sink node t to the alignment graph. Figure 4 illustrates three possible cases of input graphs.

1. If only one of the input graphs has closed Eulerian trails, wlog, let G_1 be the input graph with open Eulerian trails. Let a_1 and b_1 be the start and end of the Eulerian trail that have odd degrees. Add edges $[s, (a_1, v_2)]$ and $[(b_1, v_2), t]$ to E for all nodes $v_2 \in V_2$ (Figure 4(a)).

2. If both input graphs have closed Eulerian trails, let a_1 and a_2 be two arbitrary nodes in G_1 and G_2, respectively. Add edges $[s, (a_1, v_2)]$, $[s, (v_1, a_2)]$, $[(a_1, v_2), t]$ and $[(v_1, a_2), t]$ for all nodes $v_1 \in V_1$ and $v_2 \in V_2$ to E (Figure 4(b)).

3. If both input graphs have open Eulerian trails, add edges $[s, (a_1, a_2)]$ and $[t, (b_1, b_2)]$, where a_i and b_i are start and end nodes of the Eulerian trails in G_i, respectively (Figure 4(c)).

According to Lemma 6, we can solve $\text{GTED}(G_1, G_2)$ by finding a trail in $\mathcal{A}(G_1, G_2)$ that satisfies the projection requirements. This is equivalent to finding a s-t trail in $\mathcal{A}(G_1, G_2)$ that satisfies constraints:

$$\sum_{(u,v) \in E} x_{uv} I_i((u,v), f) = 1 \quad \text{for all } (u,v) \in E, f \in G_i, \ u \neq s, \ v \neq t, \tag{13}$$

where $I_i(e, f) = 1$ if the alignment edge e projects to f in G_i, and x_{uv} is the ILP variable for edge $(u, v) \in E$. An optimal solution to GTED in the alignment graph must start and end with the source and sink node because they are connected to all possible starts and ends of Eulerian trails in the input graphs.

Since a trail in $\mathcal{A}(G_1, G_2)$ is a flow network, we use the following flow constraints to enforce the equality between the number of in- and out-edges for each node in the alignment graph except the source and sink nodes.

$$\sum_{(s,u)\in E} x_{su} = 1 \tag{14}$$

$$\sum_{(v,t)\in E} x_{vt} = 1 \tag{15}$$

$$\sum_{(u,v)\in E} x_{uv} = \sum_{(v,w)\in E} x_{vw} \quad \text{for all } v \in V \tag{16}$$

Constraints (13) and (16) are equivalent to constraints (7) and (6), respectively. Therefore, we rewrite the ILP in (5)–(8) in terms of the modified alignment graph.

$$\underset{x\in\mathbb{N}^{|E|}}{\text{minimize}} \quad \sum_{e\in E} x_e \delta(e) \qquad\qquad \text{(lower bound ILP)}$$

subject to   constraints (13)–(16).

As we show in Section 3.2, constraints (13)-(16) do not guarantee that the ILP solution is one trail in $\mathcal{A}(G_1, G_2)$, thus allowing several disjoint covering trails to be selected in the solution and fail to model GTED correctly. We show in Section 5 that the solution to this ILP is a lower bound to GTED.

According to Lemma 1 in Dias et al. [5], a subgraph of a directed graph G with source node s and sink node t is a s-t trail if and only if it is a flow network and every strongly connected component (SCC) of the subgraph has at least one edge outgoing from it. Thus, in order to formulate an ILP for the GTED problem, it is necessary to devise constraints that prevent disjoint SCCs from being selected in the alignment graph. In the following, we describe two approaches for achieving this.

4.1   Enforcing one trail in the alignment graph via constraint generation

Section 3.2 of Dias et al. [5] proposes a method to design linear constraints for eliminating disjoint SCCs, which can be directly adapted to our problem. Let $\mathcal{C}$ be the collection of all strongly connected subgraphs of the alignment graph $\mathcal{A}(G_1, G_2)$. We use the following constraint to enforce that the selected edges form one s-t trail in the alignment graph:

$$\text{If} \sum_{(u,v)\in E(C)} x_{uv} = |E(C)|, \text{ then} \sum_{(u,v)\in\varepsilon^+(C)} x_{uv} \geq 1 \quad \text{for all } C \in \mathcal{C}, \tag{17}$$

where $E(C)$ is the set of edges in the strongly connected subgraph C and $\varepsilon^+(C)$ is the set of edges (u,v) such that u belongs to C and v does not belong to C. $\sum_{(u,v)\in E(C)} x_{uv} = |E(C)|$ indicates that C is in the subgraph of $\mathcal{A}(G_1, G_2)$ constructed by all edges (u,v) with positive x_{uv}, and $\sum_{(u,v)\in\varepsilon^+(C)} x_{uv} \geq 1$ guarantees that there exists an out-going edge of C that is in the subgraph.

We use the same technique as Dias et al. [5] to linearize the "if-then" condition in (17) by introducing a new variable β for each strongly connected component:

$$\sum_{(u,v)\in E(C)} x_{uv} \geq |E(C)|\beta_C \quad \text{for all } C \in \mathcal{C} \tag{18}$$

$$\sum_{(u,v)\in E(C)} x_{uv} - |E(C)| + 1 - |E(C)|\beta_C \leq 0 \quad \text{for all } C \in \mathcal{C} \tag{19}$$

$$\sum_{(u,v)\in\varepsilon^+(C)} x_{uv} \geq \beta_C \quad \text{for all } C \in \mathcal{C} \tag{20}$$

$$\beta_C \in \{0,1\} \quad \text{for all } C \in \mathcal{C} \tag{21}$$

To summarize, given any pair of unidirectional, edge-labeled Eulerian graphs G_1 and G_2 and their alignment graph $\mathcal{A}(G_1, G_2) = (V, E, \delta)$, $\text{GTED}(G_1, G_2)$ is equal to the optimal solution of the following ILP formulation:

$$\underset{x \in \{0,1\}^{|E|}}{\text{minimize}} \quad \sum_{e \in E} x_e \delta(e)$$

subject to   constraints (13)–(16) and   (exponential ILP)

constraints (18)–(21).

This ILP has an exponential number of constraints as there is a set of constraints for every strongly connected subgraph in the alignment graph. To solve this ILP more efficiently, we can use the procedure similar to the iterative constraint generation procedure in Dias et al. [5]. Initially, solve the ILP with only constraints (13)-(16). Create a subgraph, G', induced by edges with positive x_{uv}. For each disjoint SCC in G' that does not contain the sink node, add constraints (18)-(21) for edges in the SCC and solve the new ILP. Iterate until no disjoint SCCs are found in the solution. Algorithm 1 is the pseudo-code of this procedure, which is similar to Algorithm 1 of Dias et al. [5].

Algorithm 1 Iterative constraint generation algorithm to solve (exponential ILP).

1: **Input** Two unidirectional, edge-labeled Eulerian graphs and their alignment graph
2: $\mathcal{C} \leftarrow \emptyset$
3: **while** true **do**
4:     Solve the ILP (exponential ILP) with $\mathcal{C}$
5:     **if** the ILP variables x_{uv} induce a strongly connected component C not satisfying (17) **then**
6:         $\mathcal{C} = \mathcal{C} \cup \{C\}$
7:     **else**
8:         **return** the optimal ILP value and the corresponding optimal solution x
9:     **end if**
10: **end while**

4.2 A compact ILP for GTED with polynomial number of constraints

In the worst cases, the number of iterations to solve (exponential ILP) via constraint generation is exponential. As an alternative, we introduce a compact ILP with only a polynomial number of constraints. The intuition behind this ILP is that we can impose a partially increasing ordering on all the edges so that the selected edges forms a *s-t* trail in the alignment graph. This idea is similar to the Miller-Tucker-Zemlin ILP formulation of the Travelling Salesman problem (TSP) [14].

We add variables d_{uv} that are constrained to provide a partial ordering of the edges in the *s-t* trail and set the variables d_{uv} to zero for edges that are not selected in the *s-t* trail. Intuitively, there must exist an ordering of edges in a *s-t* trail such that for each pair of consecutive edges (u, v) and (v, w), the difference in their order variable d_{uv} and d_{vw} is 1. Therefore, for each node v that is not the source or the sink, if we sum up the order variables for the incoming edges and outgoing edges respectively, the difference between the two sums is equal to the number of selected incoming/outgoing edges. Lastly, the order variable for the edge starting at source is 1, and the order variable for the edge ending at sink is the number of selected edges. This gives the ordering constraints as follows:

$$\text{If } x_{uv} = 0, \text{ then } d_{uv} = 0 \quad \text{for all } (u,v) \in E \tag{22}$$

$$\sum_{(v,w) \in E} d_{vw} - \sum_{(u,v) \in E} d_{uv} = \sum_{(v,w) \in E} x_{vw} \quad \text{for all } v \in V \setminus \{s,t\} \tag{23}$$

$$\sum_{(s,u) \in E} d_{su} = 1 \tag{24}$$

$$\sum_{(v,t) \in E} d_{vt} = \sum_{(u,v) \in E} x_{uv} \tag{25}$$

We enforce that all variables $x_e \in \{0,1\}$ and $d_e \in \mathbb{N}$ for all $e \in E$.

The "if-then" statement in Equation (22) can be linearized by introducing an additional binary variable y_{uv} for each edge [2, 5]:

$$-x_{uv} - |E|y_{uv} \leq -1 \tag{26}$$

$$d_{uv} - |E|(1 - y_{uv}) \leq 0 \tag{27}$$

$$y_{uv} \in \{0,1\}. \tag{28}$$

Here, y_{uv} is an indicator of whether $x_{uv} \geq 0$. The coefficient $|E|$ is the number of edges in the alignment graph and also an upper bound on the ordering variables. When $y_{uv} = 1$, $d_{uv} \leq 0$, and y_{uv} does not impose constraints on x_{uv}. When $y_{uv} = 0$, $x_{uv} \geq 1$, and y_{uv} does not impose constraints on d_{uv}. As we show in Lemma 7 , these constraints prevent finding disjoint components, thus guaranteeing the correctness of the ILP.

▶ **Lemma 7.** *Let x_e and d_e be ILP variables. Let G' be a subgraph of $\mathcal{A}(G_1, G_2)$ that is induced by edges with $x_e = 1$. If x_e and d_e satisfy constraints (13)-(25) for all $e \in E$, G' is connected with one trail from s to t that traverses each edge in G' exactly once.*

Proof. We prove the lemma in 2 parts: (1) all nodes except s and t in G' have an equal number of in- and out-edges, (2) G' contains only one connected component.

The first statement holds because the edges of G' form a flow from s to t, and is enforced by constraints (16).

We then show that G' does not contain isolated subgraphs that are not reachable from s or t. Due to constraint (16), the only possible scenario is that the isolated subgraph is strongly connected. Suppose for contradiction that there is a strongly connected component, C, in G' that is not reachable from s or t.

The sum of the left hand side of constraint (23) over all vertices in C is

$$\sum_{v \in C} \left(\sum_{(u,v) \in C} d_{uv} - \sum_{(v,w) \in C} d_{vw} \right) = \sum_{v \in C} \sum_{(u,v) \in C} d_{uv} - \sum_{v \in C} \sum_{(v,w) \in C} d_{vw} \tag{29}$$

$$= \sum_{(u,v) \in E(C)} d_{uv} - \sum_{(v,w) \in E(C)} d_{vw} = 0 \tag{30}$$

However, the right-hand side of the same constraints is always positive. Hence, we have a contradiction. Therefore, G' has only one connected component.   ◀

Due to Lemma 6 and Lemma 7, given input graphs G_1 and G_2 and the alignment graph $\mathcal{A}(G_1, G_2)$, $\text{GTED}(G_1, G_2)$ is equal to the optimal objective of

$$
\begin{aligned}
&\underset{x \in \{0,1\}^{|E|}}{\text{minimize}} && \sum_{e \in E} x_e \delta(e) \\
&\text{subject to} && \text{constraints (13)–(16),} \\
& && \text{constraints (23)–(25)} \\
& && \text{and constraints (26)–(28).}
\end{aligned}
\qquad \text{(compact ILP)}
$$

5 Closed-trail Cover Traversal Edit Distance

While the (lower bound ILP) and the ILP in (11)–(12) do not solve GTED, the optimal solution to these ILPs is a lower bound of GTED. These ILP formulations also solve an interesting variant of GTED, which is a local similarity measure between two genome graphs. We call this variant as Closed-trail Cover Traversal Edit Distance (CCTED). In the following, we provide the formal definition of CCTED problem, and then show that the (lower bound ILP) is the correct ILP formulation for solving CCTED.

We first introduce the min-cost item matching problem between two multi-sets. Let two multi-sets of items be S_1 and S_2, and, wlog, let $|S_1| \leq |S_2|$. Let $c : (S_1 \cup \{\epsilon\}) \times S_2 \to \mathbb{N}$ be the cost of matching either an empty item ϵ or an item in S_1 with an item in S_2. Given S_1, S_2 and the cost function c, min-cost matching problem finds a matching, $\mathcal{M}_c(S_1, S_2)$, such that each item in $S_1 \cup \{\epsilon\}^{|S_2| - |S_1|}$ is matched with exactly one distinct item in S_2 and the total cost of the matching, $\sum_{(s_1, s_2) \in \mathcal{M}_c(S_1, S_2)} c(s_1, s_2)$, is minimized.

The min-cost item matching problem is similar to the Earth Mover's Distance defined in [18], except that only integral units of items can be matched and the cost of matching an empty item with another item is not constant. Similar to the Earth Mover's Distance, the min-cost item matching problem can be computed using the ILP formulation of the min-cost max-flow problem [23, 21]. When the cost is the edit distance, the cost to match ϵ with a string is equal to the length of the string.

Define traversal edit distance, $edit_t(t_1, t_2)$ as the edit distance between the strings constructed from a pair of trails t_1 and t_2. In other words, $edit_t(t_1, t_2) = edit(\text{str}(t_1), \text{str}(t_2))$. CCTED is defined as:

▶ **Problem 3** (Closed-Trail Cover Traversal Edit Distance (CCTED)). *Given two unidirectional, edge-labeled Eulerian graphs G_1 and G_2 with closed Eulerian trails, compute*

$$
CCTED(G_1, G_2) \triangleq \min_{\substack{C_1 \in CC(G_1), \\ C_2 \in CC(G_2)}} \sum_{(t_1, t_2) \in \mathcal{M}_{edit_t}(C_1, C_2)} edit(str(t_1), str(t_2)), \qquad (31)
$$

Here, $CC(G)$ denotes the collection of all possible sets of edge-disjoint, closed trails in G, such that every edge in G belongs to exactly one of these trails. Each element of $CC(G)$ can be interpreted as a cover of G using such trails. $\mathcal{M}_{edit_t}(C_1, C_2)$ is a min-cost matching between two covers using the traversal edit distance as the cost.

CCTED is likely a more suitable metric comparisons between genomes that undergo large-scale rearrangements. This analogy is to the relationship between the synteny block comparison [20] and the string edit distance computation, where the former is more often used in interspecies comparisons and in detecting segmental duplications [1, 24] and the latter is more often seen in intraspecies comparisons.

Following similar ideas as Lemma 6, we can compute CCTED by finding a set of closed trails in the alignment graph such that the total cost of alignment edges are minimized, and the projection of all edges in the collection of selected trails is equal to the multi-set of input graph edges.

▶ **Lemma 8.** *For any two edge-labeled Eulerian graphs G_1 and G_2,*

$$CCTED(G_1, G_2) = \underset{C}{minimize} \quad \sum_{c \in C} \delta(c) \tag{32}$$

$$subject\ to \quad C \text{ is a set of closed trails in } \mathcal{A}(G_1, G_2),$$

$$\bigcup_{e \in C} \Pi_i(e) = E_i \quad for\ i = 1, 2, \tag{33}$$

where C is a collection of trails and $\delta(c)$ is the total cost of edges in trail c.

Proof. Given any pair of covers $C_1 \in CC(G_1)$ and $C_2 \in CC(G_2)$ and their min-cost matching based on the edit distance $\mathcal{M}_{\text{edit}_t}(C_1, C_2)$, we can project each pair of matched closed trails to a closed trail in the alignment graph. For a matching between a trail and the empty item ϵ, we can project it to a closed trail in the alignment graph with all vertical edges if the trail is from G_1 or horizontal edges if the trail is from G_2. The total cost of the projected edges must be greater than or equal to the objective (32). On the other hand, every collection of trails C that satisfy constraint (33) can be projected to a cover in each of the input graphs, and $\sum_{c \in C} \delta(c) \geq CCTED(G_1, G_2)$. Hence equality holds. ◀

We show that (lower bound ILP) solves CCTED (the proof is in Appendix B).

▶ **Theorem 9.** *Given two input graphs G_1 and G_2, the optimal objective value of* (lower bound ILP) *based on $\mathcal{A}(G_1, G_2)$ is equal to $CCTED(G_1, G_2)$.*

5.1 CCTED is a lower bound of GTED

Since the constraints for (lower bound ILP) are a subset of (exponential ILP), a feasible solution to (exponential ILP) is always a feasible solution to (lower bound ILP). Since two ILPs have the same objective function, $\text{CCTED}(G_1, G_2) \leq \text{GTED}(G_1, G_2)$ for any pair of graphs. Moreover, when the solution to (lower bound ILP) forms only one connected component, the optimal value of (lower bound ILP) is equal to GTED.

▶ **Theorem 10.** *Let $\mathcal{A}'(G_1, G_2)$ be the subgraph of $\mathcal{A}(G_1, G_2)$ induced by edges $(u, v) \in E$ with $x_{uv}^{opt} = 1$ in the optimal solution to* (lower bound ILP)*, where x_{uv} is the value of the variable corresponding to edge (u, v). There exists $\mathcal{A}'(G_1, G_2)$ that has exactly one connected component if and only if $CCTED(G_1, G_2) = GTED(G_1, G_2)$.*

Proof. We first show that if $\text{CCTED}(G_1, G_2) = \text{GTED}(G_1, G_2)$, then there exists $\mathcal{A}'(G_1, G_2)$ that has exactly one connected component. Since $\text{CCTED}(G_1, G_2) = \text{GTED}(G_1, G_2)$, an optimal solution to (exponential ILP) is also an optimal solution to (lower bound ILP), which can induce a subgraph in the alignment graph that only contains one connected component.

Conversely, if x^{opt} induces a subgraph in the alignment graph with only one connected component, it satisfies constraints (18)-(21) and therefore is feasible to the ILP for GTED (exponential ILP). Since $\text{CCTED}(G_1, G_2) \leq \text{GTED}(G_1, G_2)$, this solution must also be optimal for $\text{GTED}(G_1, G_2)$. ◀

In practice, we may estimate GTED by the solution to (lower bound ILP). As we show in Section 6, the time needed to solve (lower bound ILP) is much less than the time needed to solve GTED. When the subgraph induced by the solution to (lower bound ILP) has only one connected component, CCTED is exactly equal to GTED. However, in adversarial cases, CCTED could be zero but GTED could be arbitrarily large.

6   Empirical evaluation of the ILP formulations for GTED and its lower bound

6.1   Implementation of the ILP formulations

We implement the algorithms and ILP formulations for (exponential ILP), (compact ILP) and (lower bound ILP). In practice, the multi-set of edges of each input graph may contain many duplicates of edges that have the same start and end vertices due to repeats in the strings. We reduce the number of variables and constraints in the implemented ILPs by merging the edges that share the same start and end nodes and recording the multiplicity of each edge. Each x variable is no longer binary but a non-negative integer that satisfies the modified projection constraint (13):

$$\sum_{(u,v)\in E} x_{uv} I_i((u,v),f) = M_i(f) \quad \text{for all } (u,v) \in E,\ f \in G_i, u \neq s, v \neq t, \tag{34}$$

where $M_i(f)$ is the multiplicity of edge f in G_i. Let C be the strongly connected component in the subgraph induced by positive x_{uv}, now $\sum_{(u,v)\in E(C)} x_{uv}$ is no longer upper bounded by $|E(C)|$. Therefore, the constraint (19) is changed to

$$\sum_{(u,v)\in E(C)} x_{uv} - |E(C)| + 1 - W(C)\beta_C \leq 0 \quad \text{for all } C \in \mathcal{C}, \tag{35}$$

$$W(C) = \sum_{(u,v)\in E(C)} \max\left(\sum_{f\in G_1} M_1(f) I_1((u,v),f), \sum_{f\in G_2} M_2(f) I_2((u,v),f)\right),$$

where $W(C)$ is the maximum total multiplicities of edges in the strongly connected subgraph in each input graph that is projected from C.

Likewise, constraints (27) that set the upper bounds on the ordering variables also need to be modified as the upper bound of the ordering variable d_{uv} for each edge no longer represents the order of one edge but the sum of orders of copies of (u,v) that are selected, which is at most $|E|^2$. Therefore, constraint (27) is changed to

$$d_{uv} - |E|^2 (1 - y_{uv}) \leq 0. \tag{36}$$

The rest of the constraints remain unchanged.

We ran all our experiments on a server with 48 cores (96 threads) of Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz and 378 GB of memory. The system was running Ubuntu 18.04 with Linux kernel 4.15.0. We solve all the ILP formulations and their linear relaxations using the Gurobi solver [7] using 32 threads.

6.2   GTED on simulated TCR sequences

We construct 20 de Bruijn graphs with $k = 4$ using 150-character sequences extracted from the V genes from the IMGT database [11]. We solve (exponential ILP), (lower bound ILP) and its linear relaxation, and the linear relaxation of (compact ILP) on all 190 pairs of graphs. We do not show results for solving (compact ILP) for GTED on this set of graphs as the running time exceeds 30 minutes on most pairs of graphs.
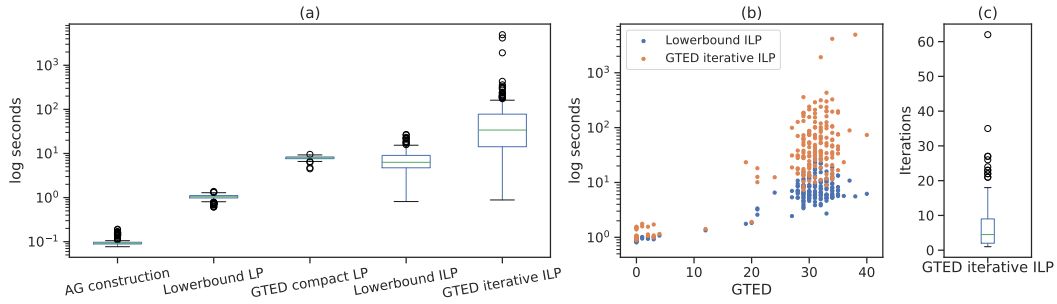
Figure 5 (a) The distribution of wall-clock running time for constructing alignment graphs, solving the ILP formulations for GTED and its lower bound, and their linear relaxations on the log scale. (b) The relationship between the time to solve (lower bound ILP), (exponential ILP) iteratively and GTED. (c) The distribution of the number of iterations to solve exponential ILP. The box plots in each plot show the median (middle line), the first and third quantiles (upper and lower boundaries of the box), the range of data within 1.5 inter-quantile range between Q1 and Q3 (whiskers), and the outlier data points.

To compare the time to solve the ILP formulations when GTED is equal to the optimal objective of (lower bound ILP), we only include 168 out of 190 graphs where GTED is equal to CCTED. On average, it takes 26 seconds wall-clock time to solve (lower bound ILP), and 71 seconds to solve (exponential ILP) using the iterative algorithm. On average, it takes 9 seconds to solve the LP relaxation of (compact ILP) and 1 second to solve the LP relaxation of (lower bound ILP). The time to construct the alignment graph for all pairs is less than 0.2 seconds. The distribution of wall-clock running time is shown in Figure 5(a). The time to solve (exponential ILP) and (lower bound ILP) is generally positively correlated with the GTED values (Figure 5(b)). On average, it takes 7 iterations for the iterative algorithm to find the optimal solution that induces one strongly connected subgraph (Figure 5(c)).

In summary, it is fastest to compute the lower bound of GTED. Computing GTED exactly by solving the proposed ILPs on genome graphs of size 150 is already time consuming. When the sizes of the genome graphs are fixed, the time to solve for GTED and its lower bound increases as the GTED between the two genome graphs increases. In the case where GTED is equal to its lower bound, the subgraph induced by some optimal solutions of (lower bound ILP) contains more than one strongly connected component. Therefore, in order to reconstruct the strings from each input graph that have the smallest edit distance, we generally need to obtain the optimal solution to the ILP for GTED. In all cases, the time to solve the (exponential ILP) is less than the time to solve the (compact ILP).

6.3   GTED on difficult cases

Repeats, such as segmental duplications and translocations [12, 4] in the genomes increase the complexity of genome comparisons. We simulate such structures with a class of graphs that contain n simple cycles of which $n-1$ peripheral cycles are attached to the n-th central cycle at either a node or a set of edges (Figure 6(a)). The input graphs in Figure 2 belong to this class of graphs that contain 2 cycles. This class of graphs simulates the complex structural variants in disease genomes or the differences between genomes of different species.

We generate pairs of 3-cycle graphs with varying sizes and randomly assign letters from {A,T,C,G} to edges. We compute the CCTED and GTED using (lower bound ILP) and (compact ILP), respectively. We group the generated 3-cycle graph pairs based on the
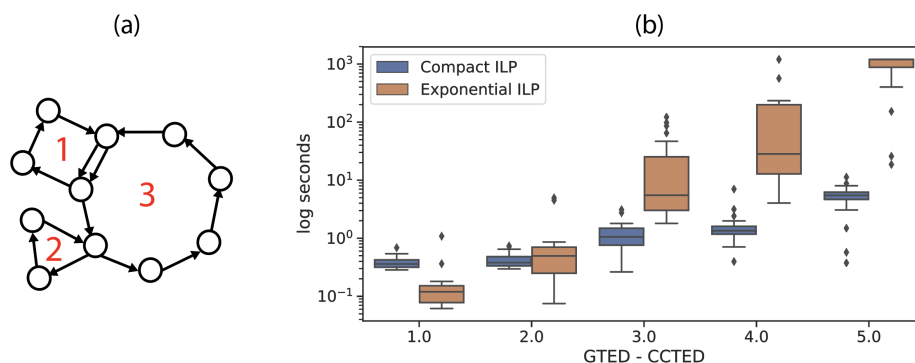
Figure 6 (a) An example of a 3-cycle graph. Cycle 1 and 2 are attached to cycle 3. (b) The distribution of wall-clock time to solve the compact ILP and the iterative exponential ILP on 100 pairs of 3-cycle graphs.

value of (GTED − CCTED) and select 20 pairs of graphs randomly for each (GTED − CCTED) value ranging from 1 to 5. The maximum number of edges in all selected graphs is 32.

We show the difficulty of computing GTED using the iterative algorithm on the 100 selected pairs of 3-cycle graphs. We terminate the ILP solver after 20 minutes. As shown in Figure 6, as the difference between GTED and CCTED increases, the wall-clock time to solve (exponential ILP) for GTED increases faster than the time to solve (compact ILP) for GTED. For pairs of graphs with (GTED − CCTED) = 5, on average it takes more than 15 minutes to solve (exponential ILP) with more than 500 iterations. On the other hand, it takes an average of 5 seconds to solve (compact ILP) for GTED and no more than 1 second to solve for the lower bound. The average time to solve each ILP is shown in Table S1.

In summary, on the class of 3-cycle graphs introduced above, the difficulty to solve GTED via the iterative algorithm increases rapidly as the gap between GTED and CCTED increases. Although (exponential ILP) is solved more quickly than (compact ILP) for GTED when the sequences are long and the GTED is equal to CCTED (Section 6.2), (compact ILP) may be more efficient when the graphs contain overlapping cycles such that the gap between GTED and CCTED is larger.

7 Conclusion

We point out the contradictions in the result on the complexity of labeled graph comparison problems and resolve the contradictions by showing that GTED, as opposed to the results in Ebrahimpour Boroojeny et al. [6], is NP-complete. On one hand, this makes GTED a less attractive measure for comparing graphs since it is unlikely that there is an efficient algorithm to compute the measure. On the other hand, this result better explains the difficulty of finding a truly efficient algorithm for computing GTED exactly. In addition, we show that the previously proposed ILP of GTED [6] does not solve GTED and give two new ILP formulations of GTED.

We further show that the previously proposed ILP solves for a lower bound of GTED. We characterize the LP relaxation of the ILP in (11)–(12) and show that, contrary to the results in Ebrahimpour Boroojeny et al. [6], the LP in (11)–(12) does not always yield optimal integer solutions.

As shown previously [6, 21], it takes more than 4 hours to solve (lower bound ILP) for graphs that represent viral genomes that contain ≈ 3000 bases with a multi-threaded LP solver. Likewise, we show that computing GTED using either (exponential ILP) or (compact ILP) is already slow on small genomes, especially on pairs of genomes that are different due to segmental duplications and translations. The empirical results show that it is currently impossible to solve GTED or its lower bound directly using this approach for bacterial- or eukaryotic-sized genomes on modern hardware. The results here should increase the theoretical interest in GTED along the directions of heuristics or approximation algorithms as justified by the NP-hardness of finding GTED.

References

1 Guillaume Bourque, Pavel A Pevzner, and Glenn Tesler. Reconstructing the genomic architecture of ancestral mammals: lessons from human, mouse, and rat genomes. *Genome Research*, 14(4):507–516, 2004.

2 Stephen P Bradley, Arnoldo C Hax, and Thomas L Magnanti. *Applied mathematical programming.* Addison-Wesley, 1977.

3 George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

4 Eva Darai-Ramqvist, Agneta Sandlund, Stefan Müller, George Klein, Stefan Imreh, and Maria Kost-Alimova. Segmental duplications and evolutionary plasticity at tumor chromosome break-prone regions. *Genome Research*, 18(3):370–379, 2008.

5 Fernando HC Dias, Lucia Williams, Brendan Mumey, and Alexandru I Tomescu. Minimum flow decomposition in graphs with cycles using integer linear programming. *arXiv preprint*, 2022. `arXiv:2209.00042`.

6 Ali Ebrahimpour Boroojeny, Akash Shrestha, Ali Sharifi-Zarchi, Suzanne Renick Gallagher, S. Cenk Sahinalp, and Hamidreza Chitsaz. Graph traversal edit distance and extensions. *Journal of Computational Biology*, 27(3):317–329, 2020.

7 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL: `https://www.gurobi.com`.

8 Steve Huntsman and Arman Rezaee. De Bruijn entropy and string similarity. *arXiv preprint*, 2015. `arXiv:1509.02975`.

9 Chirag Jain, Haowen Zhang, Yu Gao, and Srinivas Aluru. On the complexity of sequence-to-graph alignment. *Journal of Computational Biology*, 27(4):640–654, 2020.

10 Orna Kupferman and Gal Vardi. Eulerian paths with regular constraints. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

11 Marie-Paule Lefranc. IMGT, the international ImMunoGeneTics information system. *Cold Spring Harbor Protocols*, 2011(6):595–603, 2011.

12 Yilong Li, Nicola D Roberts, Jeremiah A Wala, Ofer Shapira, Steven E Schumacher, Kiran Kumar, Ekta Khurana, Sebastian Waszak, Jan O Korbel, James E Haber, et al. Patterns of somatic structural variation in human cancer genomes. *Nature*, 578(7793):112–121, 2020.

13 Serghei Mangul and David Koslicki. Reference-free comparison of microbial communities via de Bruijn graphs. In *Proceedings of the 7th ACM international conference on bioinformatics, computational biology, and health informatics*, pages 68–77, 2016.

14 Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

15 Ilia Minkin and Paul Medvedev. Scalable pairwise whole-genome homology mapping of long genomes with Bubbz. *IScience*, 23(6):101224, 2020.

16 James R Munkres. *Elements of algebraic topology.* CRC Press, 2018.

17 Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

18 Ofir Pele and Michael Werman. A linear time histogram metric for improved sift matching. In *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III 10*, pages 495–508. Springer, 2008.

19 Pavel A Pevzner, Haixu Tang, and Michael S Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences of USA*, 98(17):9748–9753, 2001.

20 Evgeny Polevikov and Mikhail Kolmogorov. Synteny Paths for Assembly Graphs Comparison. In Katharina T. Huber and Dan Gusfield, editors, *19th International Workshop on Algorithms in Bioinformatics (WABI 2019)*, volume 143 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.WABI.2019.24`.

21 Yutong Qiu and Carl Kingsford. The effect of genome graph expressiveness on the discrepancy between genome graph distance and string set distance. *Bioinformatics*, 38:i404–i412, 2022.

22 Yutong Qiu, Yihang Shen, and Carl Kingsford. Revisiting the complexity of and algorithms for the Graph Traversal Edit Distance and Its variants. *arXiv preprint*, 2023. `arXiv:2305.10577`.

23 Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The Earth Mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

24 Mitchell R Vollger, Xavi Guitart, Philip C Dishuck, Ludovica Mercuri, William T Harvey, Ariel Gershman, Mark Diekhans, Arvis Sulovari, Katherine M Munson, Alexandra P Lewis, et al. Segmental duplications and their variation in a complete human genome. *Science*, 376(6588):eabj6965, 2022.

A Proofs for the NP-completeness of GTED

A.1 Reduction from ETEW to GTED

We provide below the complete proof for Theorem 2.

▶ **Theorem 2.** *If $GTED \in P$ then $ETEW \in P$.*

Proof. Let $\langle s, G \rangle$ be an instance of ETEW. Construct a directed, acyclic graph (DAG), C, that has only one path. Let the path in C be $P = (e_1, \ldots, e_{|s|})$ and the edge label of e_i be $s[i]$. Clearly, C is a unidirectional, edge-labeled Eulerian graph, P is the only Eulerian trail in C, and $str(P) = s$.

For the graph $G = (V_G, E_G, \ell_G, \Sigma)$ from the ETEW instance, which may not be unidirectional, create another graph G' that contains all of the nodes and edges in G except the anti-parallel edges. Let $\Sigma_{G'} = \Sigma \cup \{\epsilon\}$, where ϵ is a character that is not in Σ. For each pair of anti-parallel edges (u, v) and (v, u) in G, add four edges $(u, w_1), (w_1, v), (v, w_2), (w_2, u)$ by introducing new vertices w_1, w_2 to G'. Let $\ell_{G'}(u, w_1) = \ell_G(u, v)$ and $\ell_{G'}(w_2, u) = \ell_G(v, u)$. Let $\ell_{G'}(w_1, v) = \ell_{G'}(v, w_2) = \epsilon$ for every newly introduced vertex. G' has at most twice the number of edges as G and is Eulerian and unidirectional.

Define the cost of changing a character from a to b cost(a, b) for $a, b \in \Sigma \cup \{-\}$ to be 0 if $a = b$ and 1 otherwise. "$-$" is the gap character indicating an insertion or a deletion. Define cost(a, ϵ) with $a \in \Sigma$ to be 1. Define cost$(-, \epsilon)$ to be 0.

Use the (assumed) polynomial-time algorithm for GTED to ask whether GTED$(C, G') \leq$ 0 under edit distance Σ. If yes, then let (s_1, s_2) be the 0-cost alignment of the strings spelled out by the trails in C and G', respectively. The non-gap characters of s_1 must spell out s since there is only one Eulerian trail in C. Because the alignment cost is 0, any $-$ (gap)

characters in s_1 must be aligned with ϵ characters in s_2 and any non-gap characters in s_1 must be aligned to the same character in s_2. The trail in G' that spells s_2 can be transformed to a trail that spells s_3 by collapsing the edges with ϵ character labels, and $s_3 = s_1$.

If $\text{GTED}(C, G') > 0$, G must not contain an Eulerian trail that spells s. Otherwise, such a trail could be extended to a trail introducing some ϵ characters that could be aligned to s with zero cost by aligning gaps with ϵ characters.

Hence, an (assumed) polynomial-time algorithm for GTED solves ETEW in polynomial time. ◄

A.2   Reduction from Hamiltonian Path to GTED

We provide below the complete proof for Theorem 3.

▶ **Theorem 3.** *GTED is NP-complete.*

Proof. We reduce from the HAMILTONIAN PATH problem, which asks whether a directed, simple graph G contains a path that visits every vertex exactly once. Here simple means no self-loops or parallel edges. Let $\langle G = (V, E) \rangle$ be an instance of HAMILTONIAN PATH, with $n = |V|$ vertices. The reduction is almost identical to that presented in Kupferman and Vardi [10], and from here until noted later in the proof the argument is identical except for the technicalities introduced to force unidirectionality (and another minor change described later). The first step is to construct the Eulerian closure of G, which is defined as $G' = (V', E')$ where

$$V' = \{v^{in}, v^{out} : v \in V\} \cup \{w\}, \tag{37}$$

and E' is the union of the following sets of edges and their labels:

- $E_1 = \{(v^{in}, v^{out}) : v \in V\}$, labeled a,
- $E_2 = \{(u^{out}, v^{in}) : (u, v) \in E\}$, labeled b,
- $E_3 = \{(v^{out}, v^{in}) : v \in V\}$, labeled c,
- $E_4 = \{(v^{in}, u^{out}) : (u, v) \in E\}$, labeled c,
- $E_5 = \{(u^{in}, w) : u \in V\}$, labeled c,
- $E_6 = \{(w, u^{in}) : u \in V\}$, labeled b.

Since G' is connected and every outgoing edge in G' has a corresponding antiparallel incoming edge, G' is Eulerian. It is not unidirectional, so we further create G'' from G' by adding dummy nodes to each pair of antiparallel edges and labelling the length-2 paths so created with x#, where x is the original label of the split edge (a, b, or c) and # is some new symbol (shared between all the new edges). We call these length-2 paths introduced to achieve unidirectionality "split edges".

We now argue that G has a Hamiltonian path iff G'' has an Eulerian trail that spells out

$$q = \text{a\#}(\text{b\#a\#})^{n-1}(\text{c\#})^{2n-1}(\text{c\#b\#})^{|E|+1}. \tag{38}$$

If such an Eulerian trail exists, then the trail starts with spelling the string $\text{a\#}(\text{b\#a\#})^{n-1}$, which corresponds to a Hamiltonian trail in G since it visits exactly n "vertex split edges" (type E_1, labeled a#) and each vertex split edge can be used only once (since it is an Eulerian trail). Further, successively visited vertices must be connected by an edge in G since those are the only b# split edges in G'' (except those leaving w, but w must not be involved in spelling out $\text{a\#}(\text{b\#a\#})^{n-1}$, since entering w requires using a split edge labeled c#).

For the other direction, if a G has a Hamiltonian path $v_1, \ldots, v_n$, then walking that sequence of vertices in G'' will spell out $\text{a\#}(\text{b\#a\#})^{n-1}$. This path will cover all E_1 edges and the E_2 edges that are on the Hamiltonian path. Retracing the path so far in reverse will

use $2n-1$ split edges labeled `c#`, consuming the $(\texttt{c\#})^{2n-1}$ term in q and covering all nodes'
reverse vertex edges E_3 (since the path is Hamiltonian). The reverse path also covers the
E_4 edges corresponding to reverse Hamiltonian path edges. Our Eulerian trail is now "at"
node v_1^{in}.

What remains is to complete the Eulerian walk covering (a) edges and their antiparallel
counterparts corresponding to edges in G that were not used in the Hamiltonian path, and
(b) the edges adjacent to node w. To do this, define $\text{pred}(v)$ be the vertices u in G for which
edge (u,v) exists and u is not the predecessor of v along the Hamiltonian path. For each
$u \in \text{pred}(v_1)$, traverse the split edge labeled `c#` to u^{out} then traverse the forward split edge
labeled `b#` back to v_1^{in}. This results in a string $(\texttt{c\#b\#})^{|\text{pred}(v_1)|}$. Once the predecessors of v_1
are exhausted, traverse the split edge labeled `c#` from v_1^{in} into node w and then traverse
the split edge labeled `b#` to v_2^{in}. This again generates a `c#b#` string. Repeat the process,
covering the edges of v_2's predecessors and returning to w to move to the next node along
the Hamiltonian path for each node $v_3, \ldots, v_n$. After covering the predecessors of v_n^{in}, go to
v_1^{in} through the remaining edges in E_5 and E_6, (v_n^{in}, w) and (w, v_1^{in}), which completes the
Eulerian tour. This covers all the edges of G''. The word spelled out in this last section of
the Eulerian trail is a sequence of repetitions of `c#b#`, with one repetition for each edge that
is not in the Hamiltonian path ($|E| - n + 1$) and all of the edges in E_5 and E_6 for entering
and leaving each node ($2n$), with a total of $|E| + 1$ repetitions, which is the final $(\texttt{c\#b\#})^{|E|+1}$
term in q.

This ends the slight modification of the proof in Kupferman and Vardi [10], where the
differences are (a) the introduction of the `#` characters and (b) using the exponent $|E| + 1$
of the final part of q instead of $|E| + n + 1$ as in Kupferman and Vardi [10] since we create
w-edges only to v^{in} vertices. (This second change has no material effect on the proof, but
reduces the length of the string that must be matched.)

Now, given an instance $\langle G = (V, E) \rangle$ of HAMILTONIAN PATH, with $n = |V|$ vertices, we
construct G'' as above (obtaining a unidirectional Eulerian graph) and create graph C that
only represents string q. Note that $|\Sigma| = 4$ and G'' and C can be constructed in polynomial
time. $\text{GTED}(G'', C) = 0$ if and only if an Eulerian path in G'' spells out q, since there can
be no indels or mismatches. By the above argument, an An eulerian tour that spells out q
exists if and only if G has a Hamiltonian path.   ◀

A.3   FGTED is NP-complete

▶ **Problem 4** (Flow Graph Traversal Edit Distance (FGTED) [21]). *Given unidirectional,*
edge-labeled Eulerian graphs G_1 and G_2, each of which has distinguished s_1, s_2 source and
t_1, t_2 sink vertices, compute

$$FGTED(G_1, G_2) \triangleq \min_{\substack{D_1 \in flow(G_1, s_1, t_1) \\ D_1 \in flow(G_2, s_2, t_2)}} emedit(strset(D_1), strset(D_2)), \tag{39}$$

where $flow(G_i, s_i, t_i)$ is the collection of all possible sets of s_1-t_1 trail decomposition of
saturating flow from s_i to t_i, $strset(D)$ is the multi-set of strings constructed from trails in D.

▶ **Theorem 4.** *FGTED is NP-complete.*

Proof. Let $G = (v, E)$ be an instance of the HAMILTONIAN CYCLE problem. Let $n = |V|$ be
the number of vertices in G. Construct the Eulerian closure of G and split the anti-parallel
edges. Let the new graph be $G' = (V', E')$. Attach a source s and a sink node t to an
arbitrary node v_1^{in} by adding edge (s, v_1^{in}) and (v_1^{in}, t) with labels `s` and `t`, respectively.

Construct a string q, such that

$$q = \texttt{sa\#(b\#a\#)}^{n-1}\texttt{(c\#)}^{2n-1}\texttt{(c\#b\#)}^{|E|+1}\texttt{t}. \tag{40}$$

Create a graph Q that only contains one path with labels on the edges of the path that spell the string q. The union of the set of trails in any flow decomposition of G' is equal to a set of Eulerian trails, $\mathcal{E}$, that starts at s and ends at t. All Eulerian trails in $\mathcal{E}$ are also closed Eulerian trails of $G' \setminus \{s, t\}$ that starts and ends at v_1^{in}.

Using the same line of argument in the proof of Theorem 3, an Eulerian trail in G' that spells q is equivalent to a Hamiltonian cycle in G. In addition, $\text{FGTED}(Q, G') = 0$ if and only if all Eulerian trails in $\mathcal{E}$ spell out q. Therefore, if $\text{FGTED}(Q, G') = 0$, then there is a Hamiltonian cycle in G. Otherwise, then there must not exist a Hamiltonian cycle in G. ◄

B   Correctness of the ILP formulation for CCTED

We show that the ILP in (5)–(8) proposed by Ebrahimpour Boroojeny et al. [6] solves CCTED.

▶ **Theorem 9.** *Given two input graphs G_1 and G_2, the optimal objective value of the ILP in (5)–(8) based on $\mathcal{A}(G_1, G_2)$ is equal to $CCTED(G_1, G_2)$.*

Proof. As shown in the proof of Lemma 8, any pair of edge-disjoint, closed-trail covers in the input graph can be projected to a set of closed trails in $\mathcal{A}(G_1, G_2)$, which satisfied constraints (6)-(8). The objective of this feasible solution, which is the total cost of the projected closed trails, equals CCTED. Therefore, $CCTED(G_1, G_2)$ is greater than or equal to the objective of the ILP in (5)–(8).

Conversely, we can transform any feasible solution of the ILP in (5)–(8) to a pair of covers of G_1 and G_2. We can do this by transforming one closed trail at a time from the subgraph of the alignment graph, $\mathcal{A}'$ induced by edges with ILP variable $x_{uv} = 1$. Let c be a closed trail in $\mathcal{A}'$. Let $c_1 = \Pi_1(c)$ and $c_2 = \Pi_2(c)$ be two closed trails in G_1 and G_2 that are projected from c. We can construct an alignment between $\text{str}(c_1)$ and $\text{str}(c_2)$ from c by adding match or insertion/deletion columns for each match or insertion/deletion edges in c accordingly. The cost of the alignment is equal to the total cost of edges in c by the construction of the alignment graph. We can then remove edges in c from the alignment graph and edges in c_1 and c_2 from the input graphs, respectively. The remaining edges in $\mathcal{A}'$ and G_1 and G_2 still satisfy the constraints (6)-(8). Repeat this process and we get a total cost of $\sum_{e \in E} x_e \delta(e)$ that aligns pairs of closed trails that form covers of G_1 and G_2. This total cost is greater than or equal to $CCTED(G_1, G_2)$. ◄

C   The average wall-clock time to solve ILPs on 3-cycle graphs

🟨 **Table S1** The average wall-clock time to solve lower bound ILP, exponential ILP, compact ILP and the number of iterations for pairs of 3-cycle graphs for each $\text{GTED} - \text{CCTED}$.

GTED - CCTED	lower bound ILP runtime (s)	GTED iterative runtime (s)	Iterations	GTED compact runtime (s)
1.0	0.06	0.17	3.55	0.39
2.0	0.05	0.87	13.00	0.43
3.0	0.08	25.41	67.60	1.24
4.0	0.07	205.59	179.10	1.70
5.0	0.08	943.68	502.85	5.37