The Computational Complexity of the ChordLink Model

Philipp Kindermann[1] ⓘ *Jan Sauer*[2] *Alexander Wolff*[2] ⓘ

[1]Universität Trier, Trier, Germany
[2]Universität Würzburg, Würzburg, Germany

Abstract. In order to visualize well-clustered graphs with many intra-cluster but few inter-cluster edges, hybrid approaches have been proposed. For example, ChordLink draws the clusters as chord diagrams and embeds these into a node-link diagram that represents the overall structure of the clustered graph. The ChordLink approach consists of four steps; node replication, node permutation, node merging, and chord insertion. In this paper, we focus on the optimization problems defined by two of these steps. We show that the decision version of the problem defined by node permutation is NP-complete and present an efficient algorithm for a special case. For chord insertion, we show that it is NP-complete to decide whether a crossing-free placement of the chords exists. Moreover, it is APX-hard to minimize the number of crossings among the chords. Our results answer an open question posed by Angori, Didimo, Montecchiani, Pagliuca, and Tappini, who introduced ChordLink [TVCG 2021].

1 Introduction

Node-link diagrams represent an intuitive tool for visualizing graphs. For dense graphs, however, node-link diagrams tend to degenerate into unintelligible hairballs. Less intuitive, but more robust visualization paradigms such as adjacency matrices can be a remedy. In practice, however, large graphs are often "globally sparse" and just "locally dense" [3]. This is the case, for example, in social networks such as collaboration and financial networks [4], but also in biological networks [9]. For visualizing such graphs, hybrid representations have been invented. In intersection-link representations, for example, each vertex is represented by a geometric object and each edge is either represented by a curve connecting the two objects or, if it belongs to a dense subgraph, by a non-empty intersection of the two objects [1, 2]. Another example of a hybrid representation is NodeTrix [7], which uses matrices for dense subgraphs and links between the matrices for the global

---

An earlier version of this article appeared in Proc. 38th Europ. Workshop Comput. Geom. (EuroCG 2022).

graph structure. ChordLink, recently introduced by Angori, Didimo, Montecchiani, Pagliuca, and Tappini [3], combines very effectively so-called chord diagrams [8] for dense subgraphs, again with links between them for the overall graph structure. In a chord diagram, each vertex is represented by one or several circular arcs, and each edge is a chord between any two arcs that represent the endpoints of the edge. Turning a given node-link diagram into a ChordLink visualization can, for example, be triggered by a user in an interactive system. We now formalize the ChordLink model and the four steps that are performed in order to compute a ChordLink visualization from a node-link diagram. To this end, for a graph G, let $V(G)$ be its vertex set and let $E(G)$ be its edge set. For a positive integer k, let $[k] = \{1, 2, \ldots, k\}$.

The ChordLink Model.   Given a node-link diagram Γ of a graph G, a cluster $C \subseteq V(G)$, and a circle R that contains only the vertices in C (at their positions in Γ), a ChordLink visualization of G locally modifies Γ such that $G[C]$ is drawn as a chord diagram with the vertices of C on R. There are four steps; see Figure 1.

NodeReplication: For each node $w \in C$ connected to a node $u \notin C$, create a copy w_u of w on the intersection of the edge (w, u) with R, and add the edge (w_u, u); see Figure 1b. For each node $v \in C$ whose complete neighborhood is contained in C (as the orange square node in Figure 1a), place a copy of v at an arbitrary position on R.

NodePermutation: Copies v_u and w_u of different nodes v and w that are connected to the same node $u \notin C$ can be exchanged in the order of the node copies on R. This step naturally defines the optimization problem NODEPERMUTATION, where the aim is to find a permutation of the node copies on R that exchanges only copies connected to the same node outside of R and maximizes the total number of pairs of consecutive copies of the same node; see Figure 1c. (The number of such pairs increases by two when going from Figure 1b to Figure 1c.)

NodeMerging: Replace each maximal subsequence of consecutive copies of a node w along R by a circular arc c_w; see Figure 1d.

ChordInsertion: For each edge $(v, w) \in G[C]$, select an arc c_v representing v and an arc c_w representing w, and insert a chord that connects c_v and c_w in the interior of R; see Figure 1e. Angori et al. [3] suggest to minimize the total number of crossings among the chords. (They also suggest maximizing the smallest angle formed by any pair of crossing chords, but we do not consider this here.) This defines the optimization problem CROSSINGMINIMAL CHORD-INSERTION.

Known Results.   For NODEPERMUTATION, Angori et al. [3] describe a dynamic program that yields optimal solutions if the node copies that are adjacent to the same external node form intervals along R. If this condition does not hold (as in Figure 1b), then they simply split R into maximal pieces where the condition does hold and treat each piece seperately, which yields a heuristic overall solution for the whole instance. Their algorithm takes $O(|E(G)|^3)$ time.

For CROSSINGMINIMAL CHORDINSERTION, Angori et al. suggest a greedy algorithm that first draws the chords whose endpoints are both represented by unique arcs. Then it adds the other chords one by one, making the currently best choice in terms of crossings (and, with lower weight, in terms of crossing angles). It draws chords as Bézier curves; see Figure 1e.
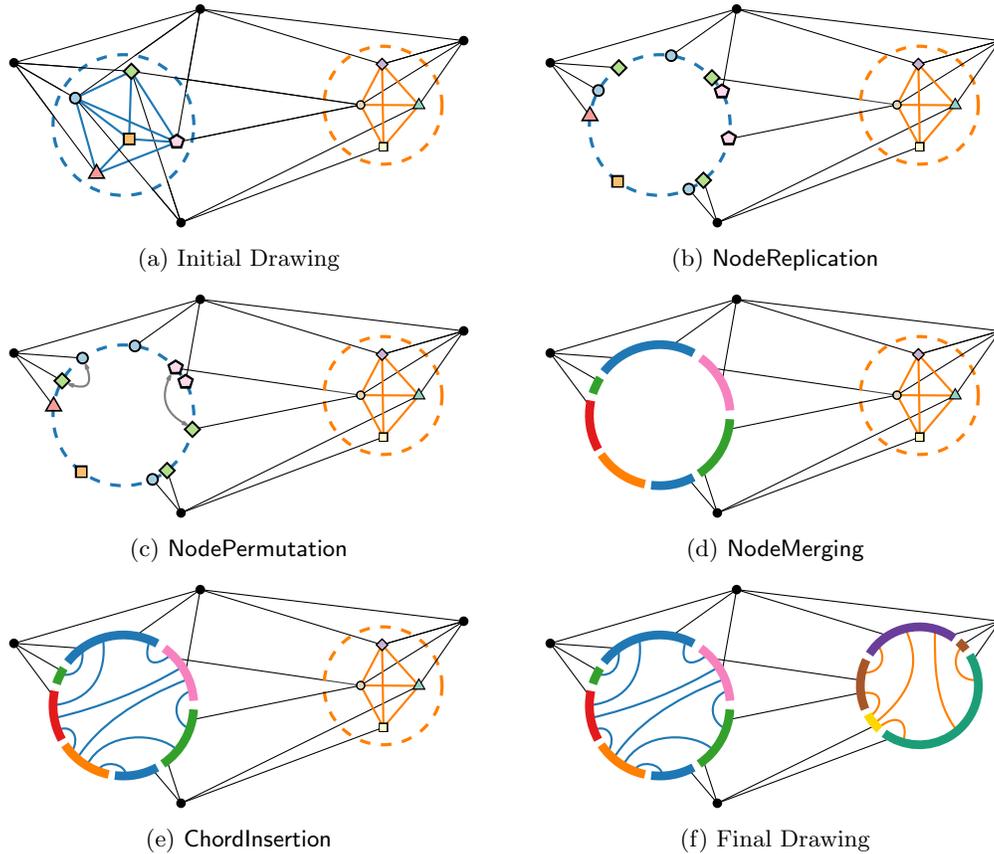
Figure 1: The steps of the ChordLink approach illustrated on the left cluster.

Contribution.   We focus on the steps NODEPERMUTATION and CHORDINSERTION of the Chord-Link model. First, we prove that NODEPERMUTATION is NP-complete; see Section 2. We reduce from VERTEXCOVER in cubic graphs. Then, we give an efficient algorithm for NODEPERMUTATION for the special case that the neighborhood of C contains only two vertices; see Section 3. Finally, we show that (even a rather special case of) CROSSINGMINIMAL CHORDINSERTION is APX-hard, even if there are only two possible choices for every edge; see Section 4. We reduce from MAX-2-SAT.

2   NP-Completeness of NODEPERMUTATION

Above, we have stated NODEPERMUTATION as an optimization problem. We now formally define the corresponding decision problem. In the ChordLink model, for every vertex c in the cluster C and each neighbor $g \notin C$ of c, a copy of c is placed on the circle R. Since G is simple, each copy can be described as a unique pair (c, g). Abstracting from the original problem, we call c the *color* and g the *group* of the pair (c, g). This leads to the following formulation of the problem, where we associate every vertex of C with a distinct color.

Let $\mathcal{C}$ be a set of colors, let $\mathcal{G}$ be a set of groups, and let $L = (L_1, \ldots, L_n, L_1)$ be a circular

list of *distinct* pairs where, for $i \in [n]$, $L_i = (c_i, g_i) \in \mathcal{C} \times \mathcal{G}$. Define $\mathcal{G}(L) = (g_1, \ldots, g_n, g_1)$, $\mathcal{C}(L) = (c_1, \ldots, c_n, c_1)$, and let $N(L)$ be the number of pairs of consecutive equal entries of $\mathcal{C}(L)$. Given $\mathcal{C}, \mathcal{G}, L$, and an integer $K > 0$, find a permutation π of L such that (i) $\mathcal{G}(\pi(L)) = \mathcal{G}(L)$ and (ii) $N(\pi(L)) \geq K$. Note that requirement (i) ensures that we can permute only elements of L that belong to the same group.

Theorem 1 NODEPERMUTATION is NP-complete.

Proof: The problem is in NP since we can verify a permutation easily.

To show hardness, we reduce from 3SETCOVER. This problem generalizes VERTEXCOVER in cubic graphs, which is NP-hard [5]. In the decision version of 3SETCOVER, given a finite universe U (the edge set of the cubic graph), a family $\mathcal{S}$ of size-3 subsets of U (for each vertex, its three incident edges), and an integer $k > 0$, the task is to find a subfamily $\mathcal{S}'$ (corresponding to a vertex cover) of $\mathcal{S}$ of size at most k that covers U. (In the special case of VERTEXCOVER, each element of the universe appears in exactly two elements of $\mathcal{S}$.)

Given an instance $(U, \mathcal{S}, k)$ of 3SETCOVER, we construct an instance $(\mathcal{G}, \mathcal{C}, L, K)$ of NODE-PERMUTATION such that one is a yes-instance if and only if the other is a yes-instance. Let $U = \{u_1, \ldots, u_n\}$, $\mathcal{S} = \{S_1, \ldots, S_m\}$, and, for $i \in [m]$, let $S_i = \{u_{i_1}, u_{i_2}, u_{i_3}\}$ with $i_1, i_2, i_3 \in [n]$. To construct an instance of NODEPERMUTATION, we use only a single color $c^\star$ that appears in more than one entry; all other entries have a unique color. Furthermore, we have one group for every $u_i \in U$, and there is exactly one entry $(c^\star, u_i)$, and we have one additional group z such that there is no entry $(c^\star, z)$; the entries with group z basically serve as *blockers* between the gadgets. The full reduction is as follows:

$$K = n - k,$$
$$\mathcal{G} = U \cup \{z\}, \text{where } z \notin U, \text{ and}$$
$$\mathcal{C} = \{c^\star\} \cup \{c_1, \ldots, c_n\} \cup \bigcup_{i=1}^{m} \{c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4}, c_{i,5}\}, \text{and}$$
$$L^0 = \left((c^\star, u_1), (c_1, z), (c^\star, u_2), (c_2, z), \ldots, (c^\star, u_n), (c_n, z) \right)$$
$$L^i = \left((c_{i,1}, u_{i_1}), (c_{i,2}, u_{i_2}), (c_{i,3}, u_{i_3}), (c_{i,4}, u_{i_1}), (c_{i,5}, z) \right) \text{ for each } i \in [m]$$
$$L = L^0 \oplus L^1 \oplus \cdots \oplus L^m, \text{where } \oplus \text{ concatenates lists.}$$

Clearly, this reduction can be performed in polynomial time. Intuitively, every sublist L^i that contains a color-$c^\star$ entry corresponds to a set S_i in a solution $\mathcal{S}'$ of $\mathcal{S}$. If these sublists contain K consecutive color-$c^\star$ entries, then there are $2K$ elements that are covered by K sets in $\mathcal{S}'$, so $|\mathcal{S}'| \leq n - K = k$.

First, we assume that $(U, \mathcal{S}, k)$ is a yes-instance of 3SETCOVER, that is, there is a size-k subfamily $\mathcal{S}'$ of $\mathcal{S}$ that covers U. We need to construct a permutation π of L such that (i) $\mathcal{G}(\pi(L)) = \mathcal{G}(L)$ and (ii) $N(\pi(L)) \geq K$.

For each $j \in [n]$, let i be the index of an arbitrary set in $\mathcal{S}'$ that contains u_j. Swap $(c^\star, u_j)$ in L^0 with an entry in L^i with group u_j. There is a choice only if $u_j = u_{i_1}$. If $(c_{i,2}, u_{i_2})$ has been or will be swapped with another entry, too, then swap $(c^\star, u_j)$ with $(c_{i,1}, u_{i_1})$; otherwise, swap $(c^\star, u_j)$ with $(c_{i,4}, u_{i_1})$. This makes sure that all swapped entries in L^i are consecutive. In total, we make n swaps. In the resulting permutation of L, the elements of color u form at most k groups of consecutive entries (as $\mathcal{S}'$ might contain "unnecessary" sets in the decision version). Hence, the number of pairs of consecutive entries with the same color $c^\star$ is at least $n - k = K$.

Now assume that $(\mathcal{G}, \mathcal{C}, L, K)$ is a yes-instance of NODEPERMUTATION, that is, there is a permutation π of L such that (i) $\mathcal{G}(\pi(L)) = \mathcal{G}(L)$ and (ii) $N(\pi(L)) \geq K$. We have to show that
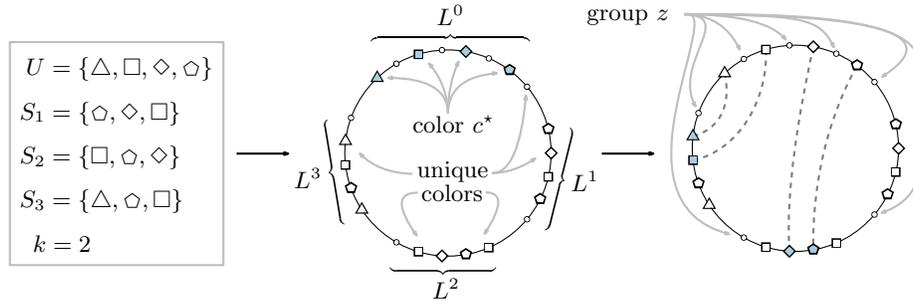
Figure 2: Here, the reduction from 3SETCOVER to NODEPERMUTATION yields the cover $\{S_2, S_3\}$. Color $c^\star$ is drawn in blue; all other list entries have their own unique color. List elements of group z are indicated by a $\circ$.

then $(U, \mathcal{S})$ admits a set cover of size $k = n - K$. Without loss of generality, we can assume that π swaps *each* color-$c^\star$ element of L^0 with a color-$c^\star$ element of $L^1 \oplus \cdots \oplus L^m$ (because such a swap does not decrease $N(\pi(L))$) and that π does not swap any other entries of L (because swapping group-z elements does not change $N(\pi(L))$). Consider the family $\mathcal{S}'$ of those sets $S_i \in \mathcal{S}$ such that π modifies L^i. We claim that (i) $\mathcal{S}'$ covers U and (ii) $|\mathcal{S}'| \leq k$.

Property (i) holds due to our assumption that π swaps all color-$c^\star$ entries of L^0 with a sublist L^i with $i \in [m]$. Thus, every element of U is contained in an element of $\mathcal{S}'$. For property (ii), note that the only pairs of consecutive entries with equal color in $\pi(L)$ are pairs of color-$c^\star$ entries in sublists L^i. Among the n color-$c^\star$ entries, at least K pairs are consecutive. Let K_1, K_2 and K_3 be the number of sublists L^i that contain one, two and three color-$c^\star$ entries, respectively. Then we have $K_2 + 2K_3 \geq K$ and $K_1 = n - 2K_2 - 3K_3$. Hence, the total number of sublists L^i that contain at least one color-$c^\star$ entry is $K_1 + K_2 + K_3 = n - K_2 - 2K_3 \leq n - K = k$, so $|\mathcal{S}'| \leq k$. $\square$

3   An Algorithm for a Special Case of NODEPERMUTATION

In this section, we describe a linear-time algorithm to solve the optimization version of NODEPERMUTATION for the special case that there are only two groups (but an arbitrary number of colors). In the ChordLink model, this corresponds to an arbitrarily large cluster C where the neighborhood of C in $V(G) \setminus C$ has size 2.

Theorem 2 NODEPERMUTATION can be solved in $O(n)$ time for two groups.

Proof: Let $\mathcal{G} = \{x, y\}$. The goal is to find a permutation π of L that maximizes $N(\pi(L))$.

Consider any three consecutive indices $i, i+1, i+2$. Recall that the elements of L are pairwise disjoint. Thus, $c_{\pi^{-1}(i)} = c_{\pi^{-1}(i+1)}$ implies that $g_i \neq g_{i+1}$. Since there are only two groups, we have either $g_{i+2} = g_i$ or $g_{i+2} = g_{i+1}$, so $c_{\pi^{-1}(i+2)} \neq c_{\pi^{-1}(i+1)}$. Hence, for any permutation π of L, we cannot have $c_{\pi^{-1}(i)} = c_{\pi^{-1}(i+1)} = c_{\pi^{-1}(i+2)}$.

For $\circ \in \{x, y\}$, let $\mathcal{C}_\circ$ be the set of colors c_i such that there exists some list element $(c_i, \circ)$, and let $S = \mathcal{C}_x \cap \mathcal{C}_y$. We say that the color c_j is *assigned* to index i if $c_{\pi^{-1}(i)} = c_j$.

We can formulate this problem as a maximum independent set problem on a graph G'. The graph contains a vertex v_i' for every pair of consecutive indices $i, i+1$ with $g_i \neq g_{i+1}$, and a vertex v_n' if $g_n \neq g_1$. If $v_i', v_{i+1}' \in V(G')$, then $E(G')$ contains the edge (v_i', v_{i+1}'). Any assignment of colors

to K pairs of consecutive indices $(i_1, i_1 + 1), \ldots, (i_K, i_K + 1)$ with $g_{i_1} \neq g_{i_1+1}, \ldots, g_{i_K} \neq g_{i_K+1}$ induces an independent set $v'_{i_1}, \ldots, v'_{i_K}$ in G'.

Hence, if we find an independent set $(v'_{i_1}, \ldots, v'_{i_k})$ of size k in G', then we can find an assignment of $K = \min\{|S|, k\}$ colors in S to consecutive pairs of indices $(i_1, i_1 + 1), \ldots, (i_K, i_K + 1)$. By construction, G' is either a cycle or a linear forest, so we can find a maximum independent set of G' in linear time with a simple greedy algorithm. To obtain a permutation π of L with $N(\pi(L)) = K$, we arbitrarily assign the remaining colors of $\mathcal{C}_x$ and $\mathcal{C}_y$ to the remaining list elements of types (c_i, x) and (c_i, y), respectively. □

4   APX-Hardness of CrossingMinimal ChordInsertion

In this section, we focus on the optimization problem CrossingMinimal ChordInsertion: Given a graph G with at least one copy of each of its vertices placed on a circle R, insert every edge between a copy of each of its endvertices such that the total number of crossings between the edges is minimized. Since the number of crossings only depends on the order of the vertex copies along R, we can also assume them to be drawn as points (rather than circular arcs) on R. We first prove that finding a crossing-free solution for ChordInsertion is NP-complete.

Theorem 3 *It is NP-complete to decide whether a given graph G (with node copies on R) admits a crossing-free solution for ChordInsertion, even if G is a matching and there are at most three copies per vertex.*

Proof: Membership in NP is obvious since we can verify an assignment easily.

To show hardness, we reduce from 3-SAT. Let $(\mathcal{X}, \mathcal{C})$ be a 3-SAT instance with variables $\mathcal{X} = X_0, \ldots, X_n$ and clauses $\mathcal{C} = C_0, \ldots, C_m$. We create a graph G as follows. The vertex set $V(G)$ contains, for each $i \in [m]$, two *clause vertices* c_i and d_i; and, for each $j \in [n]$, two *variable vertices* x_j and y_j. The edge set $E(G)$ contains for each $i \in [m]$, a *clause edge* (c_i, d_i); and for each $j \in [n]$, a *variable edge* (x_j, y_j) and a *separator edge* (z_j, z'_j). For each $j \in [n]$, let T_j be the set of clauses that contain the literal X_j, and let F_j be the set of clauses that contain the literal $\neg X_j$.

To create an instance I of ChordInsertion, we place copies of the vertices of $V(G)$ along R as follows; see Figure 3. For every variable X_j, we place, in the given order,

1. the vertex z_j;
2. for every clause $C_i \in T_j$, a copy d_i^j of d_i, in decreasing order of i;
3. one copy y_j^F of y_j;
4. for every clause $C_i \in T_j$, a copy c_i^j of c_i, in increasing order of i;
5. the vertex x_j;
6. for every clause $C_i \in F_j$, a copy c_i^j of c_i, in increasing order of i;
7. a second copy y_j^T of y_j;
8. for every clause $C_i \in F_j$, a copy d_i^j of d_i, in decreasing order of i.
9. the vertex z'_j;

Assume that I is a yes-instance, i.e., it admits a crossing-free drawing Γ. Then, for each $i \in [n]$, the drawing Γ must contain either the chord (x_i, y_i^F) or the chord (x_i, y_i^T), simply because x_i is unique and y_i has exactly two copies; see Figure 4.
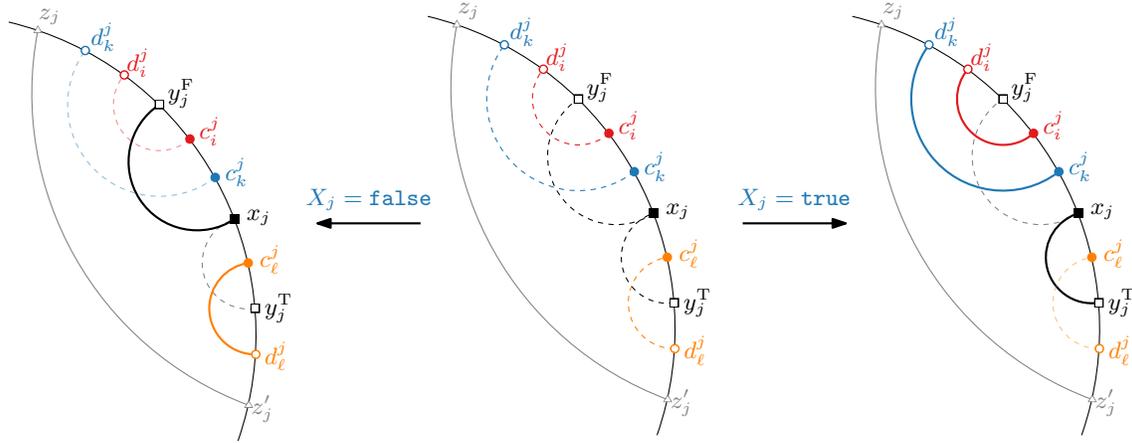
Figure 3: Illustration for the part corresponding to variable X_j in the reduction from 3-SAT to CHORDINSERTION for the formula $c_i \wedge c_k \wedge c_\ell \wedge \ldots = (X_j \vee \ldots) \wedge (\overline{X_j} \vee \ldots) \wedge (\overline{X_j} \vee \ldots) \wedge \ldots$.

Let $i \in [m]$. Then Γ must contain a chord between a copy of c_i and a copy of d_i. Because of the seperator edges, whose positions are fixed, this chord cannot connect copies c_i^j and d_i^k with $j \neq k$. Next, consider any literal X_j contained in C_i. Then, the clause edge (c_i, d_i) can be drawn as the chord (c_i^j, d_i^j) only if Γ contains the chord (x_i, y_i^{T}); otherwise, there would be a crossing between (c_i^j, d_i^j) and (x_i, y_i^{F}). Conversely, for any literal $\neg X_j$ contained in C_i, the edge (c_i, d_i) can be drawn as the chord (c_i^j, d_i^j) only if Γ contains the chord (x_i, y_i^{F}). Hence, for the edge (c_i, d_j) to be drawn in Γ, for at least one of its positive literals X_j, there must be the chord (x_j, y_j^{T}), or for at least one of its negated literals $\neg X_j$, there must be the chord (x_j, y_j^{F}).

Thus, we obtain a feasible solution for the 3-SAT instance $(\mathcal{X}, \mathcal{C})$ as follows: for each $j \in [n]$, if Γ contains the chord (x_j, y_j^{T}), then set $X_j = \mathtt{true}$, otherwise (Γ contains the chord (x_j, y_j^{F})), set $X_j = \mathtt{false}$. Since, for each $i \in [m]$, the edge (c_i, d_i) must be drawn as a chord in Γ, at least one literal in the clause C_i must be true, so the solution is feasible.

Now assume that $(\mathcal{X}, \mathcal{C})$ is a yes-instance, i.e., there exists an assignment $\phi \colon \mathcal{X} \to \{\mathtt{true}, \mathtt{false}\}$ such that, for each $i \in [m]$, at least one literal of the clause C_i is satisfied. We obtain a crossing-free drawing Γ for the corresponding instance I of CHORDINSERTION as follows. For each separator edge, both its end vertices are unique, so we draw a chord between them. For each $j \in [n]$, if $\phi(X_j) = \mathtt{true}$, we draw the variable edge (x_j, y_j) as the chord (x_j, y_j^{T}), otherwise as (x_j, y_j^{F}). For every $i \in [m]$, we choose a satisfied literal X_j (or $\neg X_j$) of the clause C_i and draw the clause edge (c_i, d_i) as the chord (c_i^j, d_i^j). By construction, this chord cannot intersect the edge (x_j, y_j), and, as observed above, it does not cross any other chord. Hence, we obtain a crossing-free drawing Γ for I. □

In the above proof we reduced from 3-SAT. If we instead reduce from MAX-2-SAT, which is APX-hard [6], we can show that our problem is even APX-hard.

Theorem 4 CROSSINGMINIMAL CHORDINSERTION is APX-hard even for matchings with at most two copies per vertex.

Proof: By reducing from MAX-2-SAT, the construction in the proof of Theorem 3 can be used to show that CROSSINGMINIMAL CHORDINSERTION is APX-hard even if there are at most two

$$C_1 = X_1 \vee \overline{X_2} \vee X_3$$
$$C_2 = \overline{X_1} \vee X_3 \vee X_4$$
$$C_3 = \overline{X_2} \vee \overline{X_4} \vee X_5$$
$$C_4 = X_2 \vee \overline{X_3} \vee X_5$$
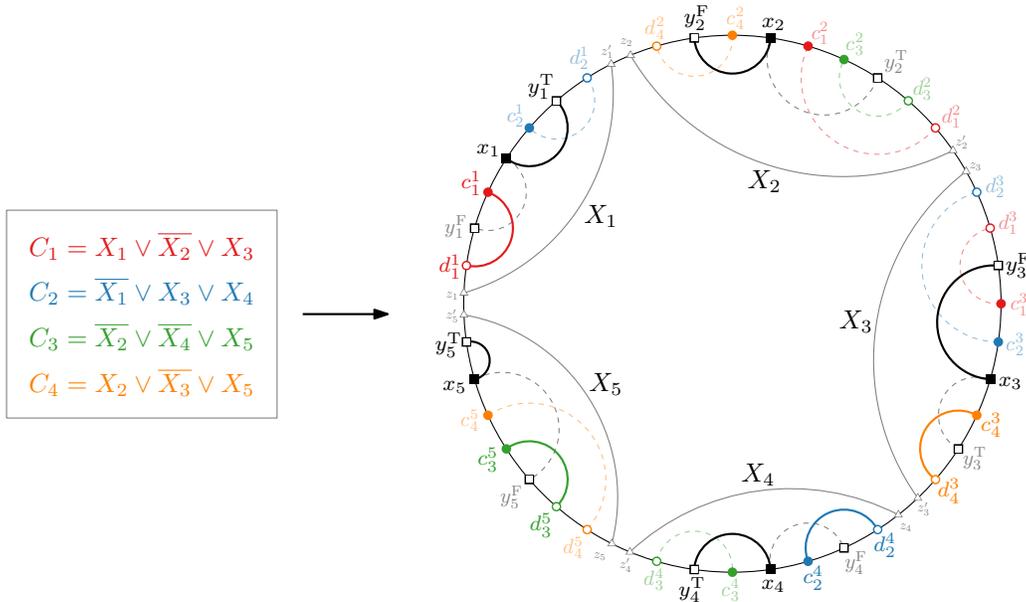
Figure 4: Example of the reduction from 3-SAT to CHORDINSERTION

possible choices for every edge. It is known that MAX-2-SAT is inapproximable within any factor greater than $21/22 > 0.954$ [6]. For the proof, observe that in the construction above a clause is unsatisfied if and only if its corresponding clause edge has a crossing with the variable edge of one of its unsatisfied literals. (In particular, it is never worth for a clause edge to cross separation edges.) Hence, the number of crossings in the optimum solution for the corresponding CROSSING-MINIMAL CHORDINSERTION instance is exactly the number of unsatisfied clauses in an optimum solution for MAX-2-SAT.   □

5   Conclusion

The dynamic program of Angori et al. [3] for NODEPERMUTATION insists that all node copies with the same group are consecutive along R. We have described an algorithm that assumes that there are just two groups. Are there other restrictions that render NODEPERMUTATION easy and are acceptable in practice? We have shown that CROSSINGMINIMAL CHORDINSERTION is inapproximable within any factor greater than $21/22 > 0.954$, but what about a, say, $1/2$-approximation?

References

[1] Patrizio Angelini and Giordano Da Lozzo. Beyond clustered planar graphs. In Seok-Hee Hong and Takeshi Tokuyama, editors, *Beyond Planar Graphs: Communications of NII Shonan Meetings*, pages 211–235. Springer, 2020. `doi:10.1007/978-981-15-6533-5_12`.

[2] Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. Intersection-link representations of graphs. *J. Graph Algorithms Appl.*, 21:731–755, 2017. `doi:10.7155/jgaa.00437`.

[3] Lorenzo Angori, Walter Didimo, Fabrizio Montecchiani, Daniele Pagliuca, and Alessandra Tappini. Hybrid graph visualizations with ChordLink: Algorithms, experiments, and applications. *IEEE Trans. Vis. Comput. Graphics*, 28(2):1288–1300, 2020. `doi:10.1109/TVCG.2020.3016055`.

[4] Punam Bedi and Chhavi Sharma. Community detection in social networks. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 6(3):115–135, 2016. `doi:10.1002/widm.1178`.

[5] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete problems. In Robert L. Constable, Robert W. Ritchie, Jack W. Carlyle, and Michael A. Harrison, editors, *Proc. 6th Ann. ACM Symp. Theory Comput. (STOC)*, pages 47–63, 1974. `doi:10.1145/800119.803884`.

[6] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. `doi:10.1145/502090.502098`.

[7] Nathalie Henry Riche, Jean-Daniel Fekete, and Michael McGuffin. NodeTrix: A hybrid visualization of social networks. *IEEE Trans. Vis. Comput. Graphics*, 13(6):1302–1309, 2007. `doi:10.1109/TVCG.2007.70582`.

[8] Martin Krzywinski, Jacqueline Schein, Inanç Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J. Jones, and Marco A. Marra. Circos: An information aesthetic for comparative genomics. *Genome Res.*, 19(9):1639–1645, 2009. `doi:10.1101/gr.092759.109`.

[9] Hassan Mahmoud, Francesco Masulli, Stefano Rovetta, and Giuseppe Russo. Community detection in protein-protein interaction networks using spectral and graph approaches. In Enrico Formenti, Roberto Tagliaferri, and Ernst Wit, editors, *Proc. 10th Int. Meeting Comput. Intell. Methods for Bioinf. Biostat. (CIBB)*, volume 8452 of *Lect. Notes Comput. Sci.*, pages 62–75. Springer, 2013. `doi:10.1007/978-3-319-09042-9\_5`.