MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **97** (2027) 821–870
ISSN: 0340–6253
doi: 10.46793/match.97-3.12126

Exact Enumeration of All Connected Maximum Common Subgraphs in Multiple Labeled Graphs: Application to Cheminformatics

Johannes B. S. Petersen[a], Akbar Davoodi[a,*], Thomas Gärtner[b], Marc Hellmuth[c], Daniel Merkle[d,e,a]

[a] *Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark*

[b] *Machine Learning Research Unit, TU Wien, Vienna, Austria*

[c] *Department of Mathematics, Stockholm University, Stockholm, Sweden*

[d] *Algorithmic Cheminformatics Group, Bielefeld University, Bielefeld, Germany*

[e] *Center for Biotechnology (CeBiTec), Bielefeld University, Bielefeld, Germany*

akb@sdu.dk

(Received April 5, 2026)

Abstract

We present an exact algorithmic framework for enumerating all maximum common subgraphs shared by multiple vertex- and edge-labeled graphs, motivated by molecular-graph comparison in cheminformatics and computational chemistry and, more generally, by comparison problems on labeled networks. The framework addresses maximum common induced subgraphs (MCIS), maximum common edge subgraphs (MCES), and their connected variants under label-preserving matching. Algorithmically, it combines labeled modular-

*Corresponding author.

This work is licensed under a Creative Commons "Attribution 4.0 International" license.

product constructions with a modified Bron–Kerbosch clique-enumeration procedure that retains the maximal intermediate candidates needed for exact multi-graph reduction. To improve practical performance, we incorporate pruning of redundant type-0 product edges and similarity-based ordering of the input graphs. Formal correctness proofs, benchmarks on the ZINC and ChEMBL22 molecular datasets, and a publicly available implementation show that the framework yields a reproducible exact method for labeled-network comparison that is practically usable on the studied molecular instance sizes.

1 Introduction

The maximum common subgraph (MCS) problem is a central computational challenge in graph-based modelling across numerous domains, including bioinformatics, cheminformatics, pharmacophore mapping, pattern recognition, computer vision, code analysis, compiler design, and model checking [12, 20, 36, 38]. More broadly, graph-theoretic methods are used in epidemiology to model disease outbreaks [22], in neuroscience to analyze brain networks [15], in landscape ecology to study habitat connectivity [41], and in transportation science to analyze metro networks [16]. Returning to MCS, its importance is underscored by applications such as structural alignment in systems biology [17], where large interaction networks or metabolic pathways must be compared to reveal conserved motifs, and in cheminformatics, where computing an MCS helps uncover shared molecular frameworks and acts as a core ingredient for atom–atom mapping and related tasks in computational chemistry.

Most foundational work addresses the pairwise MCS problem, which is already NP-complete [23]. In practice, however, many cheminformatics workflows require identifying a structural motif shared across more than two molecules [14, 27, 28]. Two classical variants are especially relevant in molecular informatics: the maximum common induced subgraph (MCIS), which seeks the largest vertex-induced subgraph shared across the graphs under comparison, and the maximum common edge subgraph (MCES), which maximizes the number of common edges [5, 7]. Both formulations are commonly approached via clique search in modular (compatibility)

product graphs [5], with foundational methods due to Bron and Kerbosch [8] for maximal clique enumeration and Carraghan and Pardalos [11] for exact maximum-clique search.

Subsequent work improved practical performance on labeled graphs through branch-and-bound methods, backtracking, dynamic programming on special graph classes, pruning strategies, and vertex-ordering heuristics [6, 9, 24, 32, 34, 35, 40, 42]. Heuristic approaches, such as those of Englert and Kovacs, further trade optimality for speed when exact enumeration becomes impractical [21].

For multiple graphs, however, the problem becomes substantially harder: combining pairwise solutions can miss substructures that are common to all inputs [28]. Existing multiple-graph approaches include sequential and parallel heuristics [10], FMCS [14], and MultiMCS [27]. These contributions are practically important, but, to our knowledge, the literature still lacks an exact framework for multiple labeled graphs that both enumerates all optimal solutions and provides formal correctness guarantees for the setting studied here. This motivates the development of an exact framework for common-subgraph analysis across multiple labeled graphs.

To address this gap, we develop an exact algorithmic framework for comparing multiple labeled graphs, with molecular graphs as the primary application domain. The framework covers MCIS and MCES problems, including connected variants, under vertex- and edge-label constraints, and is accompanied by formal correctness proofs. Algorithmically, it combines modular-product constructions with a modified Bron–Kerbosch-based clique-enumeration procedure [5, 8] and three complementary refinements: pruning redundant product edges while preserving the relevant connectivity information, retaining all maximal intermediate candidates during iterative pairwise reduction, and ordering the input graphs by precomputed similarity measures to reduce intermediate problem sizes. Benchmarks on molecular datasets show that these design choices make the exact method practically usable on the studied instances, while the publicly available implementation supports reproducibility and comparative evaluation.

Modeling assumptions and scope. Throughout the paper, molecules are represented as finite simple undirected graphs whose vertex and edge labels encode atom and bond types. Accordingly, all exactness claims are with respect to this molecular representation and to label-preserving matches. For MCES, we work via labeled line graphs; the formal connectedness and admissibility conditions used in this setting are introduced in Section 2. This modeling scope is used consistently throughout the theory and the computational experiments on molecular graphs.

Chemical interpretation of exactness. In the present paper, "exact" always means exact with respect to the chosen molecular-graph representation. The framework therefore exhaustively enumerates optimal common subgraphs under the specified vertex labels, edge labels, and connectedness constraints, but it does not infer or preserve chemical information that is not encoded in those labels. This interpretation applies throughout the theoretical results, the molecular benchmarks, and the reaction-rule example.

Generality beyond the molecular setting. Although the motivating application and the empirical study concern molecular graphs, the formal framework developed here is stated for arbitrary finite simple vertex- and edge-labeled graphs. The same constructions–labeled modular products, type-A connectivity, recursive candidate retention, and the line-graph reduction for MCES–therefore define a general exact method for labeled-network comparison. Molecular graphs serve as the primary validation setting because they provide a natural typed network representation, realistic benchmark data, and an interpretable downstream application.

Beyond molecular similarity assessment, the framework can also serve as a modeling primitive for extracting conserved reaction patterns. Section 7 illustrates this point through an application of the connected MCES framework to graph transformation rule inference from sets of related reactions.

The remainder of the paper is organized as follows. Section 2 introduces the formal definitions and notation. Section 3 summarizes the method and

states the main results, whose proofs are given in Section 4. Section 5 describes the modified Bron–Kerbosch-based pairwise MCS computation, the pruning procedure, and the graph-ordering heuristics. Section 6 presents benchmarking results, and Section 7 discusses the application to graph transformation rule inference. Section 8 concludes the paper.

2 Preliminaries

Graphs. We consider (finite undirected) graphs $G = (V, E)$, that is, V is finite and E is a subset of the 2-element subsets of V. In this case, G is always *simple*, that is, it cannot have multiple edges or edges of the form $\{v, v\}$. For a graph G, we denote its vertex set by $V(G) := V$ and its edge set by $E(G) := E$. We write $G \simeq H$ if the graphs G and H are isomorphic. We denote the Cartesian product of the vertex sets $V(G_1), V(G_2), \ldots, V(G_n)$ by $V(G_1) \times \cdots \times V(G_n)$.

A *clique* in G is a subset $C \subseteq V(G)$ such that every two distinct vertices in C are adjacent in G. A *maximal* clique is an inclusion-maximal clique, whereas a *maximum* clique is a clique of largest cardinality. A clique C of size $|C| = 3$ is a *triangle*. A graph G is a *claw* if $|V(G)| = 4$ and there is a vertex $x \in V(G)$ and edges $\{x, v\} \in E(G)$ for all $v \in V(G) \setminus \{x\}$ and no further edges exist.

A graph $H = (W, F)$ is called a *subgraph* of a graph $G = (V, E)$, in symbols $H \subseteq G$, if $W \subseteq V$ and $F \subseteq E$. A subgraph $H = (W, F)$ of a graph $G = (V, E)$ is called an *induced subgraph* of G if $F = \big\{\{u, v\} \in E \colon u, v \in W\big\}$. For $W \subseteq V(G)$, we denote by $G[W]$ the induced subgraph of G with vertex set W. We often consider cliques $C \subseteq V(G)$ also as the induced subgraph $G[C]$. For $F \subseteq E(G)$, we define the *edge-induced subgraph* $G[F] = (W, F)$ of G by setting $W = \cup_{e \in F} e$ as the set of all vertices incident to edges in F and keeping the edges in F.

Two graphs G and H are isomorphic, in symbols $G \simeq H$, if there is a bijective map $\varphi \colon V(G) \to V(H)$ such that $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$. In this case, φ is an *isomorphism* between G and H. We use the standard notion of common subgraphs *up to isomorphism*. A graph H is called a *common subgraph* of graphs G_1 and G_2 if there

exist subgraphs $H_1 \subseteq G_1$ and $H_2 \subseteq G_2$ such that $H_1 \simeq H_2 \simeq H$. If, moreover, H_1 and H_2 are induced subgraphs of G_1 and G_2, respectively, then H is called a *common induced subgraph* of G_1 and G_2. A common induced subgraph H is a *maximum common (vertex-)induced subgraph* (MCIS) of G_1 and G_2 if $|V(H)|$ is maximum among all common induced subgraphs of G_1 and G_2. Equivalently, there exist induced subgraphs $H_i \subseteq G_i$ with $H_1 \simeq H_2$ such that $|V(H_1)|$ is maximum. Similarly, a *maximum common edge subgraph* (MCES) of G_1 and G_2 is a common subgraph H that maximizes $|E(H)|$ among all graphs that are isomorphic to edge-induced subgraphs of both G_1 and G_2. These definitions naturally generalize to more than two graphs $G_1, \ldots, G_n$ by requiring the existence of subgraphs $H_i \subseteq G_i$ that are pairwise isomorphic (and induced, respectively edge-induced, in the MCIS and MCES settings).

A classical method for finding an MCIS of two graphs G_1 and G_2 proceeds via their modular product [5, 29], which we slightly generalize here.

Definition 1 ((n-ary) modular product)**.** For $i = 1, \ldots, n$, let G_i be a finite simple graph with vertex set V_i and edge set E_i. The (unlabeled, n-ary) modular product $G = \star_{i=1}^{n} G_i$ is the graph with vertex set $V(G) \coloneqq V_1 \times \cdots \times V_n$ and edge set defined as follows. Two distinct vertices $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are adjacent in G if and only if $x_i \neq y_i$ for all $i \in \{1, \ldots, n\}$ and either

(0) $\{x_i, y_i\} \notin E_i$ for all $i \in \{1, \ldots, n\}$, or

(1) $\{x_i, y_i\} \in E_i$ for all $i \in \{1, \ldots, n\}$.

Edges arising from case (0) are called *type-0* edges and edges arising from case (1) are called *type-1* edges.

For $i \in \{1, \ldots, n\}$, let $p_i \colon V(G) \to V(G_i)$ be the coordinate projection, defined by $p_i(x) = x_i$ for $x = (x_1, \ldots, x_n) \in V(G)$. For a subgraph $K \subseteq G$, set $p_i(K) \coloneqq \{p_i(x) \colon x \in V(K)\} \subseteq V(G_i)$.

The subgraph of G_i induced by the vertex set $p_i(K)$, denoted $G_i[p_i(K)]$, is called the *projection of K onto the i-th factor*.

The 2-ary modular product $G_1 \star G_2$ is also known as the *weak modular product* [26].

Labeled Graphs. In order to track specific atom types (vertex labels) and bond types (edge labels), we consider labeled graphs. A *vertex- and edge-labeled graph* is a tuple $\mathscr{G} = (G, \lambda_V, \lambda_E)$, where $G = (V, E)$ is a graph and $\lambda_V \colon V \to \Sigma_V$ assigns a label to each vertex, and $\lambda_E \colon E \to \Sigma_E$ assigns a label to each edge. We also often write $(V, E, \lambda_V, \lambda_E)$ instead of $(G, \lambda_V, \lambda_E)$ and call G the *underlying graph* of $\mathscr{G}$. We write $V(\mathscr{G}) \coloneqq V$ and $E(\mathscr{G}) \coloneqq E$ for its vertex and edge sets.

Note that each unlabeled graph $G = (V, E)$ gives rise to a labeled version $\mathscr{G} = (G, \lambda_V, \lambda_E)$ by putting the labels of all edges and vertices to 1. We call such labeled graphs *uniformly-labeled*.

In the following, let $G = (V, E)$ and $H = (V', E')$ be two graphs. For labeled graphs $\mathscr{G} = (G, \lambda_V, \lambda_E)$ and $\mathscr{H} = (H, \lambda_{V'}, \lambda_{E'})$, an isomorphism $\varphi \colon V \to V'$ between G and H is *label-preserving* if $\lambda_V(v) = \lambda_{V'}(\varphi(v))$ for all $v \in V$ and $\lambda_E(\{u, v\}) = \lambda_{E'}(\{\varphi(u), \varphi(v)\})$ for all $\{u, v\} \in E$. In this case we say that $\mathscr{G}$ and $\mathscr{H}$ are *label-preserving isomorphic*. Suppose now that $H \subseteq G$. Then, we just say that $\mathscr{H}$ is a *subgraph* of $\mathscr{G}$. In particular, a clique of $\mathscr{G}$ refers to a clique in G. Moreover, $\mathscr{H}$ is a *labeled* subgraph of $\mathscr{G}$ if $\lambda_{V'}(u) = \lambda_V(u)$ for all vertices $u \in V'$ and $\lambda_{E'}(e) = \lambda_E(e)$ for all edges $e \in E'$. Moreover, if $\mathscr{H}$ is a labeled subgraph of $\mathscr{G}$ and H is an induced subgraph of G, then $\mathscr{H}$ is an *induced* labeled subgraph of $\mathscr{G}$. Similarly, if $\mathscr{H}$ is a labeled subgraph of $\mathscr{G}$ and H is an edge-induced subgraph of G, then $\mathscr{H}$ is an *edge-induced* labeled subgraph of $\mathscr{G}$.

For a set of labels A and a graph $G = (V, E)$, a labeled graph $\mathscr{G} = (G, \lambda_V, \lambda_E)$ is *`type-`A connected* if for every two distinct vertices $u, v \in V$, there exists a u-v-path in G such that $\lambda_E(e) \in A$ for every edge e of this path. Equivalently, the spanning subgraph $(V, \{e \in E \colon \lambda_E(e) \in A\})$ of G is connected. Note that if $\mathscr{G}$ is uniformly labeled, or if $A = \Sigma_E$ (where $\Sigma_E = \mathrm{im}(\lambda_E)$), then `type-`$A$ connectedness coincides with the usual notion of connectedness.

In the following, let $\mathscr{G}_1 = (G_1, \lambda_{V(G_1)}, \lambda_{E(G_1)})$, $\mathscr{G}_2 = (G_2, \lambda_{V(G_2)}, \lambda_{E(G_2)})$ and $\mathscr{H} = (H, \lambda_{V(H)}, \lambda_{E(H)})$ be labeled graphs. If $\mathscr{H}$ is label-preserving isomorphic to an induced labeled subgraph of $\mathscr{G}_1$ as well as of $\mathscr{G}_2$, then $\mathscr{H}$ is called a *labeled common induced subgraph* of $\mathscr{G}_1$ and $\mathscr{G}_2$. A *maximum labeled common induced subgraph* (labeled MCIS) is a labeled common induced

subgraph $\mathscr{H}$ for which $|V(H)|$ is maximum among all labeled common induced subgraphs of $\mathscr{G}_1$ and $\mathscr{G}_2$. Similarly, if $\mathscr{H}$ is label-preserving isomorphic to an edge-induced labeled subgraph of $\mathscr{G}_1$ as well as of $\mathscr{G}_2$, then $\mathscr{H}$ is called a *labeled common edge-induced subgraph* of $\mathscr{G}_1$ and $\mathscr{G}_2$. A *maximum labeled common edge-induced subgraph* (labeled MCES) is such an $\mathscr{H}$ maximizing $|E(H)|$. These definitions naturally generalize to labeled common induced subgraphs and labeled common edge-induced subgraphs of more than two labeled graphs.

We generalize here the modular product to be able to deal with labeled graphs.

Definition 2 ((n-ary) labeled modular product)**.** Let $\mathscr{G}_i = (V_i, E_i, \lambda_{V_i}, \lambda_{E_i})$, $i = 1, \ldots, n$, be finite simple vertex- and edge-labeled graphs with pairwise disjoint vertex sets. The (n-ary) labeled modular product $\mathscr{G} = \star_{i=1}^{n} \mathscr{G}_i$ is the labeled graph $\mathscr{G} = (V, E, \lambda_V, \lambda_E)$ with vertex set

$$V \coloneqq \big\{(v_1, \ldots, v_n) \in V_1 \times \cdots \times V_n : \lambda_{V_1}(v_1) = \cdots = \lambda_{V_n}(v_n)\big\},$$

and, for two vertices $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in V$, we have $\{x, y\} \in E$ if and only if $x_i \neq y_i$ for all $i \in \{1, \ldots, n\}$ and either

(0) $\{x_i, y_i\} \notin E_i$ for all $i \in \{1, \ldots, n\}$, or

(1) $\{x_i, y_i\} \in E_i$ for all $i \in \{1, \ldots, n\}$ and $\lambda_{E_1}(\{x_1, y_1\}) = \cdots = \lambda_{E_n}(\{x_n, y_n\})$.

Edges arising from case (0) are called *type-0* and are assigned the dummy label $\lambda_E(\{x, y\}) \coloneqq \perp$. Edges arising from case (1) are called *type-1* product edges and receive the common edge label $\lambda_E(\{x, y\}) \coloneqq \lambda_{E_1}(\{x_1, y_1\}) \in \Sigma_E$. Moreover, for each vertex $v = (v_1, \ldots, v_n) \in V$, we put $\lambda_V(v) = \lambda_{V_1}(v_1)$.

Note that if $\mathscr{G}_1 = (G_1, \lambda_{V_1}, \lambda_{E_1})$ and $\mathscr{G}_2 = (G_2, \lambda_{V_2}, \lambda_{E_2})$ are uniformly labeled, then the modular product $\mathscr{G}_1 \star \mathscr{G}_2$ coincides with the "classical" modular product of G_1 and G_2 that has vertex set $V(G_1) \times V(G_2)$ and where, for any two *distinct* vertices $(u, v) \neq (u', v') \in V(G_1) \times V(G_2)$ with $u \neq u'$ and $v \neq v'$, $\{(u, v), (u', v')\}$ is an edge if and only if (0) $\{u, u'\} \notin E(G_1)$ and $\{v, v'\} \notin E(G_2)$, or (1) $\{u, u'\} \in E(G_1)$ and $\{v, v'\} \in E(G_2)$.

For $i \in \{1, \ldots, n\}$, let $p_i \colon V(\mathscr{G}) \to V(\mathscr{G}_i)$ be the coordinate projection, defined by $p_i(x) = x_i$ for $x = (x_1, \ldots, x_n) \in V(\mathscr{G})$. For a subgraph $K \subseteq \mathscr{G}$, set $p_i(K) \coloneqq \{p_i(x) \colon x \in V(K)\} \subseteq V(\mathscr{G}_i)$. The labeled subgraph of $\mathscr{G}_i$ induced by the vertex set $p_i(K)$, denoted $\mathscr{G}_i[p_i(K)]$, is called the *projection of K onto the i-th factor*.

Given a labeled graph $\mathscr{G} = (V, E, \lambda_V, \lambda_E)$, its labeled line graph $\mathscr{L}(\mathscr{G}) = (W, F, \lambda_W, \lambda_F)$ has vertex set $W \coloneqq E$ and edge set $F \coloneqq \{\{e, f\} \subseteq E \colon e \neq f,\ e \cap f \neq \emptyset\}$. For a vertex $e \in W$, we set the vertex label $\lambda_W(e) \coloneqq \lambda_E(e)$, and for an edge $\{e, f\} \in F$, where we always have $e \cap f = \{v\}$, we set $\lambda_F(\{e, f\}) \coloneqq \lambda_V(v)$. In this way, edge labels of $\mathscr{G}$ become vertex labels of $\mathscr{L}(\mathscr{G})$ and vertex labels of $\mathscr{G}$ become edge labels of $\mathscr{L}(\mathscr{G})$.

3 Finding maximum common subgraphs

As outlined above, all maximal (inclusion-wise) common induced subgraphs of G_1 and G_2 can be obtained, up to isomorphism, by enumerating all maximal cliques of $G_1 \star G_2$ and projecting them back to G_1 and G_2 [31]. We extend this idea to develop methods for solving the following tasks.

(1) Finding an MCIS for two or more graphs.

(2) Finding a *connected* MCIS for two or more graphs.

(3) Finding a (connected) MCES for two or more graphs.

(4) Handling the MCIS and MCES problems for vertex- and edge-labeled graphs on two or more graphs, where common subgraphs must respect the respective labels.

We now give an overview of how we address Tasks (1)–(4). In particular, we state the main theoretical results here; all proofs are deferred to Section 4.

Task (1). Suppose that $G = G_1 \star G_2$ is the "classical" modular product of two uniformly-labeled graphs G_1 and G_2. Each vertex (u, v) in a clique C of G corresponds to the vertex u in G_1 and the vertex v in G_2. Moreover, an edge $\{(u, v), (u', v')\}$ in C ensures that either both $\{u, u'\}$ and $\{v, v'\}$

are edges, or both are non-edges, in G_1 and G_2, respectively. Consequently, all maximal (inclusion-wise) common induced subgraphs of G_1 and G_2 (up to isomorphism) can be obtained by enumerating all maximal cliques in their modular product [31].

In general, we want to determine an MCIS of more than two graphs, say the graphs $G_1, G_2, \ldots, G_n$. A straightforward but naive way to achieve this goal follows from the following result.

Proposition 1. *Let $G_1, \ldots, G_n$ be graphs and let $G = \star_{i=1}^{n} G_i$ be their modular product. Let K be some clique in G and denote by $H_K := G_i[p_i(K)]$ the projection of K onto G_i for an arbitrary $i \in \{1, \ldots, n\}$. Then the following two statements are equivalent.*

1. *K is a maximum-sized clique in G.*
2. *H_K is an* MCIS *of $G_1, \ldots, G_n$.*

Proposition 1 implies that it suffices to find maximum cliques in the modular product $G := \star_{i=1}^{n} G_i$ to determine MCISs of $G_1, \ldots, G_n$, a problem that is known to be NP-hard [23]. Moreover, $|V(G)| = |V(G_1) \times \cdots \times V(G_n)| \geq 2^n$ always holds whenever all factors G_i have at least two vertices; making the search space for finding maximal or maximum cliques extremely large. Hence, although Proposition 1 provides an interesting structural characterization, it does not lead directly to an efficient method to solve the MCIS problem. Nevertheless, we will employ the ideas in Proposition 1 to design efficient methods that rely only on well-suited subgraphs of G in which we need to find maximal cliques, thereby reducing the search space significantly.

The basic idea of our method works as follows. We begin by computing all maximal cliques in $G_1 \star G_2$ and storing them in the set $\mathcal{K}^{(2)}$. For each $K \in \mathcal{K}^{(2)}$ and $1 \leq i \leq 2$, we define $H_K^i := G_i[p_i(K)]$ to be the projection of K onto G_i. By construction of the modular product, we have $H_K^1 \simeq H_K^2$ for every $K \in \mathcal{K}^{(2)}$. We then choose the representative $H_K := H_K^1$ for all $K \in \mathcal{K}^{(2)}$ and collect all such representatives in the set $\mathcal{H}_{1,2}$. Next, for each $H \in \mathcal{H}_{1,2}$, we repeat this process: we compute all maximal cliques in the modular product $H \star G_3$; collecting them over all $H \in \mathcal{H}_{1,2}$ yields a

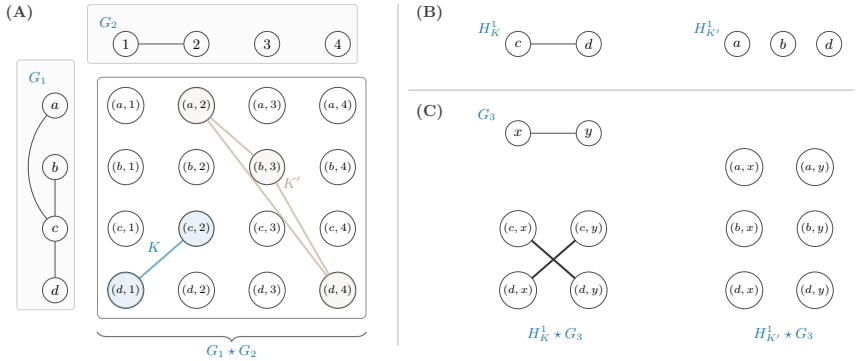

Figure 1. In panel (A), two maximal cliques K and K' in the modular product $G_1 \star G_2$ of the two graphs G_1 and G_2 are shown. For visual clarity, panel (A) does not display all edges of $G_1 \star G_2$; only the edges belonging to the highlighted cliques K and K' are drawn. Panel (B) shows the graphs $H^1_K, H^1_{K'} \in \mathcal{H}_{1,2}$, while panel (C) shows the modular products $H^1_K \star G_3$ and $H^1_{K'} \star G_3$; see Example 1.

new collection $\mathcal{K}^{(3)}$. Taking the representatives $G_1[p_1(K)]$ for all $K \in \mathcal{K}^{(3)}$ yields the set $\mathcal{H}_{1,2,3}$.

We illustrate the latter idea with the help of the following example.

Example 1. For most of the graphs used here, we refer to Figure 1. Consider the two graphs G_1, G_2 and their modular product $G_1 \star G_2$. One easily verifies that $G_1 \star G_2$ contains several maximal cliques. We have highlighted two of them in panel (A) of Figure 1: K, which is induced by the vertices $(d, 1)$ and $(c, 2)$, and K', which is induced by the vertices $(a, 2), (b, 3)$, and $(d, 4)$. In panel (B), we have shown $H^1_K = G_1[p_1(K)]$ and $H^1_{K'} = G_1[p_1(K')]$. These graphs are saved in the set $\mathcal{H}_{1,2}$. Note that only $H^1_{K'}$ leads to an MCIS of G_1 and G_2. However, we keep also the smaller common subgraph H^1_K in the set $\mathcal{H}_{1,2}$ to track potential MCISs of G_1 and G_2 and further graphs.

Indeed, if we consider a third graph G_3, we can take advantage of having stored the smaller subgraph H^1_K of G_1 and G_2, since the maximal cliques in $H^1_K \star G_3$ are larger than those in $H^1_{K'} \star G_3$ and lead to an MCIS of G_1, G_2, and G_3 that cannot be obtained using $H^1_{K'} \star G_3$; see panel (C) of Figure 1. This also illustrates why, in our recursive construction, it is

essential to retain *all* maximal cliques of the modular product $G_1 \star G_2$ rather than only maximum ones.

According to Proposition 1, we could have computed an MCIS by determining maximum cliques in $G = G_1 \star G_2 \star G_3$. Note that G has 32 vertices. In our approach, we first enumerate maximal cliques in the graph $G_1 \star G_2$ with 16 vertices, and afterwards determine maximal cliques in $H^1_K \star G_3$ (which has 4 vertices) and in $H^1_{K'} \star G_3$ (which has 6 vertices). This illustrates that the search space for determining maximal, and consequently maximum, cliques is significantly reduced, particularly when a large number of factors is involved.

We state here one of our main results, which shows that the aforementioned procedure can be used to solve the MCIS problem of unlabeled graphs.

Theorem 2. *Let $G_1, \ldots, G_n$ be graphs. Then $\mathcal{H}_{1,\ldots,n}$ contains, up to isomorphism, all* MCIS*s of $G_1, \ldots, G_n$.*

Task (2). For this task, we aim to find a *connected* MCIS of the graphs $G_1, G_2, \ldots, G_n$. Recall from Definition 1 that edges $\{(u, v), (u', v')\}$ in the "classical" modular product $G_1 \star G_2$ are formed either from edges $\{u, u'\} \in E(G_1)$ and $\{v, v'\} \in E(G_2)$, or from non-edges $\{u, u'\} \notin E(G_1)$ and $\{v, v'\} \notin E(G_2)$.

Let H be a subgraph of $G_1 \star G_2$. Clearly, H can contain both `type-0` and `type-1` edges. In [29], such edges are also called *c*-edges (connected) and *d*-edges (disconnected). If there exists a uv-path in H consisting of `type-1` edges only for every two distinct vertices u and v of H, we say that H is ***type-1** connected*. Note that, for each `type-1` edge $\{u, v\}$ of H, it holds that $\{p_i(u), p_i(v)\}$ is an edge in G_i, $i \in \{1, 2\}$. Suppose now that K is a `type-1` connected clique in $G_1 \star G_2$. Hence, for every u and v in K, there exists a uv-path in K whose edges are all of `type-1`. By the argument above, there is a corresponding $p_i(u)p_i(v)$-path in G_i, $i \in \{1, 2\}$. Therefore, in this case, $G_i[p_i(K)]$ is a connected (spanning on $p_i(K)$) subgraph of G_i for $i \in \{1, 2\}$. For example, for the two graphs G_1 and G_2 shown in Figure 1, the clique K is `type-1` connected, whereas K'

is not. Thus, if we restrict attention to connected MCISs, we would store H^1_K but not $H^1_{K'}$. This generalizes naturally to more than two factors.

Hence, restricting to maximal `type-1` connected cliques and selecting those of maximum cardinality yields exactly the connected MCISs of $G_1, \ldots, G_n$. A general version of this result, covering labeled graphs and more general connectivity constraints, is stated in Theorem 3 below.

Task (3). To solve the MCES problem, we employ the line graph $L(G) = (W, F)$ of a graph $G = (V, E)$, defined by setting $W \coloneqq E$ and $\{e, f\} \in F$ if and only if $e \neq f$ and $e \cap f \neq \emptyset$. It is straightforward to verify that if G is simple, then so is $L(G)$.

It is well-known that finding a (connected) MCES in G is equivalent to finding a (connected) MCIS in $L(G)$, with special handling required for claws and triangles; see, e.g., [29]. Using the methods from Tasks (1) and (2), we start with $L(G_1)$ and $L(G_2)$ and compute $L(G_1) \star L(G_2)$ to find all maximal (connected) common induced subgraphs. We then proceed similarly with $L(G_3), \ldots, L(G_n)$ to obtain all maximal (connected) common induced subgraphs of $L(G_1), \ldots, L(G_n)$, which correspond to all maximal (connected) common edge-induced subgraphs of $G_1, \ldots, G_n$.

However, the line-graph reduction suffers from a classical ambiguity [43]: three mutually adjacent vertices in a line graph may correspond either to a triangle or a claw in the original graph. To ensure an exact correspondence between cliques in the modular product of the line graphs and edge-induced common subgraphs of the original graphs, we restrict attention to a family of cliques that avoids such ambiguous configurations and, in particular, ensures that those cliques lead to a solution to the MCES problem, see Theorem 5. The following definition is stated for the general setting of labeled graphs and will later be used to treat the MCIS and MCES problems for such labeled graphs, where common subgraphs must respect the labels. Nevertheless, this definition carries over directly to unlabeled graphs by considering their uniformly-labeled versions.

Definition 3 (`Type-1` triangles and Δ-Y consistency and admissibility)**.** Let $\mathscr{G} = (G, \lambda_V, \lambda_E) \coloneqq \star_{i=1}^{n} \mathscr{G}_i$ be the modular product of labeled graphs $\mathscr{G}_1, \ldots, \mathscr{G}_n$. Then a triangle T in G is of `type-1` if all edges of T are `type-1`

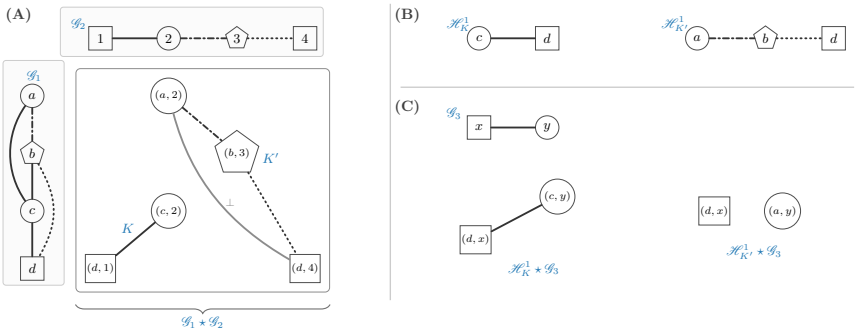

Figure 2. Shown are several labeled graphs whose vertex and edge labels are indicated by different vertex and line styles. In panel (A), the product $\mathcal{G}_1 \star \mathcal{G}_2$ contains exactly five vertices and four edges. The gray edge with label $\perp$ is a `type-0` edge in $\mathcal{G}_1 \star \mathcal{G}_2$, while all other edges in $\mathcal{G}_1 \star \mathcal{G}_2$ are `type-1`. It is easy to see that $\mathcal{G}_1 \star \mathcal{G}_2$ has two maximal cliques: K, which is induced by $(d, 1)$ and $(c, 2)$, and K' which is induced by $(a, 2)$, $(b, 3)$ and $(d, 4)$. The set $\mathcal{H}_{1,2} = \{\mathcal{H}^1_K, \mathcal{H}^1_{K'}\}$ consists of the labeled graphs obtained as projections onto the first factor, i.e., $\mathcal{H}^1_K = \mathcal{G}_1[p_1(K)]$ and $\mathcal{H}^1_{K'} = \mathcal{G}_1[p_1(K')]$; see panel (B). Note that $\mathcal{H}^1_K$ and $\mathcal{H}^1_{K'}$ are isomorphic to the labeled graphs obtained from K and K', respectively, by deleting all `type-0` edges. However, only $\mathcal{H}^1_{K'}$ leads to a labeled MCIS of $\mathcal{G}_1$ and $\mathcal{G}_2$. Nevertheless, if we now consider a third labeled graph $\mathcal{G}_3$ shown in panel (C), we use the set $\mathcal{H}_{1,2} = \{\mathcal{H}^1_K, \mathcal{H}^1_{K'}\}$ of all maximal common labeled subgraphs of $\mathcal{G}_1$ and $\mathcal{G}_2$ and compute $\mathcal{H}^1_K \star \mathcal{G}_3$ and $\mathcal{H}^1_{K'} \star \mathcal{G}_3$. The maximal labeled clique K'' in $\mathcal{H}^1_K \star \mathcal{G}_3$ then gives rise to a labeled MCIS of $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$, and we obtain $\mathcal{H}_{1,2,3} = \{\mathcal{H}^1_{K''}\}$, where $\mathcal{H}^1_{K''} = \mathcal{G}_1[p_1(K'')]$ is the subgraph of $\mathcal{G}_1$ induced by the vertices c and d.

edges of $\mathcal{G}$.

A `type-1` triangle $T = \{x, y, z\}$ in G is Δ-Y *consistent* if the subgraph of G_i induced by $\{p_i(x), p_i(y), p_i(z)\}$ is either a triangle (Δ) for all $i \in \{1, \ldots, n\}$ or a claw (Y) for all $i \in \{1, \ldots, n\}$. Otherwise, T is called *ambiguous*.

A clique C in G is Δ-Y *admissible* if all the `type-1` triangles contained in C are Δ-Y consistent.

Note that the problem of finding (connected) maximum common edge-induced subgraphs of $G_1, \ldots, G_n$ can be viewed as a special case of our labeled framework: we regard each G_i as a uniformly labeled graph and

apply the labeled modular product to their labeled line graphs $L(G_i)$, restricting the clique search to Δ-Y admissible cliques as above. In this way, the (unlabeled) MCES problem is reduced to a labeled MCIS problem on line graphs. The general labeled setting, as well as the precise correspondence guaranteed by Theorem 5, is discussed in more detail under Task (4) below.

Task (4). We address the MCIS and MCES problems for labeled graphs in order to track specific atom types (vertex labels) and bond types (edge labels).

Recall from Section 2 that labeled MCISs of labeled graphs are common induced subgraphs that are pairwise label-preserving isomorphic and have a maximum number of vertices. Moreover, a labeled subgraph $\mathscr{H} = (H, \lambda_V, \lambda_E) \subseteq \mathscr{G}$ is `type-`A connected if every two vertices of H are linked by a path whose edges have labels in A. Vertex labels are already aligned in the labeled modular product: a product vertex $(v_1, \ldots, v_n)$ exists only if $\lambda_{V_1}(v_1) = \cdots = \lambda_{V_n}(v_n)$. Thus, vertex-label agreement (atom types) along the factors is inherent in the labeled MCIS notion and requires no additional tracking.

The correspondence between cliques in the modular product $\star_{i=1}^{n}\mathscr{G}_i$ and the MCISs of $\mathscr{G}_1, \ldots, \mathscr{G}_n$ (as in Proposition 1) extends naturally to the labeled setting and to the `type-`A connectivity constraint.

Theorem 3. *Let $\mathscr{G} = \star_{i=1}^{n}\mathscr{G}_i$ be the modular product of the labeled graphs $\mathscr{G}_1, \ldots, \mathscr{G}_n$ whose edge labels are collected in the set Σ_E, and let $A \subseteq \Sigma_E$ be a set of edge labels. Moreover, let $\mathscr{K} \coloneqq (K = (V_K, E_K), \lambda_{V_K}, \lambda_{E_K})$ be a* `type-`A *connected labeled clique in $\mathscr{G}$. Fix some $i \in \{1, \ldots, n\}$, and let $\mathscr{H}_K$ be the labeled subgraph of $\mathscr{G}_i$ induced by the vertex set $\{p_i(w) \colon w \in V_K\}$, that is, $\mathscr{H}_K \coloneqq \mathscr{G}_i[p_i(V_K)]$. Then, the following two statements are equivalent:*

1. *$\mathscr{K}$ has the maximum number $|V_K|$ of vertices among all* `type-`A *connected cliques in $\mathscr{G}$.*
2. *$\mathscr{H}_K$ is a* `type-`A *connected labeled* MCIS *of $\mathscr{G}_1, \ldots, \mathscr{G}_n$.*

If $A = \Sigma_E$, then $\mathscr{H}_K$ is a connected labeled MCIS.

Suppose we want a type-A connected labeled MCIS of $\mathscr{G}_1, \ldots, \mathscr{G}_n$ for some set of labels A. As in the unlabeled case, we first compute all maximal type-A connected cliques in $\mathscr{G}_1 \star \mathscr{G}_2$, store them in the set $\mathcal{K}^{(2)}$, and derive the set $\mathcal{H}^A_{1,2}$ of all maximal common type-A connected labeled subgraphs $\mathscr{H}^1_K := \mathscr{G}_1[p_1(K)]$ of $\mathscr{G}_1$ and $\mathscr{G}_2$ for all $K \in \mathcal{K}^{(2)}$. For each $\mathscr{H} \in \mathcal{H}^A_{1,2}$ we then consider the modular product $\mathscr{H} \star \mathscr{G}_3$ and repeat this procedure to derive the set $\mathcal{H}^A_{1,2,3}$, and so on recursively, until we obtain the final set $\mathcal{H}^A_{1,\ldots,n}$.

The recursive construction in the labeled setting is illustrated in Figure 2, where panel (A) shows the initial labeled product, panel (B) the projected candidates, and panel (C) their subsequent products with $\mathscr{G}_3$.

Theorem 4. *Let $\mathscr{G}_1, \ldots, \mathscr{G}_n$ be labeled graphs and A be a set of edge labels. Then $\mathcal{H}^A_{1,\ldots,n}$ contains, up to isomorphism, all* type-A *connected* MCIS*s of $\mathscr{G}_1, \ldots, \mathscr{G}_n$.*

In the MCES (line-graph) setting, type-A paths in the product will correspond to paths in the original graphs whose intermediate vertices have labels in A as specified next. To recall, for a labeled graph $\mathscr{G} = (V, E, \lambda_V, \lambda_E)$, its labeled line graph is $\mathscr{L}(\mathscr{G}) = (E, F, \lambda_E, \lambda_F)$ where $F := \{\{e, f\} \subseteq E \colon e \neq f, e \cap f \neq \emptyset\}$ and $\lambda_F(\{e, f\}) := \lambda_V(v)$ with v being the vertex that is contained in both e and f. Thus, edge labels of $\mathscr{G}$ become vertex labels of $\mathscr{L}(\mathscr{G})$ and vertex labels of $\mathscr{G}$ become edge labels of $\mathscr{L}(\mathscr{G})$.

Definition 4 (Edge projections in the line-graph product)**.** Let $\mathscr{G}_1, \ldots, \mathscr{G}_n$ be labeled graphs and $\mathscr{G} = (G, \lambda_{V(G)}, \lambda_{E(G)}) := \star_{i=1}^n \mathscr{L}(\mathscr{G}_i)$ be the modular product of their labeled line graphs. For $i \in \{1, \ldots, n\}$, let $p_i : V(\mathscr{G}) \to V(\mathscr{L}(\mathscr{G}_i))$ be the coordinate projection as defined in Section 2. Then, for any subgraph $K \subseteq G$, it holds that $p_i(K) \subseteq E(\mathscr{G}_i)$ and we define $\mathscr{H}_i(K) := \mathscr{G}_i[p_i(K)]$.

Note that a labeled subgraph $\mathscr{H} = (H, \lambda_V, \lambda_E) \subseteq \mathscr{G} = \star_{i=1}^n \mathscr{L}(\mathscr{G}_i)$ is type-A connected if every two vertices of H are linked by a path whose edges have labels in A. Since each factor $\mathscr{L}(\mathscr{G}_i)$ is a labeled line graph, every edge $\{e, f\} \in E(\mathscr{L}(\mathscr{G}_i))$ represents the incidence of two distinct edges $e, f \in E(\mathscr{G}_i)$ and is labeled by the label of their unique common endpoint

in $\mathscr{G}_i$. Note that each vertex $x \in V(\mathscr{G})$ is of the form $x = (e_1, \ldots, e_n)$ with $e_i \in V(\mathscr{L}(\mathscr{G}_i)) = E(\mathscr{G}_i)$. Hence, if $\{x, y\} \in E(\mathscr{G})$ is a `type-1` product edge with $\lambda_E(\{x, y\}) = a \in A$, then for every $i \in \{1, \ldots, n\}$ the two edges $p_i(x), p_i(y) \in E(\mathscr{G}_i)$ are incident at a vertex v_i with $\lambda_{V_i}(v_i) = a$. Consequently, A-labeled paths in the product $\mathscr{G}$ project to chains of incident edges in the original graphs whose intermediate vertices have labels in A.

On the other hand, vertex labels in $\mathscr{L}(\mathscr{G}_i)$ are precisely the edge labels of $\mathscr{G}_i$, and these are already aligned in the labeled modular product: a product vertex $x = (e_1, \ldots, e_n)$ in $\mathscr{G}$ exists only if $\lambda_{E_1}(e_1) = \cdots = \lambda_{E_n}(e_n)$. Thus, edge-label agreement (bond types) along the factors is inherent in the labeled MCES notion and requires no additional tracking.

Theorem 5 (Δ-Y disambiguation for MCES via line graphs)**.** *Let $\mathscr{G}_1, \ldots, \mathscr{G}_n$ be labeled graphs whose edge labels are collected in the set Σ_E, and let $A \subseteq \Sigma_E$ be a set of edge labels. Moreover, let $\mathscr{G} \coloneqq \star_{i=1}^{n} \mathscr{L}(\mathscr{G}_i)$ be the modular product of their labeled line graphs. Let $K \subseteq V(\mathscr{G})$ be a* Δ-Y *admissible clique* (resp. a `type-`A connected Δ-Y admissible clique)*. Then the following two statements are equivalent.*

1. *K has maximum cardinality among all* Δ-Y *admissible cliques in $\mathscr{G}$* (resp. among all `type-`A connected Δ-Y admissible cliques in $\mathscr{G}$).

2. *For every $i \in \{1, \ldots, n\}$, the labeled graph $\mathscr{H}_i(K)$ is a* MCES *of $\mathscr{G}_1, \ldots, \mathscr{G}_n$* (resp. a `type-`$A$ connected MCES of $\mathscr{G}_1, \ldots, \mathscr{G}_n$).

In particular, Theorem 5 implies that, when we restrict the clique search in $\mathscr{G}$ to Δ-Y admissible cliques, then maximum cliques in $\mathscr{G}$ correspond exactly to maximum (connected) common edge-induced subgraphs of $\mathscr{G}_1, \ldots, \mathscr{G}_n$. Therefore, Task (4) is handled uniformly within the labeled modular product framework. For labeled MCISs we work directly in the product $\mathscr{G} = \star_{i=1}^{n} \mathscr{G}_i$, while for labeled MCESs we pass to the labeled line graphs $\mathscr{L}(\mathscr{G}_i)$ and restrict to Δ–Y admissible cliques satisfying the `type-`A connectivity constraint. Theorems 3 and 5 together guarantee that maximum (`type-`A connected) cliques in these products correspond exactly to the desired labeled MCISs and MCESs of the original graphs, without loss of any optimal solutions.

4 Theory

In this section we provide the formal proofs underlying the results stated in Section 3. We first establish the clique–MCIS correspondence for labeled graphs (subsuming the unlabeled case), then prove the claims underlying the recursive candidate-set construction under `type-`A connectivity, and finally address the line-graph formulation of MCES together with the Δ–Y disambiguation.

Proof of Theorem 3 and Proposition 1

For our proofs, we first require two additional technical results.

Lemma 1. *Let $\mathscr{G} = \star_{i=1}^{n} \mathscr{G}_i$ be the modular product of the labeled graphs $\mathscr{G}_1, \ldots, \mathscr{G}_n$ whose edge labels are collected in the set Σ_E. Moreover, let $A \subseteq \Sigma_E$ be a set of edge labels. Suppose that $\mathscr{H} = (U, F, \lambda_U, \lambda_F)$ is a labeled `type-`A connected* MCIS *of $\mathscr{G}_1, \ldots, \mathscr{G}_n$. Then, there exists a maximal `type-`A connected labeled clique $\mathscr{K} \coloneqq (K, \lambda_{V(K)}, \lambda_{E(K)})$ in $\mathscr{G}$ such that $|V(K)| = |U|$ and $\mathscr{G}_i[p_i(K)] \simeq \mathscr{H}$ for all $i \in \{1, \ldots, n\}$.*

Proof. Let $\mathscr{G}_i = (V_i, E_i, \lambda_{V_i}, \lambda_{E_i})$, $1 \leq i \leq n$, be labeled graphs, and let $\mathscr{G} = (V, E, \lambda_V, \lambda_E) = \star_{i=1}^{n} \mathscr{G}_i$ be their modular product. Suppose that $\mathscr{H} = (U, F, \lambda_U, \lambda_F)$ is a labeled `type-`A connected MCIS of $\mathscr{G}_1, \ldots, \mathscr{G}_n$. Thus, for each $i \in \{1, \ldots, n\}$ there is an injective map $\varphi_i \colon U \to V_i$ such that φ_i induces a label-preserving isomorphism between $\mathscr{H}$ and the labeled induced subgraph $\mathscr{G}_i[\varphi_i(U)]$.

To simplify writing, for each $u \in U$ and $i \in \{1, \ldots, n\}$ we write $u_i \coloneqq \varphi_i(u)$. Since φ_i is a label-preserving isomorphism, it must hold that $\lambda_{V_i}(u_i) = \lambda_U(u) = \lambda_{V_j}(u_j)$ for all $i, j \in \{1, \ldots, n\}$. This, in particular, implies that the vertex $(u_1, \ldots, u_n)$ exists in V for all $u \in U$. Therefore, the map $\kappa \colon U \to V$ that assigns to every vertex u of $\mathscr{H}$ the vertex $(u_1, \ldots, u_n)$ of $\mathscr{G}$ is well-defined. Let $W \coloneqq \{(u_1, \ldots, u_n) \colon u \in U\}$. Since φ_i is an injective map from U to V_i, the map κ is an injective map from U to V and, therefore, $|W| = |U|$. Define $K \coloneqq G[W]$, i.e., the induced subgraph of the underlying graph $G = (V, E)$ of $\mathscr{G}$ on W. Let $\lambda_{V(K)}$ and $\lambda_{E(K)}$ be the restrictions of λ_V and λ_E to $V(K) = W$ and

$E(K)$, respectively; that is, $\lambda_{V(K)}(v) \coloneqq \lambda_V(v)$ for all $v \in V(K)$ and $\lambda_{E(K)}(e) \coloneqq \lambda_E(e)$ for all $e \in E(K)$.

We show first that K is a clique in the underlying graph G of $\mathscr{G}$. Let $a, b \in U$ with $a \neq b$. Then $\kappa(a), \kappa(b) \in W = V(K)$. Write $\kappa(a) = (a_1, \ldots, a_n)$ and $\kappa(b) = (b_1, \ldots, b_n)$, where $a_i = \varphi_i(a)$ and $b_i = \varphi_i(b)$. Since each φ_i induces an isomorphism between the underlying graph (U, F) of $\mathscr{H}$ and the induced subgraph of (V_i, E_i) on $\varphi_i(U)$, we have $\{a, b\} \in F$ if and only if $\{a_i, b_i\} \in E_i$ for all i.

- If $\{a, b\} \in F$, then $\{a_i, b_i\} \in E_i$ for all i and, by label-preservation, $\lambda_{E_i}(\{a_i, b_i\}) = \lambda_F(\{a, b\})$ for all i. Hence $\{\kappa(a), \kappa(b)\} \in E$ is a type-1 product edge.

- If $\{a, b\} \notin F$, then $\{a_i, b_i\} \notin E_i$ for all i, hence $\{\kappa(a), \kappa(b)\} \in E$ is a type-0 product edge.

In either case, $\kappa(a)$ and $\kappa(b)$ are adjacent in G. Thus K is a clique in G. We next show that $\mathscr{K} = (K, \lambda_{V(K)}, \lambda_{E(K)})$ is `type-`A connected. Let $a, b \in U$ be distinct and consider the vertices $\kappa(a), \kappa(b) \in V(K)$. Since $\mathscr{H}$ is `type-`A connected, there exists an a–b path $a = u_0, u_1, \ldots, u_m = b$ in the underlying graph (U, F) such that $\lambda_F(\{u_{j-1}, u_j\}) \in A$ for all $j = 1, \ldots, m$. For each j, the edge $\{u_{j-1}, u_j\} \in F$ implies that $\{\kappa(u_{j-1}), \kappa(u_j)\}$ is a type-1 product edge in K. By label preservation,

$$\lambda_{E(K)}(\{\kappa(u_{j-1}), \kappa(u_j)\}) = \lambda_F(\{u_{j-1}, u_j\}) \in A.$$

Hence $\kappa(u_0), \kappa(u_1), \ldots, \kappa(u_m)$ is a path in K whose edges have labels in A. Therefore, $\mathscr{K}$ is `type-`A connected.

It remains to show that $\mathscr{K} = (K, \lambda_{V(K)}, \lambda_{E(K)})$ is a labeled `type-`A connected clique in $\mathscr{G}$ that is inclusion-maximal w.r.t. these properties. By construction, for each $i \in \{1, \ldots, n\}$ and each $u \in U$, we have $p_i(\kappa(u)) = u_i = \varphi_i(u)$. Hence $p_i(V(K)) = \varphi_i(U)$ and therefore $\mathscr{G}_i[p_i(K)] \simeq \mathscr{H}$ for every i. Assume, for contradiction, that $\mathscr{K}$ is not inclusion-maximal among labeled `type-`A connected cliques in $\mathscr{G}$. Then there exists a vertex $x \in V \setminus W$ such that $W' \coloneqq W \cup \{x\}$ induces a labeled `type-`A connected clique in $\mathscr{G}$. Let $K' \coloneqq G[W']$ and let $\mathscr{K}' \coloneqq (K', \lambda_{V(K')}, \lambda_{E(K')})$ be the

induced labeled subgraph of $\mathscr{G}$ on W'. Write $x = (x_1, \ldots, x_n)$. Since W' is a clique, x is adjacent to every vertex $\kappa(u) \in W$. Thus for each $u \in U$, the product edge $\{x, \kappa(u)\}$ is either type-0 (then $\lambda_E(\{x, \kappa(u)\}) = \perp$) or type-1 (then $\lambda_E(\{x, \kappa(u)\}) \in \Sigma_E$). Since $\mathscr{K}'$ is `type-`A connected and $A \subseteq \Sigma_E$, there exists $u^* \in U$ such that $\lambda_E(\{x, \kappa(u^*)\}) \in A$ (in particular, $\{x, \kappa(u^*)\}$ is type-1).

Now define a labeled graph $\mathscr{H}^x = (U^x, F^x, \lambda_{U^x}, \lambda_{F^x})$ as follows. Let $U^x := U \cup \{x^*\}$ where x^* is a new vertex. Define the vertex labeling λ_{U^x} by setting $\lambda_{U^x}(u) := \lambda_U(u)$ for all $u \in U$ and $\lambda_{U^x}(x^*) := \lambda_{V_1}(x_1)$. Let

$$F^x := F \cup \Big\{ \{u, x^*\} \;\Big|\; u \in U \text{ and } \lambda_E(\{x, \kappa(u)\}) \neq \perp \Big\}.$$

Define the edge labeling λ_{F^x} by setting $\lambda_{F^x}(f) := \lambda_F(f)$ for all $f \in F$, and for each $\{u, x^*\} \in F^x \setminus F$ put $\lambda_{F^x}(\{u, x^*\}) := \lambda_E(\{x, \kappa(u)\})$. In addition, for each $i \in \{1, \ldots, n\}$, we define $\psi_i \colon U^x \to V_i$ by $\psi_i(u) := \varphi_i(u)$ for $u \in U$ and $\psi_i(x^*) := x_i$. By construction, ψ_i induces a label-preserving isomorphism between $\mathscr{H}^x$ and the labeled induced subgraph $\mathscr{G}_i[\psi_i(U^x)]$. Hence $\mathscr{H}^x$ is a common labeled induced subgraph of $\mathscr{G}_1, \ldots, \mathscr{G}_n$.

Moreover, $\{u^*, x^*\} \in F^x$ and $\lambda_{F^x}(\{u^*, x^*\}) = \lambda_E(\{x, \kappa(u^*)\}) \in A$. Since $\mathscr{H}$ is `type-`A connected, every $u \in U$ has an A-labeled path to u^* in $\mathscr{H}$, and extending this path by the edge $\{u^*, x^*\}$ yields an A-labeled path to x^*. Thus $\mathscr{H}^x$ is `type-`A connected. Since $|U^x| = |U| + 1$, this contradicts that $\mathscr{H}$ is a labeled `type-`A connected MCIS of $\mathscr{G}_1, \ldots, \mathscr{G}_n$.

In summary, there exists a maximal `type-`A connected labeled clique $\mathscr{K} := (K, \lambda_{V(K)}, \lambda_{E(K)})$ in $\mathscr{G}$ such that $|V(K)| = |U|$ and $\mathscr{G}_i[p_i(K)] \simeq \mathscr{H}$ for all $i \in \{1, \ldots, n\}$. ■

Lemma 2. *Let $\mathscr{G} = \bigstar_{i=1}^{n} \mathscr{G}_i$ be the modular product of the labeled graphs $\mathscr{G}_1, \ldots, \mathscr{G}_n$ whose edge labels are collected in the set Σ_E. Moreover, let $A \subseteq \Sigma_E$ be a set of edge labels and suppose that $\mathscr{K} := (K = (V_K, E_K), \lambda_{V_K}, \lambda_{E_K})$ is a `type-`A connected labeled clique in $\mathscr{G}$. For each $i \in \{1, \ldots, n\}$, let $\mathscr{H}_i := \mathscr{G}_i[p_i(V_K)]$ be the labeled subgraph of $\mathscr{G}_i$ induced by the vertex set $p_i(V_K)$. Then, the graphs $\mathscr{H}_1, \ldots, \mathscr{H}_n$ are all `type-`A connected, pairwise label-preserving isomorphic, and each has $|V_K|$ vertices.*

Proof. Let $\mathscr{G}_i = (V_i, E_i, \lambda_{V_i}, \lambda_{E_i})$ for $i \in \{1, \ldots, n\}$ be labeled graphs and

let $\mathscr{G} = (V, E, \lambda_V, \lambda_E) = \star_{i=1}^{n} \mathscr{G}_i$ be their labeled modular product. Moreover, let $A \subseteq \Sigma_E$ and suppose that $\mathscr{K} \coloneqq (K = (V_K, E_K), \lambda_{V_K}, \lambda_{E_K})$ is a `type-`A connected labeled clique in $\mathscr{G}$.

Fix some $i \in \{1, \ldots, n\}$. We claim that the i-th coordinate projection $p_i \colon V_K \to V_i$ is injective. Let $x, y \in V_K$ be distinct. Since K is a clique, it follows that $\{x, y\} \in E_K$. By the definition of the labeled modular product, this is only possible for $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ if $x_k \neq y_k$ for all $k \in \{1, \ldots, n\}$.

Hence, $p_i(x) \neq p_i(y)$. Thus, $p_i \colon V_K \to V_i$ is injective. Therefore, $|V(H_i)| = |p_i(V_K)| = |V_K|$ where H_i is the underlying graph of $\mathscr{H}_i$, $1 \leq i \leq n$.

Now, fix some $i, j \in \{1, \ldots, n\}$ and define the map $\Phi_{ij} \colon p_i(V_K) \to p_j(V_K)$ by

$$\Phi_{ij}\big(p_i(x)\big) \coloneqq p_j(x) \text{ for all } x \in V_K.$$

This is well-defined because p_i is injective, so each vertex of $p_i(V_K)$ has a unique preimage in V_K. Moreover, Φ_{ji} is the inverse of Φ_{ij} and it follows that Φ_{ij} is a bijection.

We next verify that Φ_{ij} is a label-preserving graph isomorphism between the induced labeled subgraphs $\mathscr{H}_i = \mathscr{G}_i[p_i(V_K)]$ and $\mathscr{H}_j = \mathscr{G}_j[p_j(V_K)]$.

Let $v \in p_i(V_K)$. Since p_i is a map, there is a unique $x = (x_1, \ldots, x_n) \in V_K$ with $p_i(x) = v$. Hence, $v = x_i$ and $\Phi_{ij}(v) = x_j$. Since $x \in V(\mathscr{G})$, its coordinates have equal vertex labels by definition of the product vertex set, i.e., $\lambda_{V_i}(x_i) = \lambda_{V_j}(x_j)$, i.e., $\lambda_{V_i}(v) = \lambda_{V_j}(\Phi_{ij}(v))$.

Let $v, w \in p_i(V_K)$ be distinct, and let $x, y \in V_K$ be the unique vertices with $p_i(x) = v$ and $p_i(y) = w$. Since K is a clique in $\mathscr{G}$, we have $\{x, y\} \in E$. By the definition of the labeled modular product, exactly one of the following holds:

(0) $\{x_k, y_k\} \notin E_k$ for all $k \in \{1, \ldots, n\}$ (type-0 case), in which case $\lambda_E(\{x, y\}) = \bot$; or

(1) $\{x_k, y_k\} \in E_k$ for all $k \in \{1, \ldots, n\}$ and $\lambda_{E_1}(\{x_1, y_1\}) = \cdots = \lambda_{E_n}(\{x_n, y_n\})$ (type-1 case), in which case

$$\lambda_E(\{x, y\}) = \lambda_{E_k}(\{x_k, y_k\}) \in \Sigma_E \quad \text{for every } k.$$

In case (0), $\{v, w\} = \{x_i, y_i\} \notin E_i$, so v and w are non-adjacent in H_i, and likewise $\{\Phi_{ij}(v), \Phi_{ij}(w)\} = \{x_j, y_j\} \notin E_j$, so they are non-adjacent in H_j. In case (1), $\{v, w\} = \{x_i, y_i\} \in E_i$, so v and w are adjacent in H_i, and similarly $\{x_j, y_j\} \in E_j$, so $\Phi_{ij}(v)$ and $\Phi_{ij}(w)$ are adjacent in H_j. Moreover, in case (1) we also have equality of edge labels:

$$\begin{aligned}\lambda_{E_i}(\{v, w\}) &= \lambda_{E_i}(\{x_i, y_i\}) = \lambda_E(\{x, y\}) \\ &= \lambda_{E_j}(\{x_j, y_j\}) \\ &= \lambda_{E_j}(\{\Phi_{ij}(v), \Phi_{ij}(w)\}).\end{aligned}$$

Thus Φ_{ij} preserves adjacency and, whenever an edge exists, it preserves its label. Hence Φ_{ij} is a label-preserving isomorphism between $\mathscr{H}_i$ and $\mathscr{H}_j$. Since i, j were arbitrarily chosen, the graphs $\mathscr{H}_1, \ldots, \mathscr{H}_n$ are pairwise label-preserving isomorphic.

It remains to prove that each $\mathscr{H}_i$ is `type-`A connected. Fix $i \in \{1, \ldots, n\}$ and let $v, w \in V(H_i) = p_i(V_K)$ be arbitrary distinct vertices. Let $x, y \in V_K$ be the unique vertices with $p_i(x) = v$ and $p_i(y) = w$. Since $\mathscr{K}$ is `type-`A connected, there exists an x–y path $x = x_0, x_1, \ldots, x_m = y$ in K such that $\lambda_{E_K}(\{x_{t-1}, x_t\}) \in A$ for all $t = 1, \ldots, m$. Because λ_{E_K} is the restriction of the product edge-labeling λ_E, and since $A \subseteq \Sigma_E$, each edge $\{x_{t-1}, x_t\}$ must be a type-1 product edge (type-0 edges have label $\perp \notin \Sigma_E$). Therefore, for each t, the pair $\{p_i(x_{t-1}), p_i(x_t)\}$ is an edge of $\mathscr{G}_i$ and satisfies

$$\lambda_{E_i}\big(\{p_i(x_{t-1}), p_i(x_t)\}\big) = \lambda_E(\{x_{t-1}, x_t\}) = \lambda_{E_K}(\{x_{t-1}, x_t\}) \in A.$$

Hence $v = p_i(x_0), p_i(x_1), \ldots, p_i(x_m) = w$ is a v–w path in H_i whose edges all have labels in A. This shows that $\mathscr{H}_i$ is `type-`A connected. ■

Now we are ready to present the proof of Theorem 3.

Proof of Theorem 3. Let $\mathscr{G} = \star_{i=1}^{n} \mathscr{G}_i$ be the modular product of the labeled graphs $\mathscr{G}_1, \ldots, \mathscr{G}_n$ whose edge labels are collected in the set Σ_E, and let $A \subseteq \Sigma_E$. Let $\mathscr{K} \coloneqq (K = (V_K, E_K), \lambda_{V_K}, \lambda_{E_K})$ be a `type-`A connected labeled clique in $\mathscr{G}$. Now fix some $i \in \{1, \ldots, n\}$ and put $\mathscr{H}_K \coloneqq \mathscr{G}_i[p_i(V_K)]$.

Assume $\mathscr{K}$ has the maximum number $|V_K|$ of vertices among all type-A connected cliques in $\mathscr{G}$. Suppose, for contradiction, that $\mathscr{H}_K$ is *not* a type-A connected labeled MCIS of $\mathscr{G}_1, \ldots, \mathscr{G}_n$. Then there exists a type-A connected labeled common induced subgraph

$$\mathscr{H} = (U, F, \lambda_U, \lambda_F)$$

of $\mathscr{G}_1, \ldots, \mathscr{G}_n$ with $|U| > |V(\mathscr{H}_K)| = |V(K)|$. By Lemma 1, there exists a maximal type-A connected labeled clique $\mathscr{K}' := (K', \lambda_{V(K')}, \lambda_{E(K')})$ in $\mathscr{G}$ such that $|V(K')| = |U|$. Hence $|V(K')| = |U| > |V(K)|$, contradicting the assumption that $|V(K)|$ is maximum among all type-A connected cliques in $\mathscr{G}$. Therefore, $\mathscr{H}_K$ must be a type-A connected labeled MCIS of $\mathscr{G}_1, \ldots, \mathscr{G}_n$.

Assume now that $\mathscr{H}_K$ is a type-A connected labeled MCIS of $\mathscr{G}_1, \ldots, \mathscr{G}_n$. Let $\mathscr{K}' = (K', \lambda_{V(K')}, \lambda_{E(K')})$ be any type-A connected clique in $\mathscr{G}$. By Lemma 2, $\mathscr{H}_{K'} := \mathscr{G}_i[p_i(V(K'))]$ is a type-A connected labeled common induced subgraph of $\mathscr{G}_1, \ldots, \mathscr{G}_n$ such that $|V(\mathscr{H}_{K'})| = |V(K')|$.

Since $\mathscr{H}_K$ is maximum among all type-A connected labeled common induced subgraphs, we obtain

$$|V(K')| = |V(\mathscr{H}_{K'})| \leq |V(\mathscr{H}_K)| = |V(K)|.$$

Thus, $\mathscr{K}$ has the maximum number $|V_K|$ of vertices among all type-A connected cliques in $\mathscr{G}$.

Finally, if $A = \Sigma_E$, then every edge label occurring in any $\mathscr{G}_i$ lies in A, hence type-A connectedness in $\mathscr{G}_i$ is ordinary connectedness; therefore, in this case, $\mathscr{H}_K$ is a connected labeled MCIS. ■

Proof of Proposition 1. Proposition 1 is a direct consequence of Theorem 3 and the fact that there is a 1-to-1 correspondence between unlabeled graphs and their uniformly-labeled versions. ■

Proof of Theorems 2 and 4

For our proofs, we first require the following technical result.

Lemma 3. *Let $\mathscr{G}_1, \ldots, \mathscr{G}_n$ be labeled graphs whose edge labels are collected in the set Σ_E and let A be a set of labels. For each $t \in \{2, \ldots, n\}$, let $\mathscr{G}^{(t)} \coloneqq \star_{i=1}^{t} \mathscr{G}_i$ denote the modular product of the first t factors. Let $\mathcal{K}^{(t)}$ be the set of all maximal* `type-`A *connected labeled cliques in $\mathscr{G}^{(t)}$. Define the family of sets $\mathcal{H}^A_{1,\ldots,t}$ of induced labeled subgraphs of $\mathscr{G}_1$ recursively as follows.*

- *For $t = 2$, let $\mathcal{H}^A_{1,2} \coloneqq \{\mathscr{G}_1[p_1(K)] \colon \mathscr{K} = (K, \lambda_{V(K)}, \lambda_{E(K)}) \in \mathcal{K}^{(2)}\}$.*
- *For $t \geq 3$, consider $\mathcal{H}^A_{1,\ldots,t-1}$.*
  1. *Put $\mathcal{H}^A_{1,\ldots,t} \coloneqq \emptyset$.*
  2. *For all $\mathscr{H} \in \mathcal{H}^A_{1,\ldots,t-1}$, collect all maximal* `type-`A *connected labeled cliques $\mathscr{K} = (K, \lambda_{V(K)}, \lambda_{E(K)})$ in $\mathscr{H} \star \mathscr{G}_t$ and add the induced labeled subgraph $\mathscr{G}_1[p_1(K)]$ to $\mathcal{H}^A_{1,\ldots,t}$.*

Then, for every $t \in \{2, \ldots, n\}$ and every maximal `type-`A *connected labeled clique $\mathscr{K} = (K, \lambda_{V(K)}, \lambda_{E(K)}) \in \mathcal{K}^{(t)}$, the induced subgraph $\mathscr{G}_1[p_1(K)]$ belongs to $\mathcal{H}^A_{1,\ldots,t}$.*

Proof. Let $\mathscr{G}_1, \ldots, \mathscr{G}_n$, A, $\mathscr{G}^{(t)}$, $\mathcal{K}^{(t)}$ and $\mathcal{H}^A_{1,\ldots,t}$ be as stated. We proceed now by induction on t.

For the base case, assume that $t = 2$. By definition of $\mathcal{H}^A_{1,2}$, for each maximal `type-`A connected labeled clique $\mathscr{K} = (K, \lambda_{V(K)}, \lambda_{E(K)}) \in \mathcal{K}^{(2)}$ we explicitly include $\mathscr{G}_1[p_1(K)]$ in $\mathcal{H}^A_{1,2}$. Thus the claim holds for $t = 2$.

Assume now that the statement holds for some $t - 1 \geq 2$. Let $\mathscr{K} = (K, \lambda_{V(K)}, \lambda_{E(K)}) \in \mathcal{K}^{(t)}$ be a maximal `type-`A connected labeled clique in $\mathscr{G}^{(t)} = \star_{i=1}^{t} \mathscr{G}_i$. Write vertices of $\mathscr{G}^{(t)}$ as tuples $x = (x_1, \ldots, x_t)$ with $x_i \in V(\mathscr{G}_i)$. Define the set

$$V_{K'} \coloneqq \{(x_1, \ldots, x_{t-1}) \in V(\mathscr{G}^{(t-1)}) \mid (x_1, \ldots, x_{t-1}, x_t) \in V(K) \text{ for some } x_t\},$$

and let $\mathscr{K}' \coloneqq \mathscr{G}^{(t-1)}[V_{K'}]$. We now prove some claims that we use to establish the main result.

Claim 1. $\mathscr{K}'$ is a `type-`A *connected labeled clique in $\mathscr{G}^{(t-1)}$.* Let $a = (a_1, \ldots, a_{t-1})$ and $b = (b_1, \ldots, b_{t-1})$ be two distinct vertices in $V_{K'}$. By definition of $V_{K'}$, there exist a_t, b_t such that $\tilde{a} \coloneqq (a_1, \ldots, a_{t-1}, a_t)$ and $\tilde{b} \coloneqq$

$(b_1, \ldots, b_{t-1}, b_t)$ are vertices of K. Since K is a clique in $\mathscr{G}^{(t)}$, the vertices $\tilde{a}$ and $\tilde{b}$ are adjacent in $\mathscr{G}^{(t)}$. By the definition of the labeled modular product, this implies that for all $i \in \{1, \ldots, t\}$ either $\{a_i, b_i\} \notin E(\mathscr{G}_i)$ (type-0 case) or $\{a_i, b_i\} \in E(\mathscr{G}_i)$ and all corresponding edge labels coincide (type-1 case). In particular, the same alternative holds simultaneously for all $i \in \{1, \ldots, t-1\}$, which is precisely the adjacency condition in $\mathscr{G}^{(t-1)}$. Hence a and b are adjacent in $\mathscr{G}^{(t-1)}$, and thus $\mathscr{K}'$ is a labeled clique.

To see `type-`A connectedness, let $a, b \in V_{K'}$ and choose $\tilde{a}, \tilde{b} \in V(K)$ as above. Since $\mathscr{K}$ is `type-`A connected, there exists a $\tilde{a}$–$\tilde{b}$ path $\tilde{a} = x^{(0)}, x^{(1)}, \ldots, x^{(m)} = \tilde{b}$ in K such that $\lambda_{E(K)}(\{x^{(j-1)}, x^{(j)}\}) \in A$ for all $j = 1, \ldots, m$. Projecting each $x^{(j)} = (x_1^{(j)}, \ldots, x_t^{(j)})$ to its first $t-1$ coordinates yields a sequence in $V_{K'}$. Moreover, each edge $\{x^{(j-1)}, x^{(j)}\}$ is necessarily a type-1 product edge (since $A \subseteq \Sigma_E$), hence the projected vertices are adjacent in $\mathscr{G}^{(t-1)}$ by a type-1 edge carrying the same label in A. Therefore, the projected sequence is an a–b path in $\mathscr{K}'$ using only edges with labels in A. This shows that $\mathscr{K}'$ is `type-`A connected. This completes the proof of Claim 1.

Since, by Claim 1, $\mathscr{K}'$ is a `type-`A connected labeled clique in $\mathscr{G}^{(t-1)}$ and since $\mathscr{G}^{(t-1)}$ is finite, we can extend $\mathscr{K}'$ to an inclusion-maximal `type-`A connected labeled clique $\mathscr{Q} = (Q, \lambda_{V(Q)}, \lambda_{E(Q)}) \in \mathcal{K}^{(t-1)}$ with $V_{K'} \subseteq V(Q)$. By the induction hypothesis, the induced labeled subgraph $\mathscr{H} := \mathscr{G}_1[p_1(Q)]$ belongs to $\mathcal{H}^A_{1,\ldots,t-1}$. In particular, $\mathscr{H}$ is among the graphs that are processed in step (2) of the recursive construction of $\mathcal{H}^A_{1,\ldots,t}$.

Claim 2. The set $C := \{(x_1, x_t) \mid x = (x_1, \ldots, x_t) \in V(K)\}$ *induces a* `type-`A *connected labeled clique in* $\mathscr{H} \star \mathscr{G}_t$.

First note that for each $x \in V(K)$ we have $(x_1, \ldots, x_{t-1}) \in V_{K'} \subseteq V(Q)$, hence $x_1 \in p_1(Q) = V(\mathscr{H})$. Moreover, since $x \in V(\mathscr{G}^{(t)})$, all its coordinates have the same vertex label, in particular $\lambda_{V_1}(x_1) = \lambda_{V_t}(x_t)$ (and $\mathscr{H}$ inherits vertex labels from $\mathscr{G}_1$), so (x_1, x_t) is indeed a vertex of $\mathscr{H} \star \mathscr{G}_t$.

Let $(x_1, x_t), (y_1, y_t) \in C$ be distinct, and choose $x, y \in V(K)$ with these projections. Since K is a clique, x and y are adjacent in $\mathscr{G}^{(t)}$ and hence, in particular, the pair $\{x_1, y_1\}$ is an edge in $\mathscr{G}_1$ if and only if $\{x_t, y_t\}$ is an edge in $\mathscr{G}_t$, and in the edge case the labels coincide. As $\mathscr{H}$ is an induced

labeled subgraph of $\mathscr{G}_1$, the same equivalence holds with $\mathscr{H}$ in place of $\mathscr{G}_1$. Therefore, by the definition of the labeled modular product, (x_1, x_t) and (y_1, y_t) are adjacent in $\mathscr{H} \star \mathscr{G}_t$. Thus C is a clique.

For `type-`A connectedness, let $(x_1, x_t), (y_1, y_t) \in C$ and choose $x, y \in V(K)$ as above. Take an A-labeled path $x = x^{(0)}, \ldots, x^{(m)} = y$ in K. Then $(x_1^{(j)}, x_t^{(j)}) \in C$ for all j, and each edge $\{x^{(j-1)}, x^{(j)}\}$ has label in A and is type-1, hence $\{(x_1^{(j-1)}, x_t^{(j-1)}), (x_1^{(j)}, x_t^{(j)})\}$ is a type-1 product edge in $\mathscr{H} \star \mathscr{G}_t$ with label in A. This yields an A-labeled path in the clique induced by C. Hence C is `type-`A connected. This completes the proof of Claim 2.

By Claim 2, C induces a `type-`A connected labeled clique in $\mathscr{H} \star \mathscr{G}_t$. To ensure that this clique is generated in step (2) of the recursive construction of $\mathcal{H}^A_{1,\ldots,t}$, we still need inclusion-maximality, which is established in Claim 3.

Claim 3. C is inclusion-maximal among `type-`A connected cliques in $\mathscr{H} \star \mathscr{G}_t$.

Suppose, for contradiction, that there exists a vertex $(u_1, z) \in V(\mathscr{H} \star \mathscr{G}_t) \setminus C$ such that $C \cup \{(u_1, z)\}$ induces a `type-`A connected clique in $\mathscr{H} \star \mathscr{G}_t$. Since $(u_1, z) \in V(\mathscr{H} \star \mathscr{G}_t)$, we have $u_1 \in V(\mathscr{H}) = p_1(Q)$. As Q is a clique, there is a unique vertex $q(u_1) = (u_1, u_2, \ldots, u_{t-1}) \in V(Q)$ with first coordinate u_1. Define $z^\star := (u_1, u_2, \ldots, u_{t-1}, z)$.

We first note that $z^\star \in V(\mathscr{G}^{(t)})$: indeed, $q(u_1) \in V(\mathscr{G}^{(t-1)})$ implies $\lambda_{V_1}(u_1) = \cdots = \lambda_{V_{t-1}}(u_{t-1})$, and $(u_1, z) \in V(\mathscr{H} \star \mathscr{G}_t)$ implies $\lambda_{V_1}(u_1) = \lambda_{V_t}(z)$ (since $\mathscr{H}$ inherits vertex labels from $\mathscr{G}_1$). Hence all coordinates of $z^\star$ share the same vertex label.

Next, we show that $z^\star$ is adjacent in $\mathscr{G}^{(t)}$ to every vertex $x \in V(K)$. Write $x = (x_1, \ldots, x_t)$, and fix such an x. Since $C \cup \{(u_1, z)\}$ is a clique in $\mathscr{H} \star \mathscr{G}_t$, the vertex (u_1, z) is adjacent to (x_1, x_t). Thus, by the product definition, either $\{u_1, x_1\}$ and $\{z, x_t\}$ are both non-edges, or both edges with equal labels. On the other hand, since $(x_1, \ldots, x_{t-1}) \in V_{K'} \subseteq V(Q)$ and $q(u_1) \in V(Q)$, and Q is a clique, the same alternative (non-edge vs. edge-with-common-label) holds simultaneously for all pairs $\{u_i, x_i\}$ with $i \in \{1, \ldots, t-1\}$, and in the edge case their labels all coincide with the label of $\{u_1, x_1\}$. Combining these facts shows that, across all $i \in \{1, \ldots, t\}$,

either all $\{u_i, x_i\}$ are non-edges, or all are edges and all labels coincide; hence $z^\star$ is adjacent to x in $\mathscr{G}^{(t)}$.

Finally, because $C \cup \{(u_1, z)\}$ is `type-`A connected, there exists an A-labeled path from (u_1, z) to some vertex of C. In particular, the first step of such a path yields a vertex $(y_1, y_t) \in C$ with $\lambda_E\big(\{(u_1, z), (y_1, y_t)\}\big) \in A$. Let $y = (y_1, \ldots, y_t) \in V(K)$ be the corresponding vertex. Then the previous adjacency argument applies in the type-1 case, and shows that $\{z^\star, y\}$ is a type-1 product edge in $\mathscr{G}^{(t)}$ with label in A. Since $\mathscr{K}$ is `type-`A connected, it follows that $V(K) \cup \{z^\star\}$ induces a `type-`A connected clique in $\mathscr{G}^{(t)}$, contradicting the maximality of $\mathscr{K}$. Hence no such vertex (u_1, z) exists, and therefore C is inclusion-maximal among `type-`A connected cliques in $\mathscr{H} \star \mathscr{G}_t$. This completes the proof of Claim 3.

Combining Claims 2 and 3, we conclude that C is a maximal `type-`A connected labeled clique in $\mathscr{H} \star \mathscr{G}_t$. Since $\mathscr{H} \in \mathcal{H}^A_{1,\ldots,t-1}$, step (2) of the recursive construction of $\mathcal{H}^A_{1,\ldots,t}$ therefore adds the induced labeled subgraph $\mathscr{G}_1[p_1(C)]$ to $\mathcal{H}^A_{1,\ldots,t}$. By definition of C we have $p_1(C) = \{x_1 : (x_1, x_t) \in C\} = \{x_1 : (x_1, \ldots, x_t) \in V(K)\} = p_1(K)$, and therefore $\mathscr{G}_1[p_1(K)] \in \mathcal{H}^A_{1,\ldots,t}$, as required. ∎

Proof of Theorem 4. Let $\mathscr{H}^\star = (U^\star, F^\star, \lambda_{U^\star}, \lambda_{F^\star})$ be a `type-`A connected labeled MCIS of $\mathscr{G}_1, \ldots, \mathscr{G}_n$. By Lemma 1 (applied with n factors), there exists a maximal `type-`A connected labeled clique

$$\mathscr{K}^\star = (K^\star, \lambda_{V(K^\star)}, \lambda_{E(K^\star)}) \subseteq \star_{i=1}^n \mathscr{G}_i$$

such that $|V(K^\star)| = |U^\star|$ and $\mathscr{G}_i[p_i(K^\star)] \simeq \mathscr{H}^\star$ for all $i \in \{1, \ldots, n\}$. In particular, $\mathscr{G}_1[p_1(K^\star)] \simeq \mathscr{H}^\star$.

Now apply Lemma 3 with $t = n$. Since $\mathscr{K}^\star \in \mathcal{K}^{(n)}$ is a maximal `type-`A connected labeled clique in $\mathscr{G}^{(n)} = \star_{i=1}^n \mathscr{G}_i$, Lemma 3 implies that $\mathscr{G}_1[p_1(K^\star)]$ belongs to $\mathcal{H}^A_{1,\ldots,n}$. Therefore, $\mathcal{H}^A_{1,\ldots,n}$ contains an element that is label-preserving isomorphic to $\mathscr{H}^\star$. Since $\mathscr{H}^\star$ was an arbitrary `type-`A connected labeled MCIS, the claim follows. ∎

Proof of Theorem 2. Theorem 2 is a direct consequence of Theorem 4 and the fact that there is a 1-to-1 correspondence between unlabeled graphs

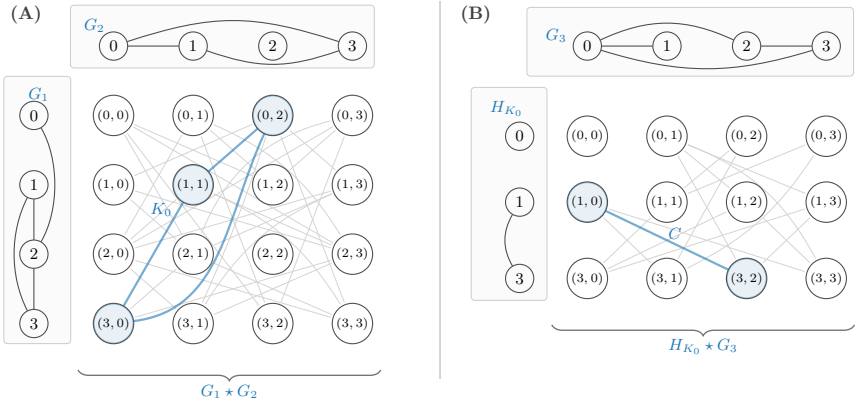

Figure 3. Example illustrating that a maximal clique obtained by the recursive construction need not be maximal in the full ternary modular product $\star_{i=1}^{3} G_i$. Starting from a maximal clique K_0 in $G_1 \star G_2$ we obtain the common subgraph $H_{K_0} = G_1[p_1(K_0)]$. In the product $H_{K_0} \star G_3$ there is a maximal clique C which, under the natural lifting, corresponds to a clique $K = \{(3,0,2),(1,1,0)\}$ in $\star_{i=1}^{3} G_i$ that is strictly contained in a larger clique $K' = \{(3,0,2),(1,1,0),(2,3,3)\}$, and hence K is not maximal in the full ternary product.

and their uniformly-labeled versions. ■

Remark. Although Lemma 3 shows that every maximal clique of the t-ary modular product $\star_{i=1}^{t} G_i$ is represented by the recursive construction (i.e., belongs to the family $\mathcal{H}1, \ldots, t$ via its projection to G_1), the converse does not hold in general. Every maximal clique encountered by the recursive procedure in an intermediate product $H \star G_t$ canonically lifts to a clique of the full t-ary product $\star i = 1^t G_i$, but this lifted clique need not be maximal there. Figure 3 illustrates this phenomenon for $t = 3$, where a maximal clique C in $H_{K_0} \star G_3$ lifts to a clique $K \subseteq \star_{i=1}^{3} G_i$ that is strictly contained in a larger clique K'.

Proof of Theorem 5

Proof of Theorem 5. Let $\mathscr{G} := \star_{i=1}^{n} \mathscr{L}(\mathscr{G}_i)$ and let $K \subseteq V(\mathscr{G})$ be a Δ-Y admissible clique (resp. a type-A connected Δ-Y admissible clique). For $i \in \{1, \ldots, n\}$ set $E_i := E_i(K) = p_i(K) \subseteq E(\mathscr{G}_i)$ and $\mathscr{H}_i := \mathscr{H}_i(K) =$

$\mathscr{G}_i[E_i]$ as in Definition 4.

Step 1: From an admissible clique to a common edge-induced subgraph. Arguing as in the proof of Lemma 2 (applied to the factors $\mathscr{L}(\mathscr{G}_i)$), each projection p_i is injective on K, and for any i, j the map $\Phi_{ij} \colon E_i \to E_j$ defined by $\Phi_{ij}(p_i(x)) \coloneqq p_j(x)$ is a label-preserving isomorphism between the induced labeled subgraphs $\mathscr{L}(\mathscr{G}_i)[E_i]$ and $\mathscr{L}(\mathscr{G}_j)[E_j]$. Using $\mathscr{L}(\mathscr{G}_i[E_i]) = \mathscr{L}(\mathscr{G}_i)[E_i]$, we obtain label-preserving isomorphisms

$$\mathscr{L}(\mathscr{H}_i) \simeq \mathscr{L}(\mathscr{H}_j) \qquad \text{for all } i, j \in \{1, \dots, n\}.$$

It is well-known [43] that a (simple) graph is determined by its line graph up to the unique triangle/claw ambiguity; cf. the discussion under Task (3) in Section 3. The only obstruction to lifting a line-graph isomorphism to an isomorphism of the underlying graphs is therefore the case that one factor realizes a 3-edge configuration as a triangle while another realizes the corresponding 3 edges as a claw. In our setting, any triple of pairwise incident edges in some $\mathscr{H}_i$ corresponds (via injectivity of p_i) to a `type-1` triangle in K. Since K is Δ-Y admissible, every such triangle is Δ-Y consistent (Definition 3), and hence the triangle/claw choice is the same in every factor. Thus the above line-graph isomorphisms lift to label-preserving isomorphisms $\mathscr{H}_i \simeq \mathscr{H}_j$ for all i, j, i.e. $\mathscr{H}_1, \dots, \mathscr{H}_n$ are pairwise label-preserving isomorphic edge-induced subgraphs. Moreover, since p_i is injective we have $|E(\mathscr{H}_i)| = |E_i| = |K|$ for every i.

If K is additionally `type-`A connected, then for any $x, y \in K$ there is a path $x = x_0, \dots, x_m = y$ in K such that every edge $\{x_{t-1}, x_t\}$ is a `type-1` edge with label in A. Projecting to coordinate i yields a path in $\mathscr{L}(\mathscr{H}_i)$ whose edge labels all lie in A. By the definition of labeled line graphs, these edge labels are precisely the labels of the intermediate vertices at which consecutive edges of $\mathscr{H}_i$ meet. Hence $\mathscr{H}_i$ satisfies the required `type-`A connectivity constraint (equivalently, $\mathscr{L}(\mathscr{H}_i)$ is `type-`A connected) for every i.

Step 2: From a common edge-induced subgraph to an admissible clique. Conversely, let $\mathscr{H}$ be a labeled common edge-induced subgraph of $\mathscr{G}_1, \dots, \mathscr{G}_n$ (resp. satisfying the `type-`A connectivity constraint), and for each i

fix a label-preserving isomorphism $\psi_i \colon \mathscr{H} \to \mathscr{G}_i[E_i']$ onto an edge-induced labeled subgraph. For an edge $e = \{u, v\} \in E(\mathscr{H})$ we write $\psi_i(e) \coloneqq \{\psi_i(u), \psi_i(v)\}$. For each $e \in E(\mathscr{H})$ define the tuple $x_e \coloneqq (\psi_1(e), \ldots, \psi_n(e))$.

Since ψ_i preserves edge labels and vertices of $\mathscr{L}(\mathscr{G}_i)$ are labeled by edge labels of $\mathscr{G}_i$, the tuple x_e is a vertex of $\mathscr{G}$. Set $K_{\mathscr{H}} \coloneqq \{x_e : e \in E(\mathscr{H})\}$.

For distinct $e, f \in E(\mathscr{H})$, if e and f are incident in $\mathscr{H}$, then $\psi_i(e)$ and $\psi_i(f)$ are incident in every factor $\mathscr{G}_i$, hence adjacent in $\mathscr{L}(\mathscr{G}_i)$ for every i, and the corresponding line-graph edge labels agree across all i because the shared endpoint vertex label is preserved. Thus $\{x_e, x_f\}$ is a `type-1` edge in $\mathscr{G}$. If e and f are not incident in $\mathscr{H}$, then $\psi_i(e)$ and $\psi_i(f)$ are non-incident for every i, hence non-adjacent in every $\mathscr{L}(\mathscr{G}_i)$, and $\{x_e, x_f\}$ is a `type-0` edge. Therefore $K_{\mathscr{H}}$ is a clique.

Moreover, every `type-1` triangle in $K_{\mathscr{H}}$ arises from three pairwise incident edges of $\mathscr{H}$, which induce the same 3-edge pattern (triangle or claw) in each image $\psi_i(\mathscr{H})$; hence $K_{\mathscr{H}}$ is Δ-Y admissible. If $\mathscr{H}$ satisfies the `type-`A connectivity constraint, then $\mathscr{L}(\mathscr{H})$ admits A-labeled paths between any two of its vertices, and the same projection argument shows that $K_{\mathscr{H}}$ is `type-`A connected. Finally, by construction $|K_{\mathscr{H}}| = |E(\mathscr{H})|$.

We are now in a position to prove the equivalence between Statements (1) and (2) in Theorem 5.

Assume Statement (1) holds; that is, K has maximum cardinality among all Δ-Y admissible cliques in $\mathscr{G}$ (resp. among all `type-`A connected Δ-Y admissible cliques in $\mathscr{G}$). Assume, for contradiction, that $\mathscr{H}_i(K)$ is not a MCES for some $i \in \{1, \ldots, n\}$. Then, there exists a labeled common edge-induced subgraph $\mathscr{H}$ with $|E(\mathscr{H})| > |E(\mathscr{H}_i(K))| = |K|$. By Step 2, $\mathscr{H}$ yields a Δ-Y admissible clique $K_{\mathscr{H}}$ with $|K_{\mathscr{H}}| = |E(\mathscr{H})| > |K|$ (and `type-`A connected if required), contradicting the maximality of K. Hence Statement (2) holds.

Conversely, assume Statement (2) holds; that is, for every $i \in \{1, \ldots, n\}$ the labeled graph $\mathscr{H}_i(K)$ is a MCES of $\mathscr{G}_1, \ldots, \mathscr{G}_n$ (resp. a `type-`A connected MCES of $\mathscr{G}_1, \ldots, \mathscr{G}_n$). For any Δ-Y admissible clique K' (resp. `type-`A connected), Step 1 yields a labeled common edge-induced subgraph $\mathscr{H}_i(K')$ with $|E(\mathscr{H}_i(K'))| = |K'|$. Since $\mathscr{H}_i(K)$ is a MCES (resp. a

type-A connected MCES), we have $|K'| = |E(\mathscr{H}_i(K'))| \leq |E(\mathscr{H}_i(K))| = |K|$. Thus K has maximum cardinality among all admissible cliques (resp. among all type-A connected admissible cliques), i.e. Statement (1) holds. ■

5 Methods

Here, we describe the algorithmic choices that ensure correctness and efficiency in solving the problem. All methods are implemented in C++ using the *Boost Graph Library (BGL)* and *Open Babel*[†]. The implementation and experiment scripts are publicly available at https://github.com/johannesborg/cmces.

5.1 Pruning steps

The methods described above implicitly define a search tree whose leaves are candidate maximum common subgraphs. Many branches of this search tree can be pruned by a depth-first search that maintains the best solution found so far and backtracks when a partial candidate cannot reach that size. To recall, $\mathcal{H}_{1,\dots,j}$ denotes the candidate set of the MCIS of the first j graphs $G_1, \dots, G_j$. Here, we first compute $\mathcal{H}_{1,2}$ as outlined in Section 3. Afterwards, for a fixed $H \in \mathcal{H}_{1,2}$, we compute candidates only for $H \star G_3$, yielding a subset $\mathcal{H}'_{1,2,3} \subseteq \mathcal{H}_{1,2,3}$, and continue recursively to obtain $\mathcal{H}'_{1,\dots,n} \subseteq \mathcal{H}_{1,\dots,n}$. Let $m \coloneqq \max\{|V(H')| \mid H' \in \mathcal{H}'_{1,\dots,n}\}$. We can then remove all elements $H \in \mathcal{H}_{1,2}$ for which $|V(H)| < m$, which can greatly reduce the number of candidates on which we need to recurse. This pruning is safe because extending a candidate can never increase its attainable size. When we intersect a candidate $H \in \mathcal{H}_{1,\dots,j}$ with a new factor G_{j+1}, any clique K found in $H \star G_{j+1}$ projects injectively back to H, so $|K| = |p_1(K)| \leq |V(H)|$. Consequently, once a full solution of size m has been witnessed, every candidate H with $|V(H)| < m$ can be discarded (no continuation of H can reach size m).

[†]Boost Graph Library: https://www.boost.org/libs/graph/; Open Babel: https://openbabel.org/.

5.2 Techniques for finding type-A connected cliques

The Bron–Kerbosch algorithm [8] is a clique-finding algorithm that iteratively extends non-maximal cliques until they can be certified as maximal, at which point they are returned. Based on variants of the Bron–Kerbosch algorithm, Koch's algorithm (Alg. 3 in [29]) can be used to find all connected maximal subgraphs of two graphs. We extend both algorithms to find type-A connected cliques, and consequently, type-A connected MCIS and MCES in multiple graphs.

In the type-A connected MCIS and MCES problems, we seek not just maximal cliques in $\star_{i=1}^{n} G_i$ and $\star_{i=1}^{n} L(G_i)$, but specifically maximal type-A connected cliques. We illustrate the key ideas using the MCIS problem on $\star_{i=1}^{n} G_i$, as the same approach naturally extends to $\star_{i=1}^{n} L(G_i)$ for the MCES problem.

At each step j, we take candidates $H \in \mathcal{H}_{1,\ldots,j-1}$, compute $G = H \star G_j$, and identify all type-A connected cliques in G. Since such cliques are contained within type-A connected subgraphs of G, we first partition G into its type-A connected components $G'_1, \ldots, G'_k$ by retaining only type-A edges. We then augment each G'_i by restoring all non-type-A edges, yielding induced subgraphs $G''_1, \ldots, G''_k$ that remain type-A connected. To find all type-A connected maximal common subgraphs of H and G_j, it suffices to compute the maximal type-A connected cliques in each G''_i separately, enabling parallel processing.

As outlined at the end of Section 3, for finding cliques in A-labeled products, we relabel "consistent" type-A edges as type-1 edges, relabel all edges in the product referring to non-edges in the factors as type-0 edges, and remove all remaining edges. To further optimize runtime, we remove additional type-0 edges in $G = H \star G_j$ before determining the type-A connected components $G''_1, \ldots, G''_k$. Specifically, we remove a type-0 edge $\{(u, v), (u', v')\}$ from G whenever there exists no *simple* type-A uu'-path in H that is isomorphic to a simple type-A vv'-path in G_j. Those particular type-0 edges $\{(u, v), (u', v')\}$ indicate that any type-A connected subgraph of H containing u and u' cannot be isomorphic to any type-A connected subgraph of G_j containing v and v', meaning (u, v) and (u', v') cannot be part of the same type-A connected clique corresponding to a

maximal type-A connected common subgraph of H and G_j. When partitioning G into type-A connected components $G'_1, \ldots, G'_k$ (as outlined in the preceding part), we include only "consistent" type-A edges. Removal of the additional type-0 edges as explained here yields a finer partition $G''_1, \ldots, G''_{k'}$. We then compute maximal type-A connected cliques in each G''_i. Although these type-0 edge-removal operations are based on computationally hard problems, Section 6 shows that they significantly speed up clique detection, likely due to the moderate size of the molecular graphs under consideration.

Determining which type-0 edges can be safely removed by the above path-based criterion is NP-hard (by reduction from the Hamiltonian path problem [37]). Typically, this does not pose a problem in the domain of molecular graphs, but we have encountered instances where it becomes problematic, in which case it is better to perform only a partial pruning of the modular product.

Algorithmic workflow. At a high level, the exact method proceeds iteratively. The input graphs are first ordered by a similarity-based heuristic to reduce the size of intermediate search problems. For the current reduction step, we then construct the appropriate modular product, directly for MCIS and on labeled line graphs for MCES, partition it into type-A connected components, optionally remove selected type-0 edges, and enumerate all maximal type-A connected cliques using the modified Bron–Kerbosch procedure. These cliques are projected back to common subgraphs of the first factor and retained as intermediate candidates, which are recursively intersected with the remaining input graphs until all optimal solutions have been identified. Algorithm 1 summarizes this exact workflow.

The pseudocode for the described algorithm is presented in Algorithm 1. The function ***PRUNE*** takes a modular product graph G as input, partitions it into its type-A connected components, prunes each component individually, and returns the resulting set of graphs $\{G'_1, \ldots, G'_k\}$. The function ***FIND-CLIQUES*** also takes a modular product graph G as input and returns all type-A connected maximal cliques in G. The function

Algorithm 1: Pseudocode for MAXIMUM-COMMON-SUBGRAPH($\{G_1, G_2, \ldots, G_n\}$)

```
Input: {G_1, G_2, ..., G_n}: A list of n labeled graphs
Output: All maximum common subgraphs of the input graphs
if n = 2 then
    G ← G_1 ⋆ G_2;
    {G'_1, ..., G'_k} ← PRUNE(G);
    foreach G'_i ∈ {G'_1, ..., G'_k} do
        foreach C ∈ FIND-CLIQUES(G'_i) do
            G_C ← PROJECT(C, G_1);
            if |V(G_C)| = MAXIMUM then
                SOLUTIONS ← SOLUTIONS ∪ {G_C};
            end
            if |V(G_C)| > MAXIMUM then
                SOLUTIONS ← {G_C};
                MAXIMUM ← |V(G_C)|;
            end
        end
    end
end
else
    G ← G_1 ⋆ G_2;
    {G'_1, ..., G'_k} ← PRUNE(G);
    foreach G'_i ∈ {G'_1, ..., G'_k} do
        foreach C ∈ FIND-CLIQUES(G'_i) do
            G_C ← PROJECT(C, G_1);
            if |V(G_C)| ≥ MAXIMUM then
                MAXIMUM-COMMON-
                  SUBGRAPH({G_C, G_3, ..., G_n});
            end
        end
    end
end
```

PROJECT takes as input a clique C in a modular product and its first factor graph G_1, and returns the projection of C onto G_1. Prior to calling Algorithm 1, the variable ***MAXIMUM*** is initialized to zero, and ***SOLUTIONS*** is initialized to the empty set. After execution, ***SOLUTIONS*** will contain all maximum common subgraphs of the input graphs.

5.3 Handling triangles and claws

Another step in our method handles special cases when finding the MCES of $G_1, \ldots, G_n$ via maximal cliques in $\star_{i=1}^{n} L(G_i)$. This issue, known in this domain as the Δ-Y exchange problem [18, 35], arises due to structural ambiguities in line graphs.

Let K_3 denote the complete graph on three vertices (a triangle) and $K_{1,3}$ the claw graph, consisting of four vertices and three edges meeting at a single vertex. It is well-known that $L(K_3) = K_3$ and $L(K_{1,3}) = K_3$, making them the only two graphs indistinguishable by their line graphs [43]. Consequently, if input graphs contain both triangles and claws before conversion to line graphs, naive methods may incorrectly report non-existent common subgraphs. To resolve this, we use the inverse mapping δ_i, which maps vertices in $V(L(G_i))$ back to their corresponding edges in $E(G_i)$. When a K_3 is found in $\star_{i=1}^{n} L(G_i)$, we define H_i as the edge-induced subgraph of G_i whose edges correspond (via δ_i) to the vertices of this K_3 in $L(G_i)$. If $H_i \not\simeq H_j$ for some $i \neq j$, then one of G_i and G_j contains a claw while the other contains a triangle, and we classify such K_3 structures in $\star_{i=1}^{n} L(G_i)$ as "bad" triangles. Instead of including a bad triangle $T \simeq K_3$ in $\star_{i=1}^{n} L(G_i)$, we replace it with its three subgraphs $T - v_1$, $T - v_2$, and $T - v_3$. These derived subgraphs serve as refined candidates for the MCES of $G_1, \ldots, G_n$, ensuring that only valid common subgraphs are considered.

5.4 Ordering of input graphs

As we will see in Section 6, the ordering of input graphs significantly impacts the algorithm's runtime. Consider three graphs ordered as $[G_1, G_2, G_3]$ versus $[G_1, G_3, G_2]$. If the number of maximal common subgraphs between G_1 and G_2 is much larger than between G_1 and G_3, then $|\mathcal{H}_{1,2}| > |\mathcal{H}_{1,3}|$. Since we compute $H \star G_3$ for each $H \in \mathcal{H}_{1,2}$ and $H \star G_2$ for each $H \in \mathcal{H}_{1,3}$, the ordering $[G_1, G_3, G_2]$ results in fewer graphs requiring clique detection, leading to improved efficiency. However, determining an optimal ordering a priori is challenging, as computing a maximum connected common subgraph between a pair of graphs is NP-hard [23]. To address this, we use a heuristic that greedily selects graph pairs with low

similarity, based on the intuition that graphs with lower similarity tend to have fewer maximal common subgraphs. Our procedure for ordering input graphs works as follows. Given $\{G_1, \ldots, G_n\}$ and a similarity measure $\kappa\colon \mathcal{G} \times \mathcal{G} \to \mathbb{R}$, we first select the two graphs G_i and G_j minimizing $\kappa_{ij} \coloneqq \kappa(G_i, G_j)$ over all pairs, and set the tentative ordering $O = [G_i, G_j]$. Then, given a tentative ordering $O = [G_{l_1}, \ldots, G_{l_k}]$, we select a graph $G_p \in \{G_1, \ldots, G_n\} \setminus \{G_{l_1}, \ldots, G_{l_k}\}$ minimizing $\max\{\kappa_{l_1 p}, \ldots, \kappa_{l_k p}\}$. We update $O = [G_{l_1}, \ldots, G_{l_k}, G_p]$ and terminate when all graphs have been added (i.e., when $\{G_1, \ldots, G_n\} \setminus \{G_{l_1}, \ldots, G_{l_k}\} = \emptyset$).

For similarity estimation, we employ two families of pairwise graph similarity measures. The first family is based on graph kernels [30], i.e., functions $\kappa\colon \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ widely used in machine learning. While standard graph kernels must be positive semi-definite, we do not impose this requirement. In particular, we use three kernels: the vertex histogram kernel (VH) [33], the Weisfeiler–Lehman optimal assignment kernel (WL) [30], and the neighborhood subgraph pairwise distance kernel (NSPD) [33]. The second measure, called the "minmax" similarity, is defined as follows. We compute the minmax similarity of two graphs G_i and G_j by first partitioning $G_i \star G_j$ into its `type-`A connected components $G'_1, \ldots, G'_k$ as outlined in Section 5.2, and then setting $\kappa_{ij} \coloneqq \max\{|V(G'_\ell)| \mid \ell = 1, \ldots, k\}$. Using the minmax similarity measure with the greedy ordering outlined above, we greedily minimize the size of the largest graph in which we have to find cliques at each step. The computational results in Section 6 demonstrate a strong relationship between the chosen similarity measures and the number of maximal connected common subgraphs, validating the choice of these measures.

Computational trade-offs. The practical cost of the exact framework is governed by three coupled factors: the size of the modular products that must be searched, the number of maximal intermediate candidates retained during the recursive reduction, and the preprocessing cost of pruning selected `type-0` edges. The ordering heuristics target the first two factors by attempting to keep intermediate products small, whereas pruning may reduce later clique-search cost at the expense of additional preprocessing.

Accordingly, the implementation should be viewed as a configurable exact workflow rather than as a single fixed pipeline, and partial pruning can be preferable when preprocessing becomes the dominant cost.

6 Results

To study the relationship between graph similarity and the number of maximal `type-`A connected common edge subgraphs (for a fixed label set A, cf. Section 2), we conducted the following experiment. For 500 molecular graphs, $\mathcal{M} = \{G_1, \ldots, G_{500}\}$, from the ZINC data set [39], each having n vertices with $20 \leq n \leq 25$, we computed for each pair of graphs their number of maximal `type-`A connected common edge subgraphs, and also the similarity measures (the three graph kernels VH, WL, NSPD, and the minmax similarity; see Section 5).

6.1 Experimental setup, verification, and reproducibility

Our evaluation follows two complementary principles: *verification* of correctness via the formal results in Section 4, and *validation* of practical performance via computational benchmarks on molecular graphs in this section. All methods are implemented in `C++`, and the implementation together with the experiment scripts is publicly available at `https://github.com/johannesborg/cmces`. To support reproducibility of the reported runtimes, we specify the hardware/software environment, termination criteria, and all benchmark construction steps alongside the respective experiments, and we encourage reporting the exact dataset versions and molecule identifiers used when re-running or extending the benchmarks.

Validation scope. The experiments in this section are designed to validate two practically relevant aspects of the framework. First, they test whether the chosen similarity measures are informative for ordering molecular-graph instances by examining their relationship to the number of

maximal type-A connected common subgraphs. Second, they assess whether similarity-based ordering and the removal of selected type-0 edges reduce runtime while preserving exactness on the studied ZINC and ChEMBL22 benchmarks. We do not interpret these experiments as establishing universal runtime dominance across arbitrary graph classes; rather, they validate practical usefulness for the molecular-graph regime considered here.

Algorithmic interpretation of the benchmarks. The experiments are intended to evaluate the proposed exact method as an algorithm for labeled-network comparison rather than to optimize a single implementation constant. In particular, they test whether pairwise similarity provides a useful proxy for ordering the input graphs and whether selective removal of admissible type-0 edges reduces the effective search space while preserving the exact solution set on the studied instances. Read in this way, the benchmark results support the algorithmic role of ordering and pruning as structural components of the method, while the reported runtimes remain specific to the molecular benchmark regime and hardware configuration considered here.

The relationship between type-A connected common edge subgraphs of G_i and G_j and the respective similarity between G_i and G_j is shown in Figure 4. To avoid having the plot dominated by outliers, we aggregated pairs of graphs from $\mathcal{M}$ into buckets. To be more precise, let κ_{ij} be one of the four normalized similarity measures VH, WL, NSPD, or minmax between $G_i, G_j \in \mathcal{M}$. Moreover, let m_{ij} be the number of maximal type-A connected cliques in the product $L(G_i) \star L(G_j)$ (and, therefore, the number of maximal type-A connected common edge subgraphs of G_i and G_j induced by these cliques). We now define a set of 1,000 buckets, $B = \{b_1, \ldots, b_{1000}\}$, where each bucket $b_\ell \coloneqq \{(i,j) \mid \kappa_{ij} \in [\ell \cdot 0.001 - 0.005,\ \ell \cdot 0.001 + 0.005]\}$ collects the index pairs whose similarity falls into the prescribed interval. For each nonempty bucket b_ℓ we compute $(x_\ell, y_\ell) = \big(\ell \cdot 0.001,\ \frac{1}{|b_\ell|} \sum_{(i,j)\in b_\ell} m_{ij}\big)$. The plot showing the pairs (x_ℓ, y_ℓ) computed over all pairs of graphs in $\mathcal{M}$ is provided in Figure 4. Roughly speaking, Figure 4 shows how the average number of maximal

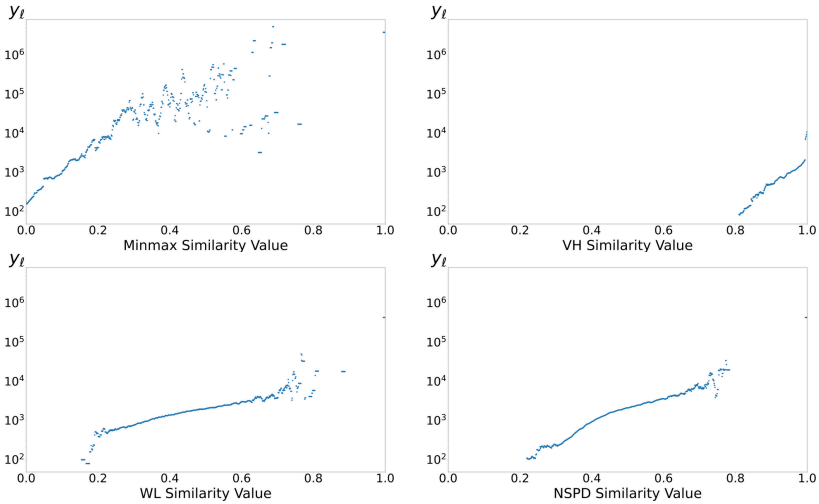

Figure 4. Plots showing the relationship between the four normalized similarity measures VH, WL, NSPD, and minmax (x-axis) and the values y_ℓ (y-axis), where y_ℓ is the average number of `type-`A connected common edge subgraphs of two graphs within a similarity bucket (see text for details).

`type-`A connected cliques in the product $L(G) \star L(H)$ (and, hence, the average number of `type-`A connected common edge subgraphs of G and H induced by those cliques) varies with the similarity between G and H, for all pairs $G, H \in \mathcal{M}$.

Figure 4 suggests a consistent trend for the VH, WL, and NSPD kernels and the minmax similarity: higher similarity values tend to correspond to larger values of y_ℓ, i.e., more `type-`A connected common edge subgraphs induced by maximal cliques in the respective products. This motivates using these similarity measures for the greedy ordering heuristic in Section 5.4.

To test the effect of ordering the input graphs together with "removal of certain `type-0` edges" for finding MCES (see the last part of Section 5.2), we created 500 problem instances, each containing 5 molecular graphs with 50 atoms (excluding hydrogen) randomly chosen from the ChEMBL22 database [44]. All experiments were performed on a machine with 32 GB of LPDDR5X RAM, an Intel Core Ultra 9 processor, and a

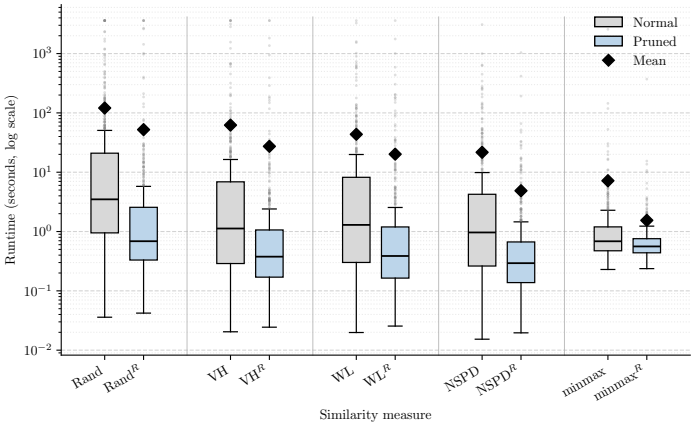

Figure 5. Box-plot showing the effect of different techniques applied to the input graphs (instances of 5 molecular graphs with 50 atoms) on the runtime for finding MCES. In particular, we applied the greedy-ordering based on the four similarity measures VH, WL, NSPD and minmax together with the computation of maximal `type-A` connected cliques with and without the removal of `type-0` edges (see Section 5). The term $Z \in \{\text{VH, WL, NSPD, minmax}\}$ on the x-axis refers to the application of measure Z without this refinement step, while Z^R means that the removal of certain `type-0` edges has been applied. The dotted lines represent the mean values across all instances. Without greedy-ordering, the runtime exceeded in many cases one hour, if it finished at all.

1 TB NVMe SSD, running Fedora Workstation 41. To determine an ordering of the input graphs, we used the four similarity measures VH, WL, NSPD, and minmax and the method outlined in Section 5.4. Furthermore, we also used a random ordering to serve as a baseline for comparison. For each ordering, we determined the maximal `type-`A connected cliques with and without the removal of certain `type-0` edges. Each of the ten different configurations was applied to the 500 test instances and the runtime in seconds was recorded. For all configurations and test instances, we terminated early if the runtime exceeded 3600 seconds (one hour), and for those instances we recorded the runtime as 3600 seconds. As such, for some configurations the recorded runtime is a lower bound on the actual

runtime. The resulting box plots are shown in Figure 5. For Rand, there were 9 instances where the runtime reached 3600 seconds, and for Rand^R there were 4 such instances. For VH there were 4 such instances, and for VH^R there were 2 such instances. For both WL and WL^R there was one such instance. For NSPD, NSPD^R, minmax, and minmax^R all instances were solved in less than one hour. Finally, we do not report the size-based ordering heuristic used in [27] separately, since it is not informative on this benchmark. Because all graphs have 50 vertices, the heuristic effectively degenerates to an arbitrary ordering and is therefore expected to behave comparably to the random-ordering baseline.

In contrast, the mean runtime for graphs ordered by similarity ranges from 1.54 to 62.23 seconds, highlighting the significant impact of the ordering methods. Moreover, the runtime improves in all cases when applying the removal of the specified `type-0` edges. The minmax ordering with the refinement step achieves the lowest mean runtime. Since NSPD, NSPD^R, minmax, and minmax^R are the only configurations where no instance reached the 3600-second cutoff, the gain in efficiency from using greedy ordering with NSPD or minmax, combined with removing certain `type-0` edges, is substantial compared to the baseline (random ordering without `type-0` edge removal). Overall, these heuristics preserve exactness while improving efficiency in practice on this benchmark.

In Figure 6 we show five molecular graphs and highlight two different labeled MCES in red and green, respectively.

7 Inference of graph transformation rules

The double-pushout (DPO) approach is a classical algebraic framework for graph transformation, introduced in the context of graph grammars by Ehrig et al. [19] and surveyed in, e.g., the handbook chapter of Corradini et al. [13]. Habel et al. [25] provide a later revisitation that discusses variants such as injective matching, which is natural in chemical settings where atoms should not be merged by a rule. DPO-style graph transformations have also been advocated as a convenient rule-based representation of chemical reactions with explicit atom maps and rule composition [1,2,4],

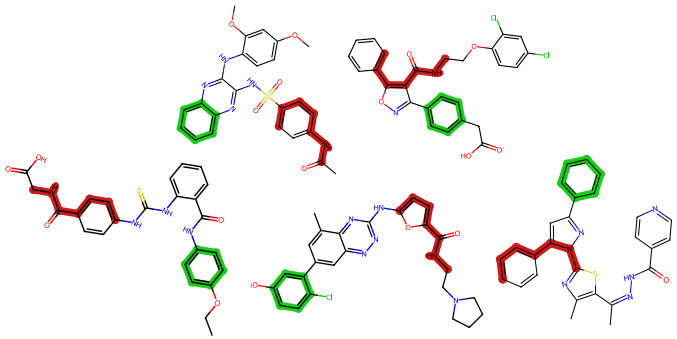

Figure 6. An example instance with five different molecular graphs from the ChEMBL22 database with 35 non-hydrogen vertices each. There are exactly two MCESs with 6 edges each. A single occurrence of each MCES within each graph is highlighted in red and green, respectively; additional occurrences may exist.

and are implemented in chemistry-facing tooling such as MØD [3].

In this formalism, a graph transformation rule can be represented by a span $L \xleftarrow{l} K \xrightarrow{r} R$, where L, K, R are graphs and l, r are monomorphisms. In a chemical context we can think of L as a molecular graph representing the reactant and R as a molecular graph representing the product. Furthermore, the monomorphisms mapping, respectively, from K to L and K to R can be thought of as the atom-atom mapping of the reaction. In what follows we assume that the atom-atom mapping is injective on vertices and covers all atoms of interest in the products (i.e. atoms not in the image are treated as absent via the symbol x_V). We can obtain a single graph representing the overall reaction by combining L and R via the atom mapping. We let $L = (V_L, E_L, \lambda_{V_L}, \lambda_{E_L})$ and $R = (V_R, E_R, \lambda_{V_R}, \lambda_{E_R})$. For ease of notation we view undirected edges as unordered pairs $\{u, v\}$ and define

$$\lambda'_{E_L}(\{u,v\}) = \begin{cases} \lambda_{E_L}(\{u,v\}), & \text{if } \{u,v\} \in E_L, \\ x, & \text{otherwise.} \end{cases}$$

Here x is an arbitrary symbol satisfying $x \notin \Sigma_E$. We also introduce an arbitrary symbol x_V with $x_V \notin \Sigma_V$ to represent an "absent" vertex label.

Similarly, define a (partial) vertex mapping $m : V_L \to V_R \cup \{\perp\}$ by

$$m(v) = \begin{cases} r(l^{-1}(v)), & \text{if } v \in \mathrm{im}(l), \\ \perp, & \text{otherwise.} \end{cases}$$

We extend the edge-labeling on R to unordered pairs of vertices in V_L by

$$\lambda'_{E_R}(\{u,v\}) = \begin{cases} \lambda_{E_R}(\{m(u), m(v)\}), & \text{if } m(u) \neq \perp,\ m(v) \neq \perp, \\ & \text{and } \{m(u), m(v)\} \in E_R, \\ x, & \text{otherwise.} \end{cases}$$

We define a function Γ which, given a graph transformation rule $L \xleftarrow{l} K \xrightarrow{r} R$, returns a single graph

$$\Gamma\big(L \xleftarrow{l} K \xrightarrow{r} R\big) = (V, E, \lambda_V, \lambda_E).$$

This graph represents the graph transformation rule $L \xleftarrow{l} K \xrightarrow{r} R$.

- Let $V = V_L$.
- For every $v \in V$ let

$$\lambda_V(v) = \big(\lambda_{V_L}(v),\ \tilde{\lambda}_{V_R}(v)\big), \quad \tilde{\lambda}_{V_R}(v) = \begin{cases} \lambda_{V_R}(m(v)), & \text{if } m(v) \neq \perp, \\ x_V, & \text{otherwise.} \end{cases}$$

- Let E be defined by

$$E = E_L \cup \big\{\{u,v\} \mid m(u) \neq \perp,\ m(v) \neq \perp, \\ \{m(u), m(v)\} \in E_R\big\}.$$

- For every $\{u,v\} \in E$ let $\lambda_E(\{u,v\}) = \big(\lambda'_{E_L}(\{u,v\}), \lambda'_{E_R}(\{u,v\})\big)$.

Given a set of graph transformation rules representing chemical reactions, we can map them to labeled graphs via Γ and compute an MCES. To interpret such an MCES as a generalization of a set of reactions, we restrict to sets of reactions that share the same *reaction core*. The reaction

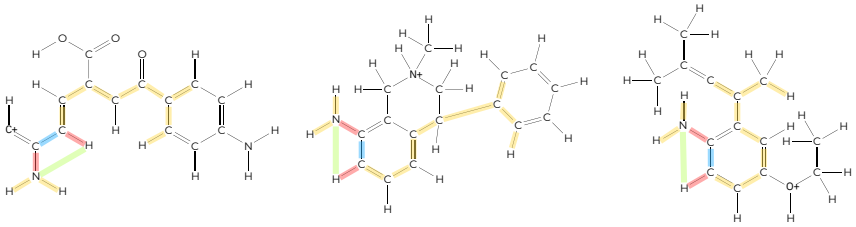

Figure 7. An instance of the anchored MCES problem for graph transformation rules. Each graph represents a fragmentation reaction taken from an in silico fragmentation tool. The maximum common subgraph is highlighted within each graph using color: yellow for non-anchor edges; and red/green/blue for anchor edges by change type. Here, a red edge symbolizes a single bond being removed, a green edge symbolizes a new single bond being created, and a blue edge symbolizes a single bond being converted to a double bond. The solution shows that all three reactions have in common that an ammonia molecule is cleaved off, which is a common fragmentation reaction occurring in gas-phase ions.

core of a graph transformation rule is the subgraph of the combined graph that contains exactly the vertices and edges that change: we remove all edges whose edge-label tuples have identical components, and we then remove isolated vertices unless their vertex-label tuples contain two different elements. We refer to this reaction core as an *anchor*. When computing an MCES for a set of reactions with isomorphic anchors under the additional constraint that the solution must contain the anchor, we call the problem the *anchored* MCES problem. An example instance is shown in Figure 7. Since the anchor is fixed, we can prune candidate solutions that do not contain it, which can reduce the search space compared to the unconstrained MCES problem.

Given a set of reaction mechanisms, we group them according to their reaction cores, such that two reactions are put in the same set if and only if they have isomorphic reaction cores. We then solve the anchored MCES problem for each set of reactions to obtain one graph transformation rule per reaction core.

The motivation for solving the anchored MCES problem is to find the most restrictive generalization of a set of chemical reactions with the same reaction cores. Typically, we use graph transformation rules to expand

chemical spaces. For instance, we might be interested in predicting what fragments of a precursor ion would be observed in a mass spectrometer. However, if the graph transformation rules we use are too general, we would produce too many spurious fragments, and if the graph transformation rules we use are too restrictive, we miss many fragments that would otherwise be observed in practice.

8 Conclusion

We have presented an exact and reproducible algorithmic framework for comparing multiple labeled graphs, with molecular-graph comparison in computational chemistry as the primary application setting. By combining modular-product formulations, an adapted Bron–Kerbosch search, pruning of redundant `type-0` product edges, and similarity-based graph ordering, the framework unifies exact MCIS and MCES computation, including `type-`A-connected variants, under explicit label and connectivity constraints. Formal proofs establish correctness, and benchmarks on the ZINC and ChEMBL22 datasets show that the method is practically usable on the studied molecular instances. Because the framework is formulated for arbitrary labeled graphs, it also provides a reproducible exact method for labeled-network comparison beyond the molecular case. The publicly available implementation supports comparative evaluation, and the reaction-rule example illustrates that the framework supports application-driven tasks beyond similarity analysis.

Practical implications. From an applied-modeling perspective, the framework provides an exact and interpretable way to identify conserved structural motifs across multiple molecular graphs under explicit label and connectivity constraints. This makes it useful both as a standalone method for common-substructure analysis and as a rigorous exact baseline against which approximate or heuristic methods can be evaluated. The reaction-rule example further shows that the framework is not limited to similarity scoring, but can also support the extraction of shared mechanistic patterns from sets of related reactions.

When the exact framework is most useful. In practice, the framework is most useful when exhaustive identification of all optimal shared substructures matters more than computational throughput, for example when constructing exact baselines, analyzing moderate-size collections of molecular graphs under explicit label constraints, or extracting conserved reaction patterns from closely related reactions. In such settings, enumerating all optimal solutions can be as important as determining the optimum size itself. When throughput on much larger or less constrained inputs is the main priority, the framework is better interpreted as a reference method against which approximate strategies can be calibrated.

Limitations and scope. As with other exact methods for NP-hard graph matching problems, worst-case runtimes can grow rapidly with instance size, structural ambiguity, and weak label constraints. The present contribution is therefore best viewed as a verified exact computational methodology for moderate-size labeled-network comparison problems, with molecular graphs as the main validation regime, rather than as a universal large-scale solver. For substantially larger inputs or less constrained application settings, approximate, hybrid, or decomposition-based strategies may be more appropriate.

Data availability

The source code and experiment scripts are publicly available at `https://github.com/johannesborg/cmces`. The molecular datasets analyzed in this study are available from the ZINC database [39] and ChEMBL (release 22) [44]; benchmark construction steps and computational settings are fully described in the manuscript.

***Acknowledgment*:** This work was supported by the Novo Nordisk Foundation (MATOMIC, 0066551); by the European Union's Horizon Europe Doctoral Network programme under the Marie-Sklodowska-Curie grant agreement No. 101072930 (TACsy – Training Alliance for Computational Systems Chemistry); and by the Vienna Science and Technology Fund

(WWTF) and the City of Vienna (Grant ID: 10.47379/ICT22059). The funders had no role in the study design, collection, analysis and interpretation of data, writing of the report, or the decision to submit the article for publication.

References

[1] J. L. Andersen, A. Davoodi, R. Fagerberg, C. Flamm, W. Fontana, J. Kolčák, C. V. F. P. Laurent, D. Merkle, N. Nøjgaard, Automated inference of graph transformation rules, *Fundam. Inf.* **196** (2026) #13347.

[2] J. L. Andersen, C. Flamm, D. Merkle, P. F. Stadler, Inferring chemical reaction patterns using rule composition in graph grammars, *J. Sys. Chem.* **4** (2013) #4.

[3] J. L. Andersen, C. Flamm, D. Merkle, P. F. Stadler, A software package for chemically inspired graph transformation, in: R. Echahed, M. Minas (Eds.), *Graph Transformation*, Springer, Cham, 2016, pp. 73–88.

[4] J. L. Andersen, C. Flamm, D. Merkle, P. F. Stadler, Rule composition in graph transformation models of chemical reactions, *MATCH Commun. Math. Comput. Chem.* **80** (2018) 661–704.

[5] H. G. Barrow, R. M. Burstall, Subgraph isomorphism, matching relational structures and maximal cliques, *Inf. Process. Lett.* **4** (1976) 83–84.

[6] J.-F. Boulicaut, M. R. Berthold, T. Horváth (Eds.), *Discovery Science: 11th International Conference, DS 2008, Budapest, Hungary, October 13–16, 2008, Proceedings*, Springer, Berlin, 2008.

[7] A. T. Brint, P. Willett, Algorithms for the identification of three-dimensional maximal common substructures, *J. Chem. Inf. Comput. Sci.* **27** (1987) 152–158.

[8] C. Bron, J. Kerbosch, Algorithm 457: finding all cliques of an undirected graph, *Commun. ACM* **16** (1973) 575–577.

[9] Y. Cao, T. Jiang, T. Girke, A maximum common substructure-based algorithm for searching and predicting drug-like compounds, *Bioinformatics* **24** (2008) i366–i374.

[10] L. Cardone, S. Quer, The multi-maximum and quasi-maximum common subgraph problem, *Computation* **11** (2023) #69.

[11] R. Carraghan, P. M. Pardalos, An exact algorithm for the maximum clique problem, *Oper. Res. Lett.* **9** (1990) 375–382.

[12] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, *Int. J. Pattern Recogn. Artif. Intell.* **18** (2004) 265–298.

[13] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, M. Löwe, Algebraic approaches to graph transformation – part I: basic concepts and double pushout approach, in: G. Rozenberg (Ed.), *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*, World Scientific, Singapore, 1997, pp. 163–246.

[14] A. Dalke, J. Hastings, FMCS: a novel algorithm for the multiple MCS problem, *J. Cheminf.* **5** (2013) #O6.

[15] A. Davoodi, M. Holeňa, M. Brunovský, A. Kathpalia, J. Hlinka, M. Bareš, M. Paluš, Response prediction of antidepressants: using graph theory tools for brain network connectivity analysis, *Biomed. Signal Process. Control* **103** (2025) #107362.

[16] S. Derrible, C. Kennedy, The complexity and robustness of metro networks, *Phys. A: Stat. Mech. Appl.* **389** (2010) 3678–3691.

[17] E. Duesbury, J. Holliday, P. Willett, Comparison of maximum common subgraph isomorphism algorithms for the alignment of 2D chemical structures, *ChemMedChem* **13** (2018) 588–598.

[18] P. J. Durand, R. Pasari, J. W. Baker, C. C. Tsai, An efficient algorithm for similarity analysis of molecules, *Internet J. Chem.* **2** (1999) 1–16.

[19] H. Ehrig, M. Pfender, H. J. Schneider, Graph-grammars: an algebraic approach, in: *14th Annual IEEE Symposium on Switching and Automata Theory (SWAT 1973)*, IEEE Computer Society, Los Alamitos, 1973, pp. 167–180.

[20] H. C. Ehrlich, M. Rarey, Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review, *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **1** (2011) 68–79.

[21] P. Englert, P. Kovács, Efficient heuristics for maximum common substructure search, *J. Chem. Inf. Model.* **55** (2015) 941–955.

[22] S. Eubank, H. Guclu, V. S. A. Kumar, M. V. Marathe, A. Srinivasan, Z. Toroczkai, N. Wang, Modelling disease outbreaks in realistic urban social networks, *Nature* **429** (2004) 180–184.

[23] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[24] A. Gupta, N. Nishimura, Finding largest subtrees and smallest supertrees, *Algorithmica* **21** (1998) 183–210.

[25] A. Habel, J. Müller, D. Plump, Double-pushout graph transformation revisited, *Math. Struct. Comput. Sci.* **11** (2001) 637–688.

[26] R. Hammack, W. Imrich, S. Klavžar, *Handbook of Product Graphs*, CRC Press, Boca Raton, 2011.

[27] R. Hariharan, A. Janakiraman, R. Nilakantan, B. Singh, S. Varghese, G. Landrum, A. Schuffenhauer, MultiMCS: a fast algorithm for the maximum common substructure problem on multiple molecules, *J. Chem. Inf. Model.* **51** (2011) 788–806.

[28] M. Hassan, R. D. Brown, S. Varma-O'Brien, D. Rogers, Cheminformatics analysis and learning in a data pipelining environment, *Mol. Divers.* **10** (2006) 283–299.

[29] I. Koch, Enumerating all connected maximal common subgraphs in two graphs, *Theor. Comput. Sci.* **250** (2001) 1–30.

[30] N. M. Kriege, F. D. Johansson, C. Morris, A survey on graph kernels, *Appl. Netw. Sci.* **5** (2020) #6.

[31] G. Levi, A note on the derivation of maximal common subgraphs of two directed or undirected graphs, *Calcolo* **9** (1973) 341–352.

[32] J. J. McGregor, Backtrack search algorithms and the maximal common subgraph problem, *Softw. Pract. Exp.* **12** (1982) 23–34.

[33] G. Nikolentzos, G. Siglidis, M. Vazirgiannis, Graph kernels: a survey, *J. Artif. Intell. Res.* **72** (2021) 943–1027.

[34] J. W. Raymond, E. J. Gardiner, P. Willett, Heuristics for similarity searching of chemical graphs using a maximum common edge subgraph algorithm, *J. Chem. Inf. Comput. Sci.* **42** (2002) 305–316.

[35] J. W. Raymond, E. J. Gardiner, P. Willett, RASCAL: calculation of graph similarity using maximum common edge subgraphs, *Comput. J.* **45** (2002) 631–644.

[36] J. W. Raymond, P. Willett, Maximum common subgraph isomorphism algorithms for the matching of chemical structures, *J. Comput. Aided Mol. Des.* **16** (2002) 521–533.

[37] M. Sipser, *Introduction to the Theory of Computation*, Course Technology, Boston, 2013.

[38] A. Soulé, V. Reinharz, R. Sarrazin-Gendron, A. Denise, J. Waldispühl, Finding recurrent RNA structural networks with fast maximal common subgraphs of edge-colored graphs, *PLoS Comput. Biol.* **17** (2021) #e1008990.

[39] T. Sterling, J. J. Irwin, ZINC 15 – ligand discovery for everyone, *J. Chem. Inf. Model.* **55** (2015) 2324–2337.

[40] J. R. Ullmann, An algorithm for subgraph isomorphism, *J. ACM* **23** (1976) 31–42.

[41] D. L. Urban, T. H. Keitt, Landscape connectivity: a graph-theoretic perspective, *Ecology* **82** (2001) 1205–1218.

[42] R. J. P. van Berlo, W. Winterbach, M. J. L. de Groot, A. Bender, P. J. T. Verheijen, M. J. T. Reinders, D. de Ridder, Efficient calculation of compound similarity based on maximum common subgraphs and its application to prediction of gene transcript levels, *Int. J. Bioinf. Res. Appl.* **9** (2013) 407–432.

[43] H. Whitney, Congruent graphs and the connectivity of graphs, *Am. J. Math.* **54** (1932) 150–168.

[44] B. Zdrazil, E. Felix, F. Hunter, E. J. Manners, J. Blackshaw, S. Corbett, M. de Veij, H. Ioannidis, D. Mendez Lopez, J. F. Mosquera, M. P. Magariños, N. Bosc, R. Arcila, T. Kizilören, A. Gaulton, A. P. Bento, M. F. Adasme, P. Monecke, G. A. Landrum, A. R. Leach, The ChEMBL database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods, *Nucleic Acids Res.* **52** (2024) D1180–D1192.