Efficient Correlated Subgraph Searches for AI-powered Drug Discovery

Hiroaki Shiokawa[1] , **Yuma Naoi**[2] and **Shohei Matsugu**[2]
[1]Center for Computational Sciences, University of Tsukuba, Japan
[2]Graduate School of Science and Technology, University of Tsukuba, Japan
shiokawa@cs.tsukuba.ac.jp, {naoi, matsugu}@kde.cs.tsukuba.ac.jp

Abstract

Correlated subgraph searches (CSSs) are essential building blocks for AI-powered drug discovery. Given a query molecule modeled as a graph, CSS finds top-k molecules correlated to the query in a database. However, the cost increases exponentially with the molecule size. Herein we present *Corgi*, a framework to accelerate CSS methods while ensuring top-k search accuracy. Corgi dynamically excludes unnecessary subgraphs to overcome the expensive cost without sacrificing search accuracy. Our experimental analysis confirms that Corgi has a shorter running time and improved accuracy compared to existing state-of-the-art methods, while a case study demonstrates that Corgi is suitable for practical AI-powered drug discovery.

1 Introduction

The motivation of this work is to accelerate AI-powered drug discovery over large molecule databases. The correlated subgraph search (CSS) [Ke *et al.*, 2007] has become an essential building block in effectively identifying molecules for clinical use [Ma *et al.*, 2019; Lin *et al.*, 2020; Zhang *et al.*, 2022; Demir *et al.*, 2022; Yagi and Shiokawa, 2022]. CSS models a biochemical database as a graph database, which is a set of small graphs corresponding to molecules. Given a query graph (molecule), CSS finds the subgraphs (substructures of a molecule) most correlated to the query in a database. Since the correlation between substructures is known to be a trustworthy feature in predicting drug properties [Askr *et al.*, 2023; Yagi *et al.*, 2023], CSS has been applied to the following AI-based drug discovery scenarios:

Ligand-based virtual screening (LBVS). LBVS is a common step to extract molecules, which are most likely to bind to a drug target. Recently, neural network models have become a standard way for large-scale LBVS [Askr *et al.*, 2023; Kawano *et al.*, 2024]. However, due to the structural complexity of molecules, existing models are overwhelmed by low extraction accuracy [Matsugu *et al.*, 2023], and it is still challenging for them to dig out the beneficial molecules for clinical use. To overcome this limitation, CSS has attracted much attention because it can uncover correlated molecule pairs with a high binding probability. For example, [Naoi and Shiokawa, 2023] and [McGibbon *et al.*, 2023] applied CSS to their LBVS models to extract effective features from molecules. Their CSS-based approaches discovered more beneficial molecules than other models. Similarly, CSS-based models achieved a better performance in detecting anti-cancer drugs [Nguyen and Wei, 2019; Prateek *et al.*, 2020].

Molecule property prediction. CSS can enhance the performance of AI models for predicting molecule properties. [Ma *et al.*, 2019] and [Lin *et al.*, 2020] proposed CSS-based models to predict drug interactions for estimating whether a drug could react with one or more other drugs. The correlated substructures of drugs can reveal non-trivial drug-drug relations among molecules. By capturing such relations, these CSS-based networks outperformed other prediction models regarding the interaction estimation accuracy. Analogously, [Lee *et al.*, 2014] and [Bhattacharjee and Vlachos, 2020] employed CSS to extract correlated proteins that frequently co-occur in an individual. Such proteins derived significant protein patterns to conduct detailed pre-clinical studies.

CSS can also be used in other AI-powered applications such as pathway analysis [Abdel-Hafiz *et al.*, 2022] and molecular dynamics simulation [Yang *et al.*, 2022].

Although CSS is effective, a large computation time is required because it incurs exponential costs [Ke *et al.*, 2007]. For each graph, CSS must explore $\mathcal{O}(2^n)$ candidate subgraphs, where n is the number of nodes in the graph. Since subgraph correlation neither yields downward nor upward closure properties, traditional *a priori* approaches [Yan and Han, 2002; Elseidy *et al.*, 2014] are not directly applicable to reduce the candidates. Furthermore, recent drug discovery requires handling massive databases, which can contain more than 10^5 molecules [Lipinski and Hopkins, 2004]. Hence, CSS requires several weeks or months for drug discovery.

1.1 Existing Approaches and Challenges

Many studies have strived to overcome the exponential costs in CSS. Although CSS must enumerate correlated subgraph candidates from an exponential space, exploring candidates only from frequent subgraphs is more reasonable. On the basis of this idea, *threshold-based methods* [Ke *et al.*, 2007; Samiullah *et al.*, 2014; Chowdhury *et al.*, 2021] drop infrequent subgraphs from the database by using user-specified thresholds. For instance, *CGSearch* [Ke *et al.*, 2007] invokes

the frequent subgraph search, gSpan [Yan and Han, 2002], before candidate enumeration. The state-of-the-art method, *bFact* [Chowdhury *et al.*, 2021], integrates several heuristic thresholds with gSpan to avoid exponential costs. Although these methods mitigate costs, they suffer from two critical drawbacks. First, finding frequent subgraphs still incurs expensive costs. Second, their heuristic thresholds sacrifice the search quality, making effective drug discovery difficult.

Bounding methods [Zou *et al.*, 2009; Ke *et al.*, 2009] have recently attracted attention as alternatives in the AI community [Naoi and Shiokawa, 2023]. These methods theoretically derive the bounds of the correlation coefficient between two subgraphs. For instance, *PG-search* [Zou *et al.*, 2009] incrementally excludes unpromising subgraphs by following the bounds. To further improve the efficiency, the state-of-the-art method, *TopCor* [Ke *et al.*, 2009], integrates heuristics with the bounds. Unlike threshold-based methods, these methods guarantee to find the exact top-k correlated subgraphs to a query graph within a reasonable running time.

Although bounding methods have improved efficiency, their computational costs are high to handle massive databases. The bounding methods require at least $\mathcal{O}(n^2)$ time to test a correlation between two subgraphs [Welke, 2020]. That is, these methods totally incur $\mathcal{O}(NMn^2)$ time, where N and M are the numbers of graphs in a database and tested subgraphs in each graph, respectively. In addition, bounding methods cannot exclude enough unpromising subgraphs if a query graph is frequent in the database because the bounds become loose. In the worst case, this incurs $\mathcal{O}(M) \approx \mathcal{O}(2^n)$ time, resulting a total time of $\mathcal{O}(N2^n n^2)$. Thus, the computational efficiency in CSS remains a challenge.

1.2 Our Approaches and Contributions

In this paper, we present a novel framework, *Corgi*, that accelerates existing CSS methods while ensuring top-k search accuracy. As discussed in Section 1.1, existing CSS methods suffer from exponential costs since they need to compute almost all subgraph candidates in a database. By contrast, Corgi dynamically removes unnecessary candidate subgraph computations from the database. In practical molecule databases, many graphs share similar substructures due to the limited varieties of chemical elements and valences [Shiokawa *et al.*, 2019]. Since Corgi can effectively exclude such shared substructures by abstracting the database, it avoids exhaustive subgraph candidate computations.

Based on the above idea, we design Corgi with two steps. First, a *summarized graph view* abstracts a graph database to skip redundant computations for the shared substructures (Section 3.2). Second, a *multi-viewed top-k search* performs CSS methods over multiple summarized graph views to ensure top-k search accuracy (Section 3.3). Consequently, Corgi has the following attractive characteristics:

- **Efficient:** Corgi is significantly faster than state-of-the-art CSS methods (Section 4.1). Corgi has a better time complexity than existing CSS methods (Theorem 1).
- **Accurate:** Corgi outputs accurate top-k correlated subgraphs to a query graph (Section 4.3). Corgi theoretically guarantees the top-k search accuracy (Theorem 2).

Symbol	Definition
g	Undirected graph, *i.e.,* $g = (V, E, l)$
V	Set of nodes in g
E	Set of edges in g
l	Label function of g
$\mathcal{D}$	Graph database (*i.e.,* $\mathcal{D} = \{g_1, g_2, \ldots, g_N\}$)
n	Number of nodes in g
m	Number of edges in g
N	Number of graphs in $\mathcal{D}$
$g_i \subseteq g_j$	g_i is a subgraph of g_j (g_j is a supergraph of g_i)
$\mathcal{D}_g$	Projected database of $\mathcal{D}$ on g
$sup(g)$	Support of g in $\mathcal{D}$
$sup(g_i, g_j)$	Joint support of g_i and g_j in $\mathcal{D}$
$\phi(g_i, g_j)$	Correlation value between g_i and g_j in $\mathcal{D}$
$\mathcal{T}_k(q)$	Top-k search results in $\mathcal{D}$ for q
$\mathcal{A}_p(\mathcal{D})$	Set of folded node pairs in Definition 2
$\mathcal{L}(\mathcal{D})$	Set of all independent node pairs in $\mathcal{D}$
$\hat{\mathcal{D}}$	Summarized graph view of $\mathcal{D}$ in Definition 4
$\hat{g}$	Abstracted graph in Definition 4
$\hat{q}$	Abstracted query graph for MvTk search
$\hat{\mathcal{T}}_k(\hat{\mathcal{D}})$	Top-k search results in $\hat{\mathcal{D}}$ for $\hat{q}$
T	Number of summarized graph views
ϵ	Bound of false negative ratio in Lemma 5

Table 1: Definitions of the main symbols.

- **Practical:** We deployed Corgi in a practical LBVS model for clinical use. Corgi effectively discovered beneficial proteins that activate ADORA2A (Section 4.6).

Corgi is the first solution that achieves a high efficiency while maintaining search accuracy for large databases. We experimentally confirmed that Corgi is up to 48.6 times faster than other state-of-the-art methods. Although CSS plays a crucial role in achieving AI-powered drug discovery, applying it to large databases is challenging due to the expensive cost. By contrast, Corgi is well suited to massive databases and should contribute to various AI-based drug discovery.

2 Preliminary

Table 1 lists the main symbols used in this paper. A molecule is modeled as a graph $g = (V, E, l)$, where a node set V and an edge set E correspond to atoms and chemical bonds, respectively. l is a label function that maps nodes and edges to corresponding chemical elements and bond types, respectively. For simplicity, we denote $n = |V|$ and $m = |E|$.

Given $g_i = (V_i, E_i, l_i)$ and $g_j = (V_j, E_j, l_j)$, we denote $g_i \subseteq g_j$ if an injective function $f : V_i \to V_j$ exists such that for every edge $(u, v) \in E_i$, $(f(u), f(v)) \in E_j$, $l_i(u) = l_j(f(u))$, $l_i(v) = l_j(f(v))$, and $l_i(u, v) = l_j(f(u), f(v))$. If $g_i \subseteq g_j$, g_i is a *subgraph* of g_j (or g_j is a *supergraph* of g_i).

A graph database $\mathcal{D}$ is defined as a set of N graphs (*i.e.,* $\mathcal{D} = \{g_1, g_2, \ldots, g_N\}$). Given a database $\mathcal{D}$ and a graph g, we denote $\mathcal{D}_g = \{g' \in \mathcal{D} \mid g \subseteq g'\}$ as all supergraphs of g in $\mathcal{D}$. The *support* of g in $\mathcal{D}$, which is denoted as $sup(g)$, is defined as the proportion of $\mathcal{D}_g$ in $\mathcal{D}$ (*i.e.,* $sup(g) = \frac{|\mathcal{D}_g|}{|\mathcal{D}|}$). Additionally, $sup(g_i, g_j) = \frac{|\mathcal{D}_{g_i} \cap \mathcal{D}_{g_j}|}{|\mathcal{D}|}$ is defined as the *joint support* of g_i and g_j in $\mathcal{D}$.

We employ the following *Pearson's correlation* [Reynolds, 1977] to measure the correlation between two graphs. Other measures can be applied in a similar manner [Ke *et al.*, 2008].

Definition 1. Given g_i and g_j, the Pearson's correlation between g_i and g_j is denoted by $\phi(g_i, g_j)$ and is defined as

$$\phi(g_i, g_j) = \frac{sup(g_i, g_j) - sup(g_i)sup(g_j)}{\sqrt{sup(g_i)sup(g_j)(1-sup(g_i))(1-sup(g_j))}}. \quad (1)$$

Note that $\phi(g_i, g_j) = 0$ if the denominator of Eq. (1) is 0.

$\phi(g_i, g_j)$ falls between -1 and 1. $\phi(g_i, g_j) = 0$ means that the occurrences of g_i and g_j in $\mathcal{D}$ are independent. If $\phi(g_i, g_j) > 0$, the occurrences of g_i and g_j are positively correlated in $\mathcal{D}$. Otherwise, they are negatively correlated.

Problem statement. We formulate the problem addressed in this paper. Given a graph database $\mathcal{D}$ and a query graph q, we focus on the top-k CSS problem to find k subgraphs that are the most positively correlated to q in $\mathcal{D}$. This is defined as

Problem 1. Given a graph database $\mathcal{D}$, a query graph $q = (V_q, E_q, l_q)$, and $k \in \mathbb{N}^+$, the top-k CSS problem is a task to find k subgraphs of graphs in $\mathcal{D}$, $\mathcal{T}_k(\mathcal{D}) = \{g_1, g_2, \ldots, g_k\}$ to maximize $\sum_{g_i \in \mathcal{T}_k(\mathcal{D})} \phi(q, g_i)$.

In the worst case, Problem 1 incurs $\mathcal{O}(N2^n n^2)$ time. Hence, existing methods fail to complete CSS on large databases.

3 Proposed Framework: Corgi

We present our framework to accelerate existing CSS methods with top-k search accuracy assurance.

3.1 Basic Ideas

Corgi aims to accelerate CSS methods while ensuring top-k search accuracy. Existing CSS methods explore the exponential number of subgraph candidates in $\mathcal{D}$. By contrast, Corgi introduces two approaches to remove unpromising subgraph computations without sacrificing search quality. The first approach theoretically derives *summarized graph views* of $\mathcal{D}$. The views produce small summaries of $\mathcal{D}$, which require significantly fewer search costs than their original counterparts. The second approach employs a *multi-viewed top-k (MvTk) search* to guarantee the search accuracy. Although the views improve the efficiency, they may lead to false negatives or positives. To overcome this, the MvTk search refines the search results using multiple summarized graph views. Unlike existing methods, Corgi computes limited subgraph candidates while maintaining the top-k search accuracy.

Our ideas have two main advantages. (1) Corgi finds the top-k correlated subgraphs with a short running time in real-world molecule databases. In practical drug discovery, the molecules often share similar substructures due to the limited varieties of chemical elements and valences. Our ideas successfully handle such structural properties because the summarized graph views can drastically reduce the graph sizes in $\mathcal{D}$ if graphs have similar substructures. This allows Corgi to enhance its performance even for large databases. (2) Corgi produces the same search results as existing CSS methods. We theoretically demonstrated that Corgi does not miss opportunities to find the top-k subgraphs if it uses a sufficient number of views. Thus, Corgi can guarantee the top-k search accuracy, unlike existing CSS methods.

3.2 Summarized Graph View

We introduce the *summarized graph view* to reduce the computational costs of CSS methods. It is a set of abstracted graphs generated by folding a set of node pairs in $\mathcal{D}$. First, we formally define the node pairs to be folded in the view.

Definition 2. For $(u, v) \in E$, a node pair is denoted by $\langle l(u), l(u, v), l(v)\rangle$. Given $\mathcal{D} = \{g_1, g_2, \ldots, g_N\}$ and $p \in \mathbb{N}^+$, $\mathcal{A}_p(\mathcal{D})$ is a set of p folded node pairs of $\mathcal{D}$ expressed as

$$\mathcal{A}_p(\mathcal{D}) = \begin{cases} \mathcal{A}_{p-1}(\mathcal{D}) \cup \psi(\mathcal{L}(\mathcal{D}) \backslash \mathcal{A}_{p-1}(\mathcal{D})) & (p > 0) \\ \emptyset & (p = 0) \end{cases}, \quad (2)$$

where $\mathcal{L}(\mathcal{D}) = \bigcup_{g_i \in \mathcal{D}} \bigcup_{(u,v) \in E_i} \{\langle l(u), l(u, v), l(v)\rangle\}$, and ψ is a function that randomly selects a node pair $\langle l(u), l(u, v), l(v)\rangle$ from $\mathcal{L}(\mathcal{D}) \backslash \mathcal{A}_{p-1}(\mathcal{D})$.

Definition 2 indicates that $\mathcal{A}_p(\mathcal{D})$ is composed of p independent node pairs, which are randomly selected from $\mathcal{L}(\mathcal{D})$.

For convenience, the following function is defined to map node v to u if $\langle l(u), l(u, v), l(v)\rangle \in \mathcal{A}_p(\mathcal{D})$.

Definition 3. Given the folded node pairs $\mathcal{A}_p(\mathcal{D})$, a mapping function $\pi_{\mathcal{A}_p(\mathcal{D})}(v)$ is defined as

$$\pi_{\mathcal{A}_p(\mathcal{D})}(v) = \begin{cases} u & (\langle l(u), l(u, v), l(v)\rangle \in \mathcal{A}_p(\mathcal{D})) \\ v & (Otherwise) \end{cases}. \quad (3)$$

Finally, we define the summarized graph view.

Definition 4. Given $\mathcal{D} = \{g_1, g_2, \ldots, g_N\}$ and $\mathcal{A}_p(\mathcal{D})$, a summarized graph view, which is denoted by $\hat{\mathcal{D}}$, is defined as $\hat{\mathcal{D}} = \{\hat{g_1}, \hat{g_2}, \ldots, \hat{g_N}\}$, where $\hat{g_i} = (\hat{V_i}, \hat{E_i}, \hat{l_i})$ is an abstracted graph composed of

$$\hat{V_i} = \bigcup_{u \in V_i} \pi_{\mathcal{A}_p(\mathcal{D})}(u), \quad (4)$$

$$\hat{E_i} = \{(u, v) \in \hat{V_i}^2 \mid (u, v) \in E_i\}, \quad (5)$$

and $\hat{l_i}$ is a label mapping function of $\hat{g_i}$.

Definition 4 indicates that folding node v into $u \in V_i$ generates an abstracted graph $\hat{g_i}$ if the corresponding node pair $\langle l(u), l(u, v), l(v)\rangle$ is included in $\mathcal{A}_p(\mathcal{D})$. By querying correlated subgraphs on the summarized graph view, Corgi can significantly reduce the search space of CSS methods compared to their original counterparts. This is because, as shown in Definition 4, $\hat{\mathcal{D}}$ drastically reduces the average graph size in $\mathcal{D}$ by folding the node pairs in $\mathcal{A}_p(\mathcal{D})$.

From Definition 4, we have the following property:

Lemma 1. Given $\mathcal{D}$ and $\mathcal{A}_p(\mathcal{D})$, a summarized graph view $\hat{D}$ can be obtained in $\mathcal{O}((p \cdot \frac{\overline{m}}{\overline{n}})N)$ time, where $\overline{n}$ and $\overline{m}$ are the average number of nodes and edges in $\mathcal{D}$, respectively.

Proof. For every graph g in $\mathcal{D}$, Definition 4 requires folding at most p node pairs included in $\mathcal{A}_p(\mathcal{D})$. Given a node pair $\langle l(u), l(u, v), l(v)\rangle \in \mathcal{A}_p(\mathcal{D})$, Corgi can fold node v into u by removing node v and edges incident on v. This incurs $\mathcal{O}(\frac{\overline{m}}{\overline{n}})$ time, where $\frac{\overline{m}}{\overline{n}}$ is the average degree of g [Shiokawa *et al.*, 2013]. Hence, Definition 4 requires $\mathcal{O}((p \cdot \frac{\overline{m}}{\overline{n}})N)$ time. □

In practice, $\frac{\overline{m}}{\overline{n}}$ should be a small constant in real-world molecule databases. Specifically, the real-world databases examined in Section 4 have $\overline{d} = \frac{\overline{m}}{\overline{n}} = 1.61$ on average (Table 2). Thus, Lemma 1 implies that Definition 4 incurs a nearly linear scalability against the database size. In other words, $\mathcal{O}((p \cdot \frac{\overline{m}}{\overline{n}})N) \approx \mathcal{O}(pN)$ time. In Section 4, we experimentally evaluate the overhead of Definition 4.

Here, we further derive the following properties, which are essential to discuss our MvTk search shown in Section 3.3.

Lemma 2. Let $g' \subseteq g_i$ be a top-k correlated subgraph, and (u, v) be an edge between $u \in V'$ and $v \in V_i \backslash V'$. If node u is folded into v, g' can be a false negative result.

Proof. Let $\hat{g}_i$ be an abstracted graph obtained by folding node u into v on g_i. $g' \not\subseteq \hat{g}_i$ clearly holds even if $g' \subseteq g_i$ because no injective mapping of u exists in $\hat{g}_i$. Thus, CSS methods fail to extract g' from $\hat{g}_i$. □

Lemma 3. Let $\hat{g}_i$ and $\hat{g}$ be abstracted from g_i and g, respectively. If $\hat{g}_i \subseteq \hat{g}$ but $g_i \not\subseteq g$, then $\hat{g}_i$ can be a false positive.

Proof. Since $\hat{g}_i \subseteq \hat{g}$, $\hat{g}$ can be regarded as the top-k subgraph in spite of $g_i \not\subseteq g$. Thus, $\hat{g}_i$ can be a false positive. □

Lemmas 2 and 3 indicate that Definition 4 may lead to false negatives or positives in the top-k search results.

3.3 MvTk Search

The *MvTk search* is introduced to guarantee the top-k search accuracy on the summarized graph views. As shown in Lemmas 2 and 3, Corgi produces false negative or positive results if it performs CSS on a summarized graph view. The MvTk search employs (1) an *extraction step* and (2) a *validation step* to exclude false negatives and positives, respectively.

(1) Extraction step. To avoid false negatives, Corgi extracts candidate subgraphs $\mathcal{C}$ using multiple summarized graph views. Let $\mathrm{CSS}(\mathcal{D}, q, k)$ be an existing CSS method that returns top-k subgraphs correlated to q in $\mathcal{D}$. First, we define the candidate subgraphs $\mathcal{C}$.

Definition 5. Given a query q and T summarized graph views $\hat{\mathcal{D}}_1, \hat{\mathcal{D}}_2, \ldots, \hat{\mathcal{D}}_T$, a set of candidate subgraphs $\mathcal{C}$ is defined as $\mathcal{C} = \bigcup_{i=0}^{T} \hat{\mathcal{T}}_k(\hat{\mathcal{D}}_i)$, where $\hat{\mathcal{T}}_k(\hat{\mathcal{D}}_i)$ is the top-k search result of $\mathrm{CSS}(\hat{\mathcal{D}}_i, \hat{q}_i, k)$ and $\hat{q}_i$ is a query graph abstracted by $\mathcal{A}_p(\mathcal{D})$ corresponding to $\hat{\mathcal{D}}_i$.

$\mathcal{C}$ is $T \times k$ correlated subgraphs obtained by querying $\hat{q}_i$ on $\hat{\mathcal{D}}_i$ for all views $\hat{\mathcal{D}}_1, \hat{\mathcal{D}}_2, \ldots, \hat{\mathcal{D}}_T$. Recall that $\mathcal{A}_p(\mathcal{D})$ is randomly selected from Definition 2. Consequently, $\hat{\mathcal{T}}_k(\hat{\mathcal{D}}_1), \ldots, \hat{\mathcal{T}}_k(\hat{\mathcal{D}}_T)$ can have diverse correlated subgraphs.

Herein, we derive the following property from Lemma 2:

Lemma 4. Given a summarized graph view $\hat{\mathcal{D}}$, and the corresponding folded node pairs $\mathcal{A}_p(\mathcal{D})$, the false negative ratio of top-k CSS on $\hat{\mathcal{D}}$ is $1 - (1 - \frac{p}{|\mathcal{L}(\mathcal{D})|})^{\overline{m}}$, where $\overline{m} = \frac{1}{N} \sum_{g_i \in \mathcal{D}} \frac{1}{2^{n_i}} \sum_{g_j \subseteq g_i} m_j$.

Proof. Suppose a subgraph $g_i \in \mathcal{D}$ is a top-k correlated subgraph of a query graph q (*i.e.,* $g_i \in \mathcal{T}_k(\mathcal{D})$). From Lemma 2, g_i is a false negative subgraph if E_i has at least one folded edge in $\hat{\mathcal{D}}$. Recall that $\mathcal{D}$ has $|\mathcal{L}(\mathcal{D})|$ node pairs by Definition 2, and p pairs are now folded in $\hat{\mathcal{D}}$. Thus, each edge in E_i is folded with a probability of $\frac{p}{|\mathcal{L}(\mathcal{D})|}$. That is, g_i is a false negative subgraph with the probability of $1 - (1 - \frac{p}{|\mathcal{L}(\mathcal{D})|})^{m_i}$. Averaging m_i over all subgraphs in $\mathcal{D}$ gives a probability $1 - (1 - \frac{p}{|\mathcal{L}(\mathcal{D})|})^{\overline{m}}$. This completes the proof. □

On the basis of Lemma 4, we can theoretically demonstrate that the subgraph candidates bound the false negative ratio.

Lemma 5. Given subgraph candidates derived from T summarized graph views, the subgraph candidates can bound false negative ratio by $\epsilon \in [0, 1]$ if T satisfies

$$T \geq \left\lceil \frac{\log(\epsilon)}{\log(1 - (1 - \frac{p}{|\mathcal{L}(\mathcal{D})|})^{\overline{m}})} \right\rceil, \tag{6}$$

where $\overline{m} = \frac{1}{N} \sum_{g_i \in \mathcal{D}} \frac{1}{2^{n_i}} \sum_{g_j \subseteq g_i} m_j$.

Proof. Let $P = 1 - (1 - \frac{p}{|\mathcal{L}(\mathcal{D})|})^{\overline{m}}$. Then, Inequation (6) can be rewritten as $T \geq \frac{\log(\epsilon)}{\log(P)}$. Since $\log(P) < 0$, we can derive $P^T \leq \epsilon$. From Definition 5 and Lemma 2, Corgi fails to avoid false negatives only if all summarized graph views yield the false negative top-k search results. Lemma 4 shows that the false negative subgraphs have a probability of P^T. Hence, $P^T \leq \epsilon$ indicates that the false negative ratio of Corgi is bounded by ϵ. This completes the proof. □

Lemma 5 demonstrates that Definition 5 avoids false negatives if T is sufficiently large. Let $\overline{n}$ be the average graph size in $\mathcal{D}$. In practice, $\overline{m} \approx \frac{\overline{n}}{2}$, because molecules generally have $m \approx n$, which gives $\overline{m} \approx \frac{1}{N} \sum_{g_i \in \mathcal{D}} \frac{1}{2^{n_i}} \sum_{k=1}^{n_i} \binom{n_i}{k} k = \frac{\overline{n}}{2}$.

(2) Validation step. Finally, Corgi excludes false positives from the candidate subgraphs. Lemma 3 indicates that the candidate graphs need to be unfolded to filter out the false positives. To this end, we introduce the following definitions:

Definition 6. Given $\hat{g} \in \mathcal{C}$ folded by $\mathcal{A}_p(\mathcal{D})$, the unfolded subgraph of $\hat{g}$ is $g' = (V', E', l')$ such that

$$V' = \hat{V} \cup \{v \,|\, \exists u \in \hat{V}, \langle l(u), l(u, v), l(v) \rangle \in \mathcal{A}_p(\mathcal{D})\}, \tag{7}$$

$$E' = \hat{E} \cup \{(u, v) \,|\, \exists u \in \hat{V}, \langle l(u), l(u, v), l(v) \rangle \in \mathcal{A}_p(\mathcal{D})\}, \tag{8}$$

and l' is a label mapping function of g'

Definition 7. Given an unfolded graph g' and $\mathcal{D}_{g'}$, the unfolded candidate subgraphs of g', which are denoted by $\mathcal{G}(g')$, are defined as $\mathcal{G}(g') = \{g'_1, \ldots, g'_{|D_{g'}|}\}$, where g'_i is an induced subgraph $g_i[V']$ of g_i included in $\mathcal{D}_{g'}$.

Based on Definitions 6 and 7, Corgi initially unfolds every candidate subgraph in $\mathcal{C}$ into g'. Then, it induces all unfolded candidate subgraphs from $\mathcal{D}_{g'}$ by using g'.

From Definitions 6 and 7, we derive the following lemma:

Lemma 6. Let $\hat{g}$ be an abstracted subgraph of $g \subseteq g_i$. If g' is the unfolded subgraph of $\hat{g}$, then g is isomorphic to $g_i[V']$.

Proof. Suppose that $g_i[V'] = (\dot{V}, \dot{E}, \dot{l})$ and $g = (V, E, l)$. Since $g_i[V']$ is a subgraph induced in g_i by V', $\dot{V} = V'$ holds. From Definition 6, $V' = \hat{V} \cup \{v \mid \exists u \in \hat{V}, \langle l(u), l(u, v), l(v) \rangle \in \mathcal{A}_p(\mathcal{D})\} = V$. Hence, $g_i[V'] = g_i[V] = (V, E, l)$. This completes the proof. □

Algorithm 1 (Phase 1) View generation

Input: $\mathcal{D} = \{g_1, g_2, \ldots, g_N\}$, ϵ, p;
Output: $\mathcal{I}$;
1: $\mathcal{I} \leftarrow \emptyset$;
2: $T \leftarrow \lceil \frac{\log(\epsilon)}{\log(1-(1-\frac{p}{|\mathcal{L}(\mathcal{D})|})^{\overline{m}})} \rceil$ by Lemma 5;
3: **for** $t = 1$ to T **do**
4: Obtain $\hat{\mathcal{D}}_t$ by Definition 4;
5: $\mathcal{I} \leftarrow \mathcal{I} \cup \{\hat{\mathcal{D}}_t\}$;
6: **return** $\mathcal{I}$;

Algorithm 2 (Phase 2) MvTk search

Input: $\mathcal{D} = \{g_1, g_2, \ldots, g_N\}$, q, k, $\mathcal{I}$;
Output: $\mathcal{T}_k(\mathcal{D})$;
1: $\mathcal{T}_k(\mathcal{D}) \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset, \mathcal{G} \leftarrow \emptyset$;
2: **If** $\mathcal{I} = \emptyset$ **then** Obtain $\mathcal{I}$ by Algorithm 1;
▷ **(1) Extraction step:**
3: **for each** $\hat{\mathcal{D}}_t \in \mathcal{I}$ **do**
4: $\hat{\mathcal{T}}_k(\hat{\mathcal{D}}_t) \leftarrow \mathrm{CSS}(\hat{\mathcal{D}}_t, \hat{q}_t, k)$ by Definition 5;
5: $\mathcal{C} \leftarrow \mathcal{C} \cup \hat{\mathcal{T}}_k(\hat{\mathcal{D}}_t)$;
▷ **(2) Validation step:**
6: **for each** $\hat{g} \in \mathcal{C}$ **do**
7: $\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{G}(\hat{g})$ by Definitions 6 and 7;
8: **for each** $g \in \mathcal{G}$ **do**
9: Obtain ϕ_k and g_k from $\mathcal{T}_k(\mathcal{D})$;
10: **if** $\phi(q, g) > \phi_k$ **then** $\mathcal{T}_k(\mathcal{D}) \leftarrow (\mathcal{T}_k(\mathcal{D}) \backslash \{g_k\}) \cup \{g\}$;
11: **return** $\mathcal{T}_k(\mathcal{D})$;

Lemma 6 indicates that Corgi can exclude false positives by screening all unfolded candidate subgraphs derived from the candidate subgraphs $\mathcal{C}$ (Definition 7).

3.4 Algorithm

Algorithms 1 and 2 fully describe Corgi. Corgi is divided into two phases: (Phase 1) view generation and (Phase 2) MvTk search. Phase 1 is an *offline* step. It is implemented only once prior to Phase 2. By contrast, Phase 2 is an *online* step, which is executed whenever a query is inputted.

Algorithm 1 constructs T summarized graph views. To estimate a sufficient T value without false negatives, Corgi obtains T from ϵ by Lemma 5 (line 2). Then the views are generated by following Definition 4. Once a query is inputted, Corgi invokes the MvTk search shown in Algorithm 2. In the extraction step, Corgi invokes an existing CSS method denoted by $\mathrm{CSS}(\hat{\mathcal{D}}_t, \hat{q}_t, k)$ on each summarized graph view generated by Algorithm 1. Consequently, Corgi obtains $T \times k$ results (*i.e.,* $\mathcal{C}$). In the validation step, Corgi unfolds $\mathcal{C}$ into $\mathcal{G}$ by Definitions 6 and 7. Afterward, false positives are filtered from $\mathcal{G}$, and Corgi outputs $\mathcal{T}_k(\mathcal{D})$.

Finally, Corgi has the following theoretical properties:

Theorem 1. Corgi incurs $\mathcal{O}((k \times T)N \cdot \overline{n} + T \cdot \mathcal{F}_{\mathrm{CSS}}(N, \overline{n}'))$ time, where $\overline{n}$ and $\overline{n}'$ are the average graph size in $\mathcal{D}$ and $\hat{\mathcal{D}}$, respectively. $\mathcal{F}_{\mathrm{CSS}}(N, \overline{n}')$ is the computational cost of CSS methods on N graphs with an average size $\overline{n}'$.

ID	Dataset	N	$\overline{n}$	$\overline{n}'$	$\overline{d}$	Source
$\mathcal{D}_1$	nci10k	10,000	32.2	10.7	1.01	NCI
$\mathcal{D}_2$	nci20k	20,000	34.6	10.4	1.01	NCI
$\mathcal{D}_3$	nci40k	40,000	35.2	10.2	1.02	NCI
$\mathcal{D}_4$	nci60k	60,000	35.6	10.1	1.02	NCI
$\mathcal{D}_5$	GPCR-A	200,000	31.8	8.63	1.40	ZINC20
$\mathcal{D}_6$	Protease	84,000	33.4	9.22	1.54	ZINC20
$\mathcal{D}_7$	Reductase	20,000	26.1	7.10	1.54	ZINC20
$\mathcal{D}_8$	AA2AR	34,000	28.0	7.73	2.17	DUD-E
$\mathcal{D}_9$	ADBR2	312,500	25.0	6.39	2.16	LIT-PCBA
$\mathcal{D}_{10}$	PKM2	246,069	24.5	6.25	2.16	LIT-PCBA
$\mathcal{D}_{11}$	ALDH1	145,133	23.9	5.98	2.14	LIT-PCBA
$\mathcal{D}_{12}$	MAPK1	62,937	24.1	6.72	2.17	LIT-PCBA

Table 2: Statistics of datasets.

ID	Corgi	bFact	ID	Corgi	bFact
$\mathcal{D}_1$	3.6 sec.	187.1 sec.	$\mathcal{D}_7$	6.5 sec.	2,028 sec.
$\mathcal{D}_2$	8.0 sec.	3,463 sec.	$\mathcal{D}_8$	11.0 sec.	1,938 sec.
$\mathcal{D}_3$	15.5 sec.	$>$ 7,200 sec.	$\mathcal{D}_9$	89.6 sec.	$>$ 7,200 sec.
$\mathcal{D}_4$	25.4 sec.	$>$ 7,200 sec.	$\mathcal{D}_{10}$	72.1 sec.	$>$ 7,200 sec.
$\mathcal{D}_5$	74.3 sec.	$>$ 7,200 sec.	$\mathcal{D}_{11}$	40.5 sec.	$>$ 7,200 sec.
$\mathcal{D}_6$	32.4 sec.	$>$ 7,200 sec.	$\mathcal{D}_{12}$	18.3 sec.	$>$ 7,200 sec.

Table 3: Pre-computation time.

Proof. From Lemma 1, Algorithm 1 incurs $\mathcal{O}(p \cdot \frac{\overline{m}}{\overline{n}})N) \approx \mathcal{O}(N)$ time since $p \cdot \frac{\overline{m}}{\overline{n}} \ll N$. In Algorithm 2, the extraction step repeats the CSS method T times, requiring $\mathcal{O}(T \cdot \mathcal{F}_{\mathrm{CSS}}(N, \overline{n}'))$ time. The validation step computes Definitions 6 and 7 $|\mathcal{C}| = k \times T$ times. Since Definitions 6 and 7 totally require $\mathcal{O}(p + N \cdot \overline{n})$ time, lines 6-7 in Algorithm 2 incur $\mathcal{O}((k \times T)N \cdot \overline{n})$ time. Additionally, lines 8-10 need $\mathcal{O}(N)$ time. Overall, Corgi requires $\mathcal{O}((k \times T)N \cdot \overline{n} + T \cdot \mathcal{F}_{\mathrm{CSS}}(N, \overline{n}'))$ time. □

Corgi has smaller computational costs than other state-of-the-art methods, which require $\mathcal{O}(N 2^{\overline{n}} \overline{n}^2)$ time. In practice, $\overline{n}' \ll \overline{n}$ since Corgi repeatedly folds node pairs to generate the summarized graph view. Specifically, Table 2 shows that the real-world databases examined in the next section have at most $\overline{n}' \leq 10.1$, while $\overline{n} \leq 35.6$. Thus, Corgi improves computational costs compared to existing CSS methods.

Theorem 2. If $\epsilon = 0$ in Algorithm 1, Corgi outputs the same top-k correlated subgraphs as existing CSS methods.

Proof. By Lemma 5, Corgi can avoid false negatives if $\epsilon = 0$. Additionally, from Lemma 6, it can exclude all false positives included in the unfolded candidate subgraphs. Thus, Corgi outputs the same results as existing CSS methods. □

4 Experimental Evaluation

We evaluated the effectiveness of Corgi in a comparison experiment with three types of state-of-the-art CSS methods.

- **Corgi:** Our framework proposed in Section 3. We employed TopCor [Ke *et al.*, 2009] for the CSS method invoked in Algorithm 2 and set $\epsilon = 0.05$ and T to the

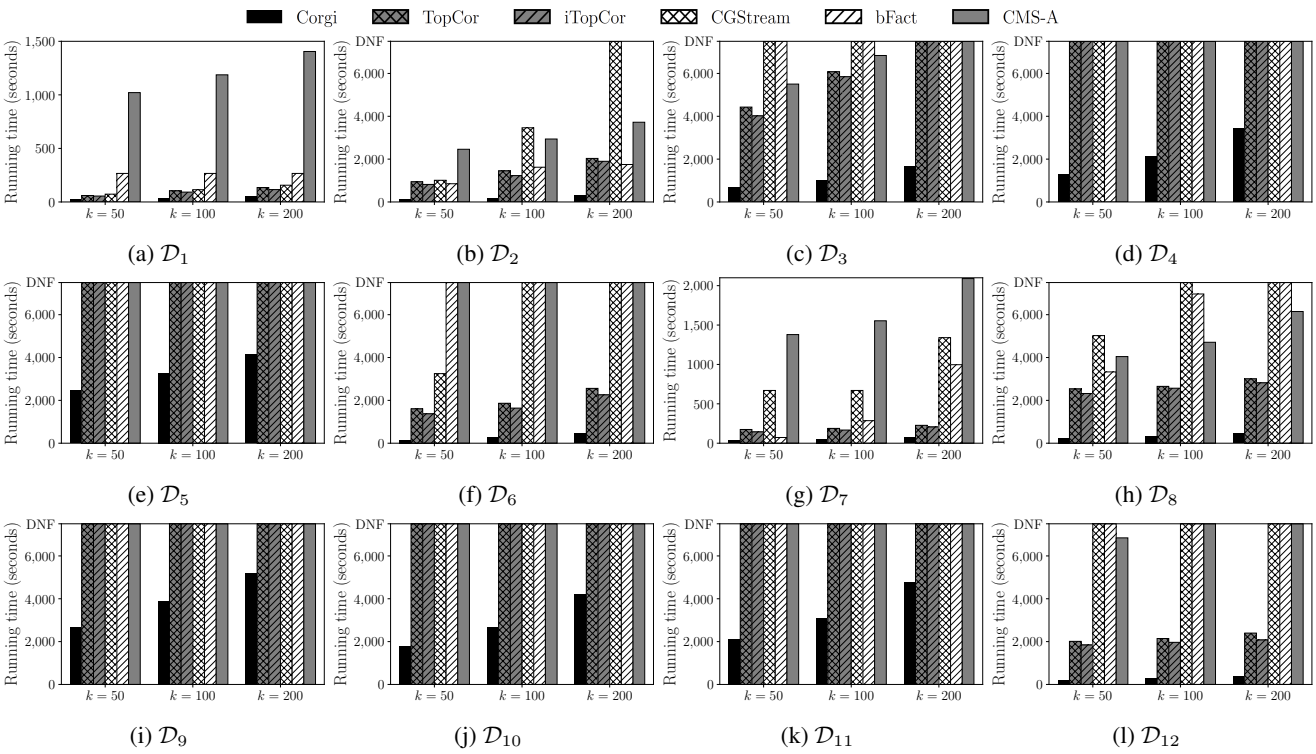


Figure 1: Running time of top-k CSS. "DNF" indicates the method did not finish within 7,200 seconds.

smallest value derived by Lemma 5. For each database, p is set to the largest possible value.

- **Threshold-based methods:** We evaluated **CGStream** [Pan and Zhu, 2012] and **bFact** [Chowdhury *et al.*, 2021], which enumerate all subgraphs having a correlation larger than a threshold θ. By following [Ke *et al.*, 2009], θ is varied from 1.0 to 0.0 in 0.1 decrements to obtain top-k correlated subgraphs.
- **Bounding methods:** We evaluated **TopCor** [Ke *et al.*, 2009] and **iTopCor** [Latsiou and Papadopoulos, 2011], which automatically specify the bounds to output the exact top-k correlated subgraphs.
- **Single-graph methods:** We evaluated **CSM-A** [Prateek *et al.*, 2020], which extracts top-k correlated subgraphs from a single graph. Candidate subgraphs are extracted by applying CSM-A to every graph in the databases, and subsequently filtering false positives.

Evaluations were conducted on a server with an Intel Xeon CPU 2.90 GHz and 1 TiB RAM. All methods were implemented in C/C++ using the "-O3" option.

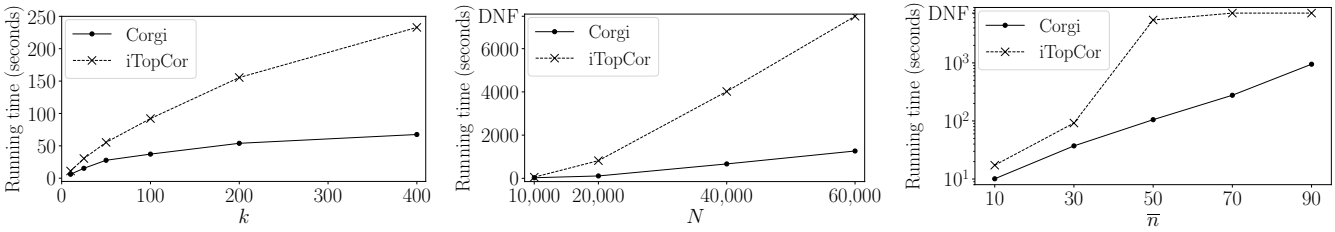
Datasets. We tested 12 public molecule databases published by NCI [Nicklaus *et al.*, 2012], DUD-E [Mysinger *et al.*, 2012], LIT-PCBA [Nguyen *et al.*, 2020], and ZINC 20 [Irwin *et al.*, 2012]. Table 2 shows their statistics, where $\overline{n}$, $\overline{n}'$, and $\overline{d}$ denote the average graph size, the average summarized graph view size, and the average degree, respectively. For more details, please refer Appendix A.

Query. Consistent with [Ke *et al.*, 2009; Prateek *et al.*, 2020], ten queries are generated for each database by randomly selecting subgraphs from the database. The results are averaged over the above ten queries. Following the practical requirement of drug discovery [Garcia-Hernandez *et al.*, 2019; Naoi and Shiokawa, 2023], we regard a query as not finishing if its running time exceeds 7,200 seconds.

4.1 Efficiency

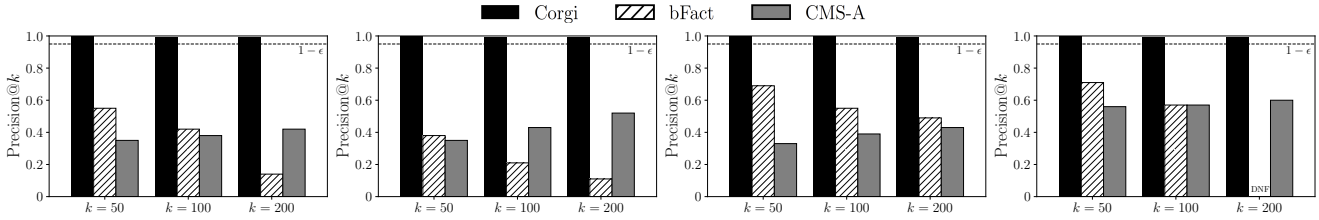
Pre-computation time. As discussed in Section 3, Corgi invokes the view generation phase prior to the MvTk search phase. Here, we assess the pre-computation time by comparing Corgi with bFact because both require the pre-computations. Table 3 shows the running time consumed in the pre-computations, where "$> 7,200$ sec." indicates the corresponding pre-computation did not finish within 7,200 seconds. Corgi is significantly faster than bFact. As proved in Lemma 1, the pre-computation time of Corgi is proportional to N, whereas bFact fails to complete the pre-computation for $N \geq 40,000$. Corgi requires at most 90 seconds, which is negligible compared to the query processing time.

Query processing time. Figure 1 shows the running time of top-k queries on real-world databases, where DNF indicates that the running time exceeded 7,200 seconds. Corgi achieves the highest overall performance. On average, it is 29.8 times faster than the other CSS methods. Not only does Corgi ensure the top-k search accuracy by Lemma 5, but it is also up to 6.6, 5.9, 48.6, 24.9, and 26.1 times faster than TopCor, iTopCor, CGStream, bFact, and CMS-A, respectively. The superior running time is attributed to the summarized



(a) Effects of k. (b) Effects of N. (c) Effects of $\overline{n}$.

Figure 2: Scalability analysis of Corgi. "DNF" indicates that bFact did not finish top-k queries within 7,200 seconds.



(a) $\mathcal{D}_1$ (b) $\mathcal{D}_2$ (c) $\mathcal{D}_7$ (d) $\mathcal{D}_8$

Figure 3: Precision against the ground-truth top-k search results. "DNF" indicates that bFact did not finish top-k queries on $\mathcal{D}_8$ if $k = 200$. The dashed lines represent the lower bound of precision guaranteed by Corgi based on Lemma 5, *i.e.,* $1 - \epsilon = 0.95$.

graph view since it drastically reduces the graph sizes (Section 3.2). Corgi is more efficient than the other methods, even if the dataset has a large $\overline{n}$. As an example, consider the running times on $\mathcal{D}_2$ and $\mathcal{D}_7$ in Figure 1. Recall that $\mathcal{D}_2$ has a larger $\overline{n}$ than $\mathcal{D}_7$ but these two databases have the same number of graphs (*i.e.,* $N = 20,000$). TopCor, iTopCor, CGStream, bFact, and CMS-A show significantly longer running times on $\mathcal{D}_2$ compared with the results of $\mathcal{D}_7$. This is because these methods incur $\mathcal{O}(2^{\overline{n}})$ costs in the worst case. By contrast, Corgi introduces the summarized graph views to help mitigate the exponential costs even if $\overline{n}$ increases. Thus, Corgi can efficiently find top-k subgraphs for large $\overline{n}$.

4.2 Scalability

We experimentally discuss the scalability of Corgi. Figure 2 compares the running time of Corgi with the most competitive method, iTopCor. We varied k as 10, 25, 50, 100, 200, and 400 using $\mathcal{D}_1$ (Figure 2 (a)) and N from 10,000 to 60,000 with $k = 50$ on $\mathcal{D}_4$ (Figure 2 (b)). Corgi achieves a better scalability than iTopCor. iTopCor exponentially increases its running time as N increases due to its high time complexity. By contrast, as we proved in Theorem 1, Corgi achieves a nearly linear scalability against N.

Figure 2 (c) shows the running time ($k = 100$) by varying the average graph size $\overline{n}$ on the synthetic databases. We used GraphGen [Goyal *et al.*, 2020] to generate five databases by varying the size $\overline{n}$ as 10, 30, 50, 70, and 90. We produced $N = 10,000$ graphs with 10 node labels and 4 edge labels, with a density of 0.15. In the figure, Corgi also outperforms iTopCor for all $\overline{n}$ settings. Corgi completes CSS within 1,000 seconds even if $\overline{n} \geq 90$, while iTopCor exceeds 7,200 seconds. Corgi can drastically reduce the average graph size by generating summarized graph views. Hence, it can mitigate the exponential costs incurred by the CSS methods.

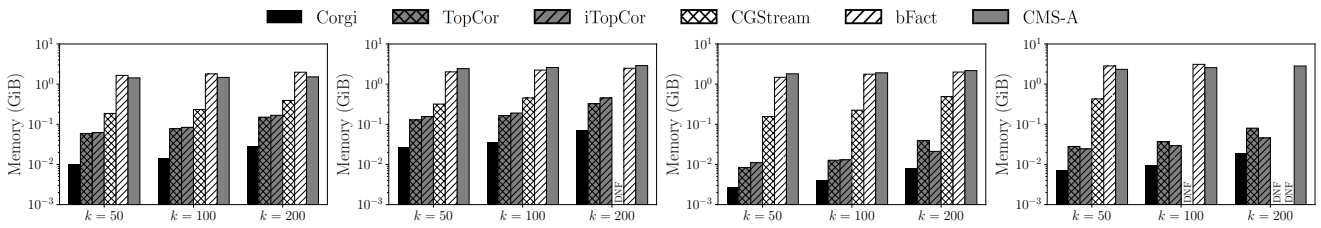
4.3 Accuracy

One advantage of Corgi is that it outputs the top-k correlated subgraphs with accuracy assurance, while folding graphs to improve the runtime efficiency. To confirm this advantage, we evaluated the top-k search accuracy against the ground-truth top-k search results. We used *precision* to measure the accuracy compared with the ground-truth results. Since TopCor, iTopCor, and CGStream output exact top-k subgraphs, their results are regarded as the ground truth.

Figure 3 compares the precision scores of Corgi, bFact, and CMS-A to the ground truth. Similar to the previous section, "DNF" indicates that the top-k queries did not finish within 7,200 seconds. Additionally, the dashed lines represent the accuracy (*i.e.,* $1 - \epsilon = 0.95$), guaranteed by Corgi discussed in Lemma 5. Corgi gives higher precision scores than bFact and CMS-A. Because $\epsilon = 0.05$ for Corgi, it achieves higher precision scores than $1 - \epsilon = 0.95$ for all settings. By contrast, bFact and CMS-A fail to reproduce the ground-truth results because they exclude subgraphs on the basis of thresholds without theoretical guarantees. As proved in Lemma 5, Corgi can bound the false negative ratio by ϵ. In addition, by Lemma 6, its validation step filters all false positives. Therefore, Corgi can find highly accurate top-k subgraphs.

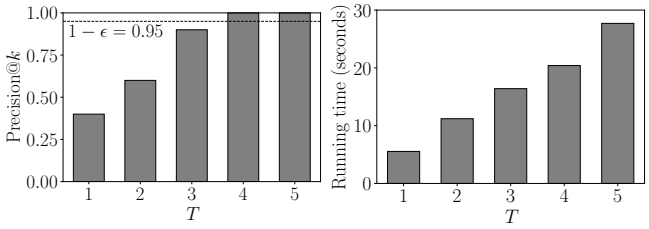
4.4 Memory Footprint

Herein we experimentally discuss the space costs required by Corgi. We measured the memory footprint of Corgi compared with other CSS methods on the real-world datasets. Figure 4 shows the memory footprint of each method, where "DNF" in the figures indicates the corresponding CSS method could not within 7,200 seconds. In this section, we report results only for $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_7$, and $\mathcal{D}_8$ since all methods outputted very similar results.



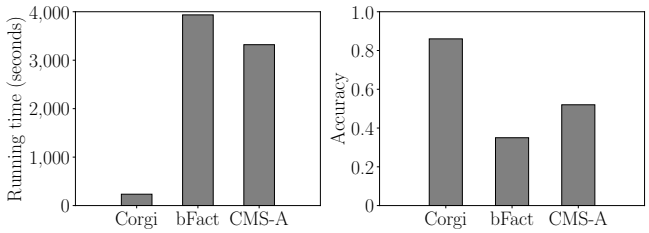
(a) $\mathcal{D}_1$ (b) $\mathcal{D}_2$ (c) $\mathcal{D}_7$ (d) $\mathcal{D}_8$

Figure 4: Memory footprint comparisons among CSS methods. "DNF" in the figures indicates that the corresponding CSS method could not finish within 7,200 seconds. The above results indicate that Corgi outperforms the other CSS methods in terms of the memory footprint, even though it achieves a faster search than the others while ensuring the search accuracy.



(a) Precision. (b) Running time.

Figure 5: Effects of T on $\mathcal{D}_1$ ($k = 50$).



(a) Running time. (b) Accuracy.

Figure 6: Case study.

As shown in Figure 4, Corgi outperforms the other CSS methods in terms of the memory footprint. Specifically, Corgi requires up to a 198 times smaller memory footprint compared to the others. As discussed in Sections 1.1 and 2, existing CSS methods need to enumerate an exponential number of subgraph candidates, which imposes an exponential space cost against graph sizes. By contrast, Corgi drastically reduces the graph size by introducing the summarized graph views. Consequently, our proposed framework successfully mitigates the drastic increase of the memory footprint compared to the other CSS methods.

4.5 Effectiveness of the MvTk Search

We experimentally discuss how the MvTk search effectively avoids false positives and negatives. We measured the precision of the top-k search results of Corgi and their corresponding running time by varying the number of summarized graph views T; T is increased from 1 to the value obtained by Lemma 5 on $\mathcal{D}_1$, and we used $k = 50$ and $\epsilon = 0.05$.

Figure 5 (a) shows that Corgi successfully achieves higher precision than $1-\epsilon$ if $T = 5$, which is derived from Lemma 5. From Lemma 6, Corgi guarantees to exclude all false positives from its top-k search results. That is, Figure 5 (a) indicates that Lemma 5 effectively bounds the false negative ratio. Also, in Figure 5 (b), Corgi has a linear scalability against T. Given T summarized graph views, Algorithm 2 needs to iterate extraction and validation steps T times. Thus, as proved in Theorem 1, Corgi requires a linear running time against T.

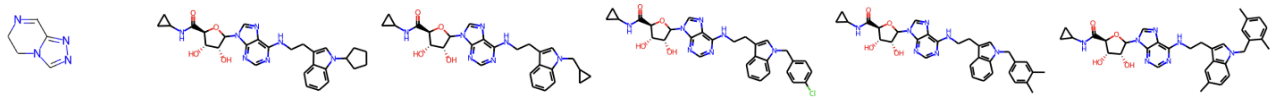
4.6 Case Study

To demonstrate the practicality of Corgi, we deployed Corgi into an LBVS model [Naoi and Shiokawa, 2023] to discover proteins that activate the adenosine A_{2A} receptor (ADORA2A). Because ADORA2A regulates the myocardial oxygen demand and increases coronary circulation, proteins activating ADORA2A are potential therapeutic targets for cardiovascular diseases. We executed Corgi, bFact, and CMS-A on a public database containing over 30,000 proteins [Mysinger *et al.*, 2012]. We inputted a query composed of a common substructure of *adenosine* (Figure 7 (a)), and we set $k = 50$ and $\epsilon = 0.05$. Once the top-k subgraphs were obtained, we chemically tested how many subgraphs actually activated ADORA2A.

Figures 6 (a) and (b) show the running time and accuracy of the LBVS models in this case study, respectively. The accuracy is the fraction of the top-k graphs that activate ADORA2A. Corgi achieves the best accuracy and shortest running time among the CSS methods. Specifically, Corgi discovered 43 of the 50 proteins that activate ADORA2A. Figure 7 (b) shows the top-5 largest proteins activating ADORA2A obtained by Corgi. We observed that all of the top-5 proteins yield the following chemical properties: (1) the H-bond acceptor ratio is larger than 10, and (2) the partition coefficient is greater than 2. These results are reasonable and consistent with the biochemical researchers' intuition because the properties indicate that the proteins are more lipophilic, one of the standard features activating ADORA2A. These results suggest that Corgi can be a powerful option for AI-powered drug discovery.

5 Conclusion

Corgi is a novel CSS framework that accelerates existing CSS methods for drug discovery while ensuring search accuracy.



(a) A query graph. (b) The top-5 largest proteins obtained by Corgi. All proteins activate the adenosine A_{2A} receptor.

Figure 7: Case study in practical drug discovery

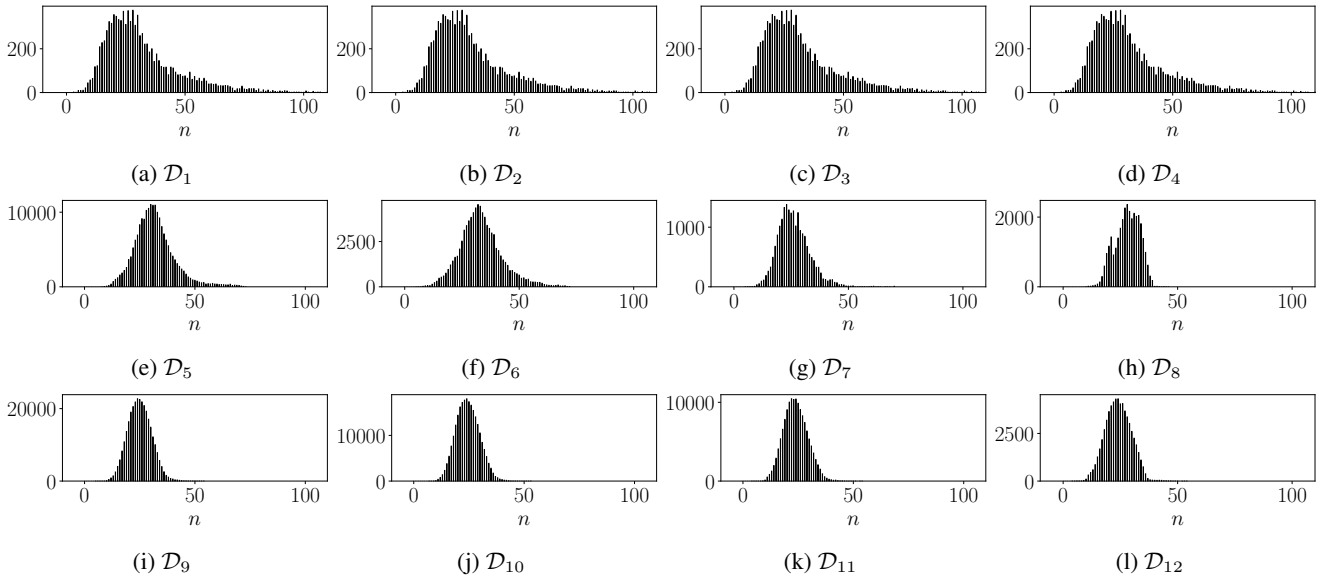


Figure 8: Distribution of the graph sizes.

Corgi constructs summarized graph views to exclude unnecessary computations. Our experiments demonstrate that Corgi improves the efficiency of state-of-the-art CSS methods. Moreover, Corgi successfully discovered beneficial proteins to activate ADORA2A.

A Distributions of Graph Size

In Section 4, we tested 12 datasets that are detailed in Table 2. As discussed in Sections 2 and 3, existing CSS methods require an exponential costs against the graph size in a database. To support the discussions in Section 4, Figure 8 show the distribution of graph sizes in each dataset.

As shown in the figures, most databases have a limited graph sizes, which is at most $n = 65$, except for $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$, and $\mathcal{D}_4$, while NCI datasets yield a long tail distribution. As shown in Figure 1 of the paper, Corgi can complete the query processing within a reasonable time regardless of the graph size distributions. By contrast, in the existing CSS methods such as iTopCor and bFact, their running time is significantly affected by the distribution; if a database has a large graph, they incur a large running time. Unlike existing CSS methods, as discussed in the paper, Corgi can shrink the graph size by introducing the summarized graph views even if databases have a long tail distribution. Hence, it can successfully mitigate the exponential costs incurred by the CSS methods.

B Limitations of Corgi

Corgi still has one limitation in efficiency. Although Corgi has better scalability than other the CSS methods against the average graph size, Corgi still requires a large computation time for the CSS problem if the size significantly increases, *e.g.,* $\overline{n}' \geq 200$. As proved in Theorem 1, Corgi requires at least $\mathcal{O}(\mathcal{F}_{\mathrm{CSS}}(N, \overline{n}'))$ time, where $\mathcal{F}_{\mathrm{CSS}}(N, \overline{n}')$ is the time complexity of existing CSS methods on N graphs with an average graph size $\overline{n}'$. Thus, for a large $\overline{n}'$, Corgi still needs to explore a large search space to solve the CSS problem. However, this limitation does not have a serious impact for our primary focus, *i.e.,* AI-powered drug discovery. This is because that, as shown in Table 2, real-world molecule databases practically have a small average graph size.

Acknowledgments

This paper was partially supported by JST AIP Acceleration Research JPMJCR23U2, JST PRESTO JPMJPR2033, and JSPS KAKENHI 22K17894.

Contribution Statement

H. Shiokawa and Y. Naoi are equally contributed to this paper.

References

[Abdel-Hafiz *et al.*, 2022] Mohamed Abdel-Hafiz, Mesbah Najafi, Shahab Helmi, Katherine A Pratte, Yonghua Zhuang, Weixuan Liu, Katerina J Kechris, Russell P Bowler, Leslie Lange, and Farnoush Banaei-Kashani. Significant Subgraph Detection in Multi-omics Networks for Disease Pathway Identification. *Frontiers in Big Data*, 5:20, 2022.

[Askr *et al.*, 2023] Heba Askr, Enas Elgeldawi, Heba Aboul Ella, Yaseen A. M. M. Elshaier, Mamdouh M. Gomaa, and Aboul Ella Hassanien. Deep Learning in Drug Discovery: An Integrative Review and Future Challenges. *Artificial Intelligence Review*, 56(7):5975–6037, 2023.

[Bhattacharjee and Vlachos, 2020] Himaghna Bhattacharjee and Dionisios G. Vlachos. Thermochemical Data Fusion Using Graph Representation Learning. *Journal of Chemical Information and Modeling*, 60:4673–4683, 2020.

[Chowdhury *et al.*, 2021] Mohammad Ehsan Shahmi Chowdhury, Chowdhury Farhan Ahmed, and Carson K. Leung. A New Approach for Mining Correlated Frequent Subgraphs. *ACM Transactions on Management Information Systems*, 13(1):28, 2021.

[Demir *et al.*, 2022] Andac Demir, Baris Coskunuzer, Yulia R. Gel, Ignacio Segovia-Dominguez, Yuzhou Chen, and Bulent Kiziltan. ToDD: Topological Compound Fingerprinting in Computer-Aided Drug Discovery. In *Proceedings of Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS2022*, 2022.

[Elseidy *et al.*, 2014] Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. Grami: Frequent subgraph and pattern mining in a single large graph. *Proceedings of VLDB Endowment*, 7(7):517–528, 2014.

[Garcia-Hernandez *et al.*, 2019] Carlos Garcia-Hernandez, Alberto Fernández, and Francesc Serratosa. Ligand-Based Virtual Screening Using Graph Edit Distance as Molecular Similarity Measure. *Journal of Chemical Information and Modeling*, 59:1410–1421, 2019.

[Goyal *et al.*, 2020] Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. GraphGen: A Scalable Approach to Domain-agnostic Labeled Graph Generation. In *Proceedings of The Web Conference 2020, WWW2020*, page 1253–1263, 2020.

[Irwin *et al.*, 2012] John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. ZINC: A Free Tool to Discover Chemistry for Biology. *Journal of Chemical Information and Modeling*, 52(7):1757—-1768, 2012.

[Kawano *et al.*, 2024] Keisuke Kawano, Satoshi Koide, Hiroaki Shiokawa, and Toshiyuki Amagasa. Multi-Dimensional Fused Gromov Wasserstein Discrepancy for Edge-Attributed Graphs. *IEICE Transactions on Information and Systems*, E107.D(5):683–693, 2024.

[Ke *et al.*, 2007] Yiping Ke, James Cheng, and Wilfred Ng. Correlation Search in Graph Databases. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD2007*, page 390–399, 2007.

[Ke *et al.*, 2008] Yiping Ke, James Cheng, and Wilfred Ng. Efficient Correlation Search from Graph Databases. *IEEE Transactions on Knowledge and Data Engineering*, 20(12):1601–1615, 2008.

[Ke *et al.*, 2009] Yiping Ke, James Cheng, and Jeffrey Xu Yu. Top-k Correlative Graph Mining. In *Proceedings of the SIAM International Conference on Data Mining, SDM2009*, pages 1038–1049, 2009.

[Latsiou and Papadopoulos, 2011] Georgia S. Latsiou and Apostolos N. Papadopoulos. Incremental Discovery of Top-k Correlative Subgraphs. In *Proceedings of the 15th Panhellenic Conference on Informatics, PCI2011*, pages 56–60, 2011.

[Lee *et al.*, 2014] En-Shiun Annie Lee, Sanderz Fung, Ho-Yin Sze-To, and Andrew K C Wong. Discovering Co-occurring Patterns and Their Biological Significance in Protein Families. *BMC Bioinformatics*, 15(S2):13, 2014.

[Lin *et al.*, 2020] Xuan Lin, Zhe Quan, Zhi-Jie Wang, Tengfei Ma, and Xiangxiang Zeng. KGNN: Knowledge Graph Neural Network for Drug-Drug Interaction Prediction. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2739–2745, 7 2020.

[Lipinski and Hopkins, 2004] Christopher Lipinski and Andrew Hopkins. Navigating Chemical Space for Biology and Medicine. *Nature*, 432:855–861, 2004.

[Ma *et al.*, 2019] Tengfei Ma, Junyuan Shang, Cao Xiao, and Jimeng Sun. GENN: Predicting Correlated Drug-drug Interactions with Graph Energy Neural Networks. *CoRR*, abs/1910.02107, 2019.

[Matsugu *et al.*, 2023] Shohei Matsugu, Yasuhiro Fujiwara, and Hiroaki Shiokawa. Uncovering the Largest Community in Social Networks at Scale. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI2023*, pages 2251–2260, 2023.

[McGibbon *et al.*, 2023] Miles McGibbon, Sam Money-Kyrle, Vincent Blay, and Douglas R. Houston. SCORCH: Improving structure-based Virtual Screening with Machine Learning Classifiers, Data Augmentation, and Uncertainty Estimation. *Journal of Advanced Research*, 46:135–147, 2023.

[Mysinger *et al.*, 2012] Michael M. Mysinger, Michael Carchia, John J. Irwin, and Brian K. Shoichet. Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking. *Journal of Medical Chemistry*, 55(14):6582—-6594, 2012.

[Naoi and Shiokawa, 2023] Yuma Naoi and Hiroaki Shiokawa. Boosting Similar Compounds Searches via Correlated Subgraph Analysis. In *Proceedings of the 25th International Conference on Information Integration and Web Intelligence, iiWAS2023*, pages 464–477, 2023.

[Nguyen and Wei, 2019] Duc Duy Nguyen and Guo-Wei Wei. AGL-Score: Algebrac Graph Learning Score for Protein-Ligand Binding Scoring, Ranking, Docking, and Screening. *Journal of Chemical Information and Modeling*, 59:3291–3304, 2019.

[Nguyen *et al.*, 2020] Viet Khoa Tran Nguyen, Célien Jacquemard, and Didier Rognan. LIT-PCBA: An Unbiased Data Set for Machine Learning and Virtual Screening. *Journal of Chemical Information and Modeling*, 60(9):4263–4273, 2020.

[Nicklaus *et al.*, 2012] Marc Nicklaus, Wolf-Dietrich Ihlenfeldt, Frank Oellien, Igor Filippov, and Markus Sitzmann. NCI Open Database Compounds. https://cactus.nci.nih.gov, 2012.

[Pan and Zhu, 2012] Shirui Pan and Xingquan Zhu. CGStream: Continuous Correlated Graph Query for Data Streams. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM2012*, page 1183–1192, 2012.

[Prateek *et al.*, 2020] Arneish Prateek, Arijit Khan, Akshit Goyal, and Sayan Ranu. Mining Top-k Pairs of Correlated Subgraphs in a Large Network. *Proceedings of VLDB Endowment*, 13(9):1511–1524, 2020.

[Reynolds, 1977] H. T. Reynolds. *The Analysis of Cross-Classifications*. The Free Press, New York, 1977.

[Samiullah *et al.*, 2014] Md. Samiullah, Chowdhury Farhan Ahmed, Anna Fariha, Md. Rafiqul Islam, and Nicolas Lachiche. Mining Frequent Correlated Graphs with A New Measure. *Expert Systems with Applications*, 41(4, Part 2):1847–1863, 2014.

[Shiokawa *et al.*, 2013] Hiroaki Shiokawa, Yasuhiro Fujiwara, and Makoto Onizuka. Fast Algorithm for Modularity-based Graph Clustering. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI2013*, pages 1170–1176, 2013.

[Shiokawa *et al.*, 2019] Hiroaki Shiokawa, Toshiyuki Amagasa, and Hiroyuki Kitagawa. Scaling Fine-grained Modularity Clustering for Massive Graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4597–4604, 2019.

[Welke, 2020] Pascal Welke. Efficient Frequent Subgraph Mining in Transactional Databases. In *Proceedings of 2020 IEEE 7th International Conference on Data Science and Advanced Analytics, DSAA2020*, pages 307–314, 2020.

[Yagi and Shiokawa, 2022] Ryuichi Yagi and Hiroaki Shiokawa. Fast Top-k Similar Sequence Search on DNA Databases. In *Proceedings of the 24th International Conference on Information Integration and Web Intelligence, iiWAS2022*, pages 145–150, 2022.

[Yagi *et al.*, 2023] Ryuichi Yagi, Yuma Naoi, and Hiroaki Shiokawa. Fast Correlated DNA Subsequence Search via Graph-Based Representation. In *Proceedings of the 25th International Conference on Information Integration and Web Intelligence, iiWAS2023*, pages 339–347, 2023.

[Yan and Han, 2002] Xifeng Yan and Jiawei Han. gSpan: Graph-based Substructure Pattern Mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM2002*, pages 721–724, 2002.

[Yang *et al.*, 2022] Tianyi Yang, Li Han, and Shuanghong Huo. Dynamics and Allosteric Information Pathways of Unphosphorylaed c-Cbl. *Journal of Chemical Information and Modeling*, 62:6148–6159, 2022.

[Zhang *et al.*, 2022] Peiliang Zhang, Ziqi Wei, Chao Che, and Bo Jin. DeepMGT-DTI: Transformer Network Incorporating Multilayer Graph Information for Drug–target Interaction Prediction. *Computers in Biology and Medicine*, 142:105214, 2022.

[Zou *et al.*, 2009] Lei Zou, Lei Chen, and Yansheng Lu. Top-K Correlation Sub-graph Search in Graph Databases. In *Proceedings of the 14th International Conference on Database Systems for Advanced Applications, DASFAA2009*, pages 168–185, 2009.