Learning to Bound for Maximum Common Subgraph Algorithms

Buddhi W. Kothalawala
The Australian National University, Canberra, Australia

Henning Koehler
Massey University, Palmerston North, New Zealand

Qing Wang
The Australian National University, Canberra, Australia

Abstract

Identifying the maximum common subgraph between two graphs is a computationally challenging NP-hard problem. While the McSplit algorithm represents a state-of-the-art approach within a branch-and-bound (BnB) framework, several extensions have been proposed to enhance its vertex pair selection strategy, often utilizing reinforcement learning techniques. Nonetheless, the quality of the upper bound remains a critical factor in accelerating the search process by effectively pruning unpromising branches. This research introduces a novel, more restrictive upper bound derived from a detailed analysis of the McSplit algorithm's generated partitions. To enhance the effectiveness of this bound, we propose a reinforcement learning approach that strategically directs computational effort towards the most promising regions within the search space.

2012 ACM Subject Classification Theory of computation → Branch-and-bound; Mathematics of computing → Combinatorial optimization

Keywords and phrases Combinatorial Search, Branch and Bound, Graph Theory

Digital Object Identifier 10.4230/LIPIcs.CP.2025.22

Supplementary Material *Software (Source Code)*: `https://github.com/BuddhiWathsala/mcsplit-dsb`, archived at `swh:1:dir:7bbe80724d5f242e89478b8ab3956ac6a35d2daa`

Funding This research was supported partially by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP210102273).

Acknowledgements We are thankful to Dr Muhammad Farhan for helpful discussions and feedback.

1 Introduction

Graphs are widely employed in various real-world applications due to their effectiveness in modeling complex structures. A significant challenge in these applications is identifying common patterns between graphs. Given two graphs, G and H, the task of finding two induced subgraphs from G and H that are isomorphic and contain the maximum number of vertices is known as the *Maximum Common Induced Subgraph (MCIS)* problem [7]. For simplicity, the MCIS problem is also referred to as the maximum common subgraph problem. The MCIS problem is NP-hard [2], making it computationally challenging. Despite its complexity, it has broad applications across multiple fields. In chemical analysis, it helps to group chemical compounds based on their structural features [6, 9]. In bioinformatics, it facilitates the comparison of molecular structures, supporting drug discovery and genomic analysis [8]. MCIS is also crucial in graph similarity measures, which are commonly used for anomaly detection in networks [1], as well as in pattern recognition tasks like video analysis and image processing [20].

Several exact [16] and approximate algorithms [19] have been developed to solve the MCIS problem. The exact algorithms employ various techniques such as constraint programming [25, 18], maximum clique algorithms [11, 10, 4], and the branch-and-bound (BnB) approach [16, 24, 17]. Among these, the branch-and-bound algorithm called McSplit, introduced by McCreesh et al. [16], has shown exceptional performance. The McSplit algorithm effectively tackles the MCIS problem using two essential heuristics: a branching heuristic that selects optimal vertex pairs for matching and a bounding heuristic that estimates the maximum possible size of the common subgraph within a given search branch. For branching, McSplit prioritizes vertices with higher degrees when choosing vertex pairs, while its bounding heuristic calculates an upper bound by counting the number of potential matching vertex pairs.

Several machine learning-based improvements have been introduced to enhance McSplit's branching heuristic using reinforcement learning (RL) techniques, all while maintaining its exactness. Prominent examples include McSplit+RL [12], McSplit+LL [27], GLSearch [3], and McSplit+DAL [13]. McSplit+RL selects vertex pairs with a high potential to significantly reduce the bound, using bound reduction as the RL reward. McSplit+LL extends this by incorporating both long-term and short-term rewards in selecting the next vertex pair and enables the simultaneous matching of leaf vertices to improve search efficiency. McSplit+DAL further enhances McSplit+RL's reward mechanism by prioritizing vertex pairs that simplify the MCIS problem. It also diversifies vertex pair selection using a hybrid branching policy that alternates between McSplit+RL and McSplit+DAL. In contrast, GLSearch leverages a deep Q-network to identify the next vertex pair, focusing on maximizing the size of the common subgraph rather than simply ending the search quickly, as done in McSplit+RL, McSplit+LL, and McSplit+DAL.

While previous research on enhancing the McSplit algorithm in reinforcement learning has largely focused on optimizing branching strategies, this paper introduces a new perspective on improving upper bound estimation. We observed that analyzing the internal structure of matching pairs can provide a tighter upper bound, allowing for earlier pruning of search branches compared to the current bound calculation. The key contributions of this paper are as follows:

1. **A Tighter Upper Bound for MCIS:** We establish a tighter upper bound for the MCIS problem by utilizing structural graph properties, such as edge cardinality and degree sequence, to improve pruning efficiency. Since the new bound is theoretically valid, it preserves the exactness of the MCIS search.
2. **Reinforcement Learning to Applying the Bound:** We introduce a RL framework that dynamically prioritizes bound computation in high-potential search regions, reducing redundant calculations while maintaining solution quality.
3. **Empirical Performance Gains:** We empirically demonstrate that improvements after incorporating the proposed bound improves McSplit and its extensions in solving the MCIS problem.
4. **Achieving Near-Maximal Bound Reduction:** We empirically show that our bound closely approximates the theoretically maximum achievable bound reduction in McSplit, validating its near-optimal pruning effectiveness.

We introduce a novel algorithm, McSplit+DSB (Degree Sequence Bound), which replaces the original bound calculation of the McSplit algorithm proposed by McCreesh et al. [16]. The proposed bound calculation is also adaptable to other RL-based McSplit extensions. Building on the reinforcement learning extensions McSplit+RL, McSplit+LL, and McSplit+DAL, we present McSplit+RL+DSB, McSplit+LL+DSB, and McSplit+DAL+DSB, which incorporate

Table 1 Comparison of branching and bounding heuristics across different McSplit variants. The table highlights the selection strategies for bidomains, vertices, and matches in various methods, as well as the proposed modification to the bounding heuristic in our approach (McSplit+DSB), which incorporates an additional reduction term δ to enhance pruning efficiency.

Method	Branching Heuristics			Bounding Heuristics
	Bidomain Selection	Vertex Selection	Match Selection	
McSplit	$max(\|V_l\|, \|U_l\|) \downarrow$	$d_G(v) \downarrow$	$d_H(u) \downarrow$	$\sum_{\langle V_l, U_l \rangle \in P_{GH}} \min(\|V_l\|, \|U_l\|)$
McSplit+RL	no change	$S_G(v) \downarrow$	$S_H(u) \downarrow$	no change
McSplit+LL	no change	$S_G(v)$ with decay $\downarrow$	$S_{GH}(v,u)$ with decay $\downarrow$	no change
McSplit+DAL	no change	$S'_G(v)$ with decay $\downarrow$	$S'_{GH}(v,u)$ with decay $\downarrow$	no change
McSplit+DSB (Ours)	no change	no change	no change	$\sum_{\langle V_l, U_l \rangle \in P_{GH}} \min(\|V_l\|, \|U_l\|) - \delta$

our new bound calculation into these existing algorithms. To assess the effectiveness of these approaches, we conducted extensive experiments on benchmark datasets for the MCIS problem.

The paper is structured as follows: Section 2 introduces the notations and definitions. Section 3 describes the McSplit algorithm. Section 4 details our degree sequence bound calculation method and the reinforcement learning approach designed to integrate it into the search process. Section 5 presents the experimental evaluation, where the proposed method is compared with state-of-the-art techniques. Finally, Section 6 provides concluding remarks.

2 Preliminaries

Let G be a simple and unweighted graph, where $V(G)$ is the set of vertices and $E(G)$ is the set of edges. For a vertex $v \in V(G)$, the *neighborhood* of v is defined as $N(G, v) = \{u \in V(G) \mid (u, v) \in E(G)\}$. Given a subset of vertices $V' \subseteq V(G)$, we use $G[V']$ to denote the *induced subgraph* of G, which contains all vertices in V' and edges in $\{(u, v) \in E(G) \mid \{u, v\} \subseteq V'\}$. To simplify, we use $G[V'] \subseteq G$ to indicate that $G[V']$ is an induced subgraph of G. Let G be a graph with $V(G) = n$ and $d_G(v)$ denote the degree of a vertex v in G. Then, the degree sequence of G is a non-decreasing sequence of the degrees of all its vertices, i.e. $Deg(G) = (d_G(v_1), \ldots, d_G(v_n))$, and the degree sum of G is defined as $d(G) = \sum_{i=1}^{n} d_G(v_i)$.

Two graphs G and H are *isomorphic*, denoted as $G \cong H$, if there exists a bijection $f : V(G) \to V(H)$ such that $(v_1, v_2) \in E(G) \Leftrightarrow (f(v_1), f(v_2)) \in E(H)$. Two induced subgraphs $G[V']$ and $H[U']$ are called *common induced subgraphs* of G and H iff $G[V'] \cong H[U']$. We use $\bar{G}$ to denote the *complement graph* of a graph G. If $G \cong H$, then $\bar{G} \cong \bar{H}$, and vice versa.

▶ **Definition 1.** *Given two graphs G and H, the* maximum common induced subgraph *(MCIS) problem is to find two induced subgraphs $G[V'] \subseteq G$ and $H[U'] \subseteq H$ such that $G[V'] \cong H[U']$ and there exist no common induced subgraphs of G and H that have a size larger than $|V'|$.*

Without loss of generality, throughout the paper, we consider two input graphs G and H where $|V(G)| \leqslant |V(H)|$, with $|V(G)| = n$ and $|V(H)| = m$.

3 Branch and Bound for MCIS

The branch-and-bound algorithm demonstrates significantly better performance than other algorithms for the MCIS problem. It typically employs a depth-first search, starting with an empty subgraph and adding matching vertex pairs, while partitioning the remaining

vertices based on their connectivity to the selected ones. In the BnB approach to MCIS, the branching strategy is based on selecting the next matching vertex, and the bound is calculated by estimating the number of matching pairs that can still be added to the current common subgraph.

3.1 McSplit: Partitioning-Based Algorithm

McCreesh et al. [16] propose a BnB algorithm to solve the MCIS problem, called McSplit. Given two graphs G and H, McSplit solves the MCIS problem by finding a maximum-cardinality mapping $M^* = \{(v_1, u_1), \ldots, (v_m, u_m)\}$, where all $v_i \in V(G)$ and $u_i \in V(H)$ are distinct from one another and satisfy the condition $(v_i, v_j) \in E(G) \Leftrightarrow (u_i, u_j) \in E(H)$ for any v_i, v_j with $1 \leqslant i \neq j \leqslant m$. Each vertex pair $(v_i, u_i) \in M^*$ is called a *matching vertex pair*. Such a maximum-cardinality mapping corresponds to the maximum common subgraphs $G[V^*]$ and $H[U^*]$ of G and H, where $V^* = \{v_1, \ldots, v_m\}$ and $U^* = \{u_1, \ldots, u_m\}$.

To find a maximum-cardinality mapping between G and H, McSplit employs a depth-first search, starting with the empty mapping. It then iteratively extends the mapping with matching vertex pairs, adding one pair at each iteration until no further extensions are possible. The key idea of finding matching vertex pairs is based on bidomain partitioning. This process uses a labelling function that assigns labels to unmatched vertices based on their connectivity with matched vertices.

▶ **Definition 2** (Labelling). *Let $M = \{(v_1, u_1), \ldots, (v_k, u_k)\}$ be a (not necessarily maximum-cardinality) mapping between two graphs G and H, with $V' = \{v_1, \ldots, v_k\}$ and $U' = \{u_1, \ldots, u_k\}$. A labeling function $\ell : V(G) \to \{0,1\}^k$ assigns a label to each vertex $w \in V(G)\backslash V' \cup V(H)\backslash U'$ such that the i^{th} bit of $\ell(w)$ is equal to 1 if $(w, v_i) \in E(G)$ or $(w, u_i) \in E(H)$; otherwise, it is 0.*

Based on the labels, McSplit partitions unmatched vertices in $V(G)\backslash V' \cup V(H)\backslash U'$ into a set of bidomains $P_{GH} = \{\langle V_1, U_1\rangle, \ldots, \langle V_t, U_t\rangle\}$ such that $\forall v \in V_i$ and $\forall u \in U_j$, $i = j \Leftrightarrow \ell(v) = \ell(u)$, ensuring that all vertices within a single partition share the same label, while vertices in different partitions have distinct labels.

Specifically, the McSplit algorithm starts by initializing $P_{GH} = \{\langle V(G), V(H)\rangle\}$ and picks a matching bidomain pair $\langle V_l, U_l\rangle$. Then, the algorithm selects a vertex $v \in V_l$ and v is matched against all the vertices $u \in U_l$. Matching v with any u gives $M = \{(v, u)\}$. Then, unmatched vertices in G and H are labelled based on the connectivity of v and u, respectively. This leads to bidomain re-partitioning such that each bidomain $\langle V_l, U_l\rangle \in P_{GH}$ is partitioned into two bidomains $\langle V_l \cap N(G, v), U_l \cap N(H, u)\rangle$ and $\langle V_l\backslash N(G, v), U_l\backslash N(H, u)\rangle$. Iteratively, the algorithm selects another vertex pair to expands M. Once v is matched against all the vertices $u \in U_l$, the algorithm removes the vertex v from V_l and continues with the search. The removal of v is denoted as matching v with the special symbol $\perp$. The algorithm continues until there are no more matching vertex pairs to select. In every iteration, alongside the current mapping M being expanded, the algorithm keeps track of the best solution discovered so far, referred to as the *incumbent*. It then uses the following upper bound to eliminate unpromising search branches.

$$UB \leftarrow |M| + \sum_{\langle V_l, U_l\rangle \in P_{GH}} \min(|V_l|, |U_l|) \tag{1}$$

If the current mapping M cannot be expanded more than *incumbent* in size, i.e., $UB \leqslant |incumbent|$, the algorithm immediately terminate the search, otherwise continues.

3.2 McSplit + Reinforcement Learning

Recent studies extend McSplit using reinforcement learning techniques [12, 27, 13], where an agent explores an environment by taking actions [23]. All these studies consider the McSplit algorithm as an agent whose goal is to reach a search tree leaf as early as possible to reduce the overall size of the search tree. The choice of a matching vertex pair (v, w) at each branching step is modelled as an action. This viewpoint naturally leads to using the bound reduction as a crucial factor for defining the reward of the action (v, w), and then incorporating accumulated rewards into heuristics for selecting matching vertex pairs.

3.2.1 McSplit+RL

McSplit+RL [12] extends McSplit by selecting a matching vertex pair (v, u) using a learning-based method. The main idea behind McSplit+RL is to define the reward for an action (v, u) based on how much *bound* is reduced after selecting (v, u) as the matching vertex pair. The value function maintains two score lists $S_G(v)$ and $S_H(u)$ for all $v \in V_l$ and $u \in U_l$ during the search. Then, at each branch step, a bidomain $\langle V_l, U_l \rangle$ is selected in the same way as McSplit. A vertex $v \in V_l$ with the highest score $S_G(v)$ is chosen and matched with a vertex $u \in U_l$ in decreasing order of $S_H(u)$. These together determine the choice of a matching vertex pair (v, u) in McSplit+RL.

3.2.2 McSplit+LL

McSplit+LL [27] further extends McSplit+RL with two additional operators: 1) long-short memory, and 2) leaf vertex union match. Essentially, McSplit+LL defines the reward and maintains the score list $S_G(v)$ in the same way as McSplit+RL. However, it uses a score list $S_{GH}(v, u)$ for matching vertex pairs, which is also arranged in decreasing order. Further, the scores of $S_G(v)$ and $S_{GH}(v, u)$ are decayed to a half when a matching vertex pair reaches their thresholds, respectively. When a matching vertex pair (v, w) is chosen, McSplit+LL allows to match as many unmatched leaf vertex pairs as possible simultaneously to expand the mapping M.

3.2.3 McSplit+DAL

McSplit+DAL [13] attempts to address two limitations of McSplit+RL: 1) bidomain simplification relating to different matches are not considered, and 2) vertices with high accumulated rewards are repeatedly chosen during backtracking. It defines a new reward for an action (v, u) by adding $|P_{GH}|$ to the original reward of McSplit+RL, which encourages the reward is given to more simplified states. Based on the new reward, two score lists $S'_G(v)$ and $S'_{GH}(v, u)$ are defined in a similar way as $S_G(v)$ and $S_{GH}(v, u)$ in McSplit+LL. A hybrid strategy that combines its proposed heuristic with the heuristic of McSplit+RL is also introduced to diversify the search space.

3.3 Limitations of Existing Algorithms

Within the branch and bound framework, search heuristics used by algorithms for solving the MCIS problem can be generally categorized into two types:

- *Branching heuristic*: This typically involves *partition selection* (selecting a bidomain $\langle V_l, U_l \rangle$ from P_{GH}), *node selection* (selecting a vertex v from V_l), and *match selection* (selecting a vertex u from U_l to form a matching vertex pair (v, u)).

- *Bounding heuristic*: This estimates the maximum number of vertices (i.e., upper bound) in the maximum common subgraph. If this estimated upper bound is less than the found common subgraph, it prunes that search branch.

Currently, all existing learning-based methods for McSplit focus on improving branching heuristics, which essentially involves deciding the "best" matching vertex pair to branch out, assuming that choosing different matching vertex pairs may lead to different bound reductions and thus reach the pruning condition at different speeds. They all use the bounding heuristic of McSplit that estimates the upper bound based on $\sum_{\langle V_l, U_l \rangle \in P_{GH}} \min(|V_l|, |U_l|)$ (see Equation (1)).

Observation. The bounding heuristic of McSplit is too generous. For instance, if $G[V_l]$ is a clique of five vertices (K_5) and $H[U_l]$ is the complement graph ($\bar{K_5}$), the size of their maximum common subgraphs is one, which is significantly smaller than $\min(|V_l|, |U_l|) = 5$ estimated in the bound.

This observation motivates us to develop a learning-based bounding heuristic that can adaptively estimate the bound by considering the underlying graph structure. Table 1 summarizes branching and bounding heuristics used in McSplit and the existing learning-based extensions.

4 Proposed Method

We propose a novel bounding heuristic with two synergistic components: (1) a tight upper bound calculation leveraging degree sequences and partition degree sums, and (2) a reinforcement learning framework that strategically deploys this bound in high-impact regions of the BnB search space. Before introducing our heuristic, we analyze existing approaches to motivate the research problem and discuss the unique challenges in developing learning-based bounding heuristics.

Section 4.1 analyzes the key challenges in developing a novel bound calculation method. Section 4.2 presents the theoretical foundation of our proposed bound, supported by practical examples. Finally, Section 4.3 introduces the reinforcement learning (RL) framework designed to effectively integrate this new bounding strategy.

4.1 Challenges

We consider the McSplit algorithm or any of its extensions McSplit+RL, McSplit+LL, and McSplit+DAL as an agent. Instead of reaching a search tree leaf as early as possible, the goal of our agent is to prune branches as early as possible. Thus, unlike existing methods where an action is to choose the best matching vertex pair, an action in our method is to decide whether to employ a new and tighter upper bound to prune a branch earlier. However, there are two major challenges in designing a learning-based bounding heuristic in this setting: (1) How to design an efficient algorithm that can provide a tighter upper bound than the original bound of McSplit? (2) How to design the reward and value function that can accurately predict when a pruning condition can be met under new and tighter bounds? Addressing these two challenges are the key to enhancing the overall performance of the algorithm. We discuss our approach below in detail.

4.2 Degree Sequence Bound

We begin by proposing a novel upper bound for MCIS, which is tighter than the bound of McSplit and can be used to reduce the search space for BnB algorithms based on McSplit. The intuition behind this novel upper bound is simple. Given a bidomain $\langle V_l, U_l \rangle$, we derive an upper bound for maximum common subgraphs between $G[V_l]$ and $H[U_l]$ based on their degree sequences, not merely the sizes of $|V_l|$ and $|U_l|$ as used by $\min(|V_l|, |U_l|)$ in McSplit. This is because the degree sequence of a graph can provide more useful structural information than computing the size of the small graph.

Let $M^*(G, H)$ denote a maximum-cardinality mapping between two graphs, G and H. An *independent set* is a set of pairwise non-adjacent vertices. Based on degree sequences, we derive the following (recall that degree sequences are monotonically increasing).

▶ **Lemma 3.** *For a graph G with degree sequence $Deg(G) = (d_G(v_1), \ldots, d_G(v_k), \ldots, d_G(v_n))$, the degree sum of any induced subgraph on k vertices is bounded from below by*

$$\sum_{i=1}^{k} d_G(v_i) - \sum_{j=k+1}^{n} d_G(v_i)$$

If $v_{k+1}, \ldots, v_n$ form an independent set, then this bound is tight.

Proof. Any induced subgraph $G[V_k]$ on k can be obtained by removing $n-k$ distinct vertices from G in some arbitrary order. Removing a vertex v_i reduces the degree-sum by precisely twice the degree of v_i in the current subgraph. The degree of v_i in any subgraph of G is upper-bounded by $d_G(v_i)$, so the degree-sum of the $n-k$ vertices removed to obtain $G[V_k]$ is upper-bounded by $\sum_{j=k+1}^{n} d_G(v_i)$. This shows the lower bound claim.

If $v_{k+1}, \ldots, v_n$ form an independent set, then removing some of them from G will not affect the degrees of others. Thus the bound is tight for $G[V_k] = G[v_1, \ldots, v_k]$. ◀

▶ **Theorem 4.** *Let $|V(G)| \leqslant |V(H)|$ and $Deg(G) = (d_G(v_1), \ldots, d_G(v_n))$. If*

(1) $\sum_{i=1}^{k} d_G(v_i) - \sum_{j=k+1}^{n} d_G(v_j) > d(H)$, *or*

(2) $\sum_{j=n-k-1}^{n} d_{\bar{G}}(v_j) - \sum_{i=1}^{n-k} d_{\bar{G}}(v_i) > d(\bar{H})$

then $|M^*(G, H)| \leqslant k-1$, where $d_{\bar{G}}(v_i) = |V(G)| - d_G(v_i) - 1$ for $i = 1, \ldots, n$.

Proof. From Lemma 3, we know that the expression,

$$\sum_{i=1}^{k} d_G(v_i) - \sum_{j=k+1}^{n} d_G(v_j)$$

represents lower bound for the degree sum of an induced subgraph $G[V'] \subseteq G$ with k vertices. Condition (1) implies that the degree sum of any induced subgraph $G[V_k]$ with k vertices exceeds the degree sum $d(H)$ of graph H. In particular $d(G[V_k])$ exceeds the degree-sum of any (induced) subgraph H_k of H, and thus cannot be isomorphic to it. This proves the first statement. Similarly, by arguing on complement graphs $\bar{G}$ and $\bar{H}$, we can prove the second statement. ◀

In light of Theorem 4, when $k \in [1, n]$ satisfies the property of being the smallest number satisfying Case (1) or Case (2), we call k the *dividing number* of the graphs G and H. It can be proven that Case (1) and Case (2) cannot be satisfied simultaneously, not even for different values of k, and that they can only be satisfied for $k > n/2$, so there exists at most one dividing number $k \in (n/2, n]$ between G and H.

Obviously, the bound in Theorem 4 is tighter than the bound of McSplit. To characterize the impact of Theorem 4 on the upper bounds of maximum common subgraphs of G and H, we define the *bound gap* δ_{GH} to be the difference between the bound of McSplit and the new bound in Theorem 4. Let k be the dividing number of G and H. Then $\delta_{GH} = n - k + 1$ if k satisfies Case (1) or Case (2). When there does not exist any dividing number $k \in (n/2, n]$ satisfying Case (1) or Case (2), we set $\delta_{GH} = 0$. This leads to the following corollary.

▶ **Corollary 5.** *For any bidomain $\langle V_l, U_l \rangle \in P_{GH}$, the size of maximum common subgraphs of $G[V_l]$ and $H[U_l]$ is upper bounded by $\min(|V_l|, |U_l|) - \delta$ where δ (the subscript is omitted) refers to the bound gap of $G[V_l]$ and $H[U_l]$.*

Proof. From Theorem 4, we know the δ is a valid reduction that we can apply to obtain a more restricted bound for $\langle V_l, U_l \rangle$. So, in McSplit bound in Equation 1, we can replace $\min(|V_l|, |U_l|)$ with $\min(|V_l|, |U_l|) - \delta$. Hence, the corollary is proved. ◀

Based on Corollary 5, we define the following new bound:

$$UB_{dsb} \leftarrow |M| + \sum_{\langle V_l, U_l \rangle \in P_{GH}} \min(|V_l|, |U_l|) - \delta \tag{2}$$

We shall refer to this new bound as the *Degree Sequence Bound (DSB)*.

McSplit is an exact algorithm that guarantees the retrieval of the maximum common subgraph upon completion. Given that δ provides a valid estimate for the maximum common subgraph of V_l and U_l as stated in Theorem 4, incorporating the proposed bound from Equation (2) preserves the exactness of the McSplit algorithm.

4.2.1 Further Insights

In this section, we further explore the theoretical foundations of Theorem 4, examining the key properties of its conditions. In particular, we demonstrate that the two cases presented in the theorem are mutually exclusive and analyze the feasibility of meeting these conditions based on the parameters n and k. These insights offer a deeper understanding of the theorem.

▶ **Lemma 6.** *Cases (1) and (2) of Theorem 4 are mutually exclusive.*

Proof. Assume both conditions hold. From (1) it follows that $d(G) > d(H)$, while (2) implies that $d(\bar{G}) > d(\bar{H})$. Thus we have $d(G) + d(\bar{G}) > d(H) + d(\bar{H})$, contradicting $|V(G)| \leqslant |V(H)|$. ◀

▶ **Lemma 7.** *Conditions (1) and (2) of Theorem 4 cannot be satisfied for $k \leqslant n/2$. For every (n, k) with $k > n/2$ there exist graph pairs (G_1, H_1) and (G_2, H_2) satisfying conditions (1) and (2).*

Proof. For $k \leqslant n/2$ the lower bounds in (1) and (2) become non-positive, and thus cannot be strictly greater than $d(H)$.
For $k > n/2$ let G_1 be a cycle of length n, so that all vertices have degree two, and H_1 the empty graph. Then the left hand side of the inequality in (1) remains strictly positive, so the condition in (1) is met. For (2) let $(G_2, H_2) = (\bar{G}_1, \bar{H}_1)$. ◀

4.2.2 Complexity Analysis

McSplit has a time complexity of $O(n+m)$ for calculating the bound in Equation (1). The new bound requires $O(m^2)$ time because the computation of $d(H)$ take $O(m^2)$ and the computation of $Deg(G)$ takes $O(n^2 + n \log n)$, where $n \leqslant m$. Since calculating the new bound is asymptotically dominated by the quadratic term m^2, it is desirable to impose a limit of $\sigma_{\max}$ on the size of $|V_l|$ and $|U_l|$ to maintain efficiency, and use the original bound if the limit is exceeded. This constraint is applied to individual bidomains rather than the entire input graph. When appropriately chosen (we used $\sigma_{max} = 16$), this limit does not significantly affect pruning power, as Theorem 4 is more effective with smaller bidomains.

4.3 Learning-based Bounding Heuristic

Since computing the new bound requires additional computational time compared to McSplit bound, it is essential to determine where the new bound can be most effectively utilized during the search - particularly in areas where there is a high likelihood of pruning branches. To address this, we propose incorporating a reinforcement learning agent into a learning-based framework. This agent will dynamically decide when to apply the proposed tighter bound, aiming to maximize both the efficiency and effectiveness of the pruning process.

We denote power sets by $\mathcal{P}(\cdot)$. Let ϕ be a property of bidomains, e.g., given a bidomain $\langle V_l, U_l \rangle$, ϕ may hold if $|V_l|$ or $|U_l|/|V_l|$ is smaller than a threshold. Then a *ϕ-activation function* is defined as $\lambda : \mathcal{P}(V(G)) \times \mathcal{P}(V(H)) \rightarrow \{active, inactive\}$ such that $\lambda(V', U') = active$ if $\langle V', U' \rangle$ satisfies the property ϕ. At each branching step, P_{GH} contains a subset of active bidomains $P^{\dagger}_{GH} = \{\langle V_l, U_l \rangle \in P_{GH} \mid \lambda(V_l, U_l) = active\}$.

The reward for an action on P_{GH} is based upon the reward for each active bidomain $\langle V_l, U_l \rangle \in P_{GH}$. This is because the estimated bound gap between the bound of McSplit and the new bound for P_{GH} depends on the estimated bound gaps for its active bidomains. Thus, we define the reward for a bidomain as the bound gap between the bound of McSplit and the new bound.

$$R(V_l, U_l) = \delta_{G[V_l]H[U_l]} \tag{3}$$

The value function of a bidomain $\langle V_l, U_l \rangle$ maintains a score $S(V_l, U_l)$, which is initialized to 1 and is then averaged with the previous score each time to capture the historical information. Let $\alpha \in [0, 1]$. When α is close to 1, more weight is given to recent bound reductions (short-term), whereas when α is close to 0, the bound reductions over the entire history (long-term) are given more weight.

$$S(V_l, U_l) \leftarrow (1 - \alpha) S(V_l, U_l) + \alpha R(V_l, U_l) \tag{4}$$

The value function for the set of bidomains maintains a score $S(P_{GH})$ for P_{GH} which reflects the accumulated rewards of its active bidomains.

$$S(P_{GH}) \leftarrow \sum_{\langle V_l, U_l \rangle \in P^{\dagger}_{GH}} S(V_l, U_l) \tag{5}$$

At each branching step, the algorithm checks the following condition to decide whether to compute the new bounds:

$$UB - S(P_{GH}) \leqslant |incumbent| \tag{6}$$

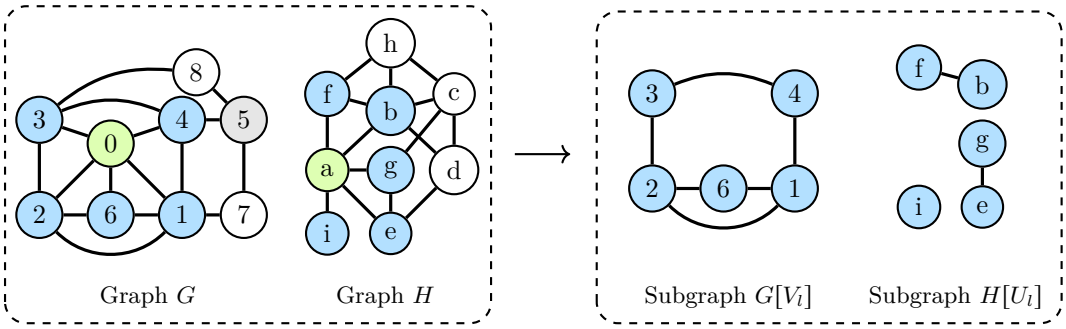
The intention of the above condition is to predict whether computing the new bound can lead to prune the branch. Thus, if the condition is satisfied, the new bound is computed for each active bidomain, and the reward and value scores are updated; otherwise the algorithm uses the bound of McSplit without any changes on rewards and value scores.

Algorithm 1 McSplit+DSB.

```
1  function DSB(P_GH, M):
2      UB_dsb ← |M| + Σ_{⟨V_l,U_l⟩∈P_GH} min(|V_l|, |U_l|)
3      P†_GH ← select active bidomains using φ
4      S(P_GH) ← Σ_{⟨V_l,U_l⟩∈P†_GH} S(V_l, U_l)
5      if UB_dsb − S(P_GH) > |incumbent| then
6          return UB_dsb
7      for ⟨V_l, U_l⟩ ∈ P†_GH do
8          δ ← compute using Theorem 4
9          S(V_l, U_l) ← (1 − α)S(V_l, U_l) + αδ
10         UB_dsb ← UB_dsb − δ
11     return UB_dsb
12
13 function Search(M, P_GH):
14     if |M| > |incumbent| then
15         incumbent ← M
16     bound ← DSB(P_GH, M)
17     if bound ⩽ |incumbent| then
18         return
19     ⟨V_l, U_l⟩ ← max(|V_l|, |U_l|) ↓
20     v ← d_G(v) ↓
21     for w ∈ U_l do
22         P′_GH ← compute new partitions
23         Search(M ∪ {(v, w)}, P′_GH)
24     V_l ← V_l\{v}
25     if |V_l| = 0 then
26         P_GH ← P_GH\{⟨V_l, U_l⟩}
27     Search(M, P_GH)
28
29 function McSplit+DSB(G, H):
30     global incumbent ← ∅
31     Search(∅, {⟨V(G), V(H)⟩})
32     return incumbent
```

4.3.1 Algorithm Description

Algorithm 1 details the proposed algorithm, McSplit+DSB. The function `CalcUB` calculates the upper bound UB_{dsb} by first summing $\min(|V_l|, |U_l|)$ for each bidomain $\langle V_l, U_l \rangle \in P_{GH}$ along with the size of the current mapping $|M|$ (Line 2). Active bidomains are then selected from the partitions P_{GH} and included in $P^{\dagger}_{GH}$ using the ϕ-activation function (Line 3). Then, the score $S(P_{GH})$ is calculated by summing $S(V_l, U_l)$ from these active bidomains (Line 4). If $UB_{dsb} - S(P_{GH})$ is insufficient to prune the branch compared to $|incumbent|$, UB_{dsb} is returned (Lines 5-6). Otherwise, for each active bidomain $\langle V_l, U_l \rangle \in P^{\dagger}_{GH}$, the function calculates δ using Theorem 4, updates the score $S(V_l, U_l)$ with δ, and subsequently updates UB_{dsb} based on δ. The `Search` function is similar to the `Search` function in McSplit [16], except the bound calculation is replaced by the `CalcUB`.



(a) Partitions P_{GH} after matching $(0, a)$ and $(5, \perp)$.

(b) Subgraphs formed by pairs of matching partitions $V_l = \{1, 2, 3, 4, 6\}$ and $U_l = \{b, e, f, g, i\}$.

Figure 1 Figure (a) shows the partitions $P_{GH} = \{\langle V_1, U_1\rangle, \langle V_2, U_2\rangle\}$ after matching $(0, a)$ and $(5, \perp)$, where $V_1 = \{1, 2, 3, 4, 6\}$, $U_1 = \{b, e, f, g, i\}$, $V_2 = \{7, 8\}$, and $U_2 = \{c, d, h\}$. Figure (b) shows the subgraphs $G[V_1]$ and $H[U_1]$, respectively. Since $Deg(G[V_1]) = (2, 2, 2, 3, 3)$ and $d(H[U_1]) = 4$, by Case (1) of Theorem 4, we have the dividing number $k = 4$ for $G[V_1]$ and $H[U_1]$, which yields $|M^*(G[V_1], H[U_1])| \leqslant 3$ and $\delta_{G[V_1]H[U_1]} = 2$. For $G[V_2]$ and $H[U_2]$, we have $\delta_{G[V_2]H[U_2]} = 0$. Hence, although the bound of McSplit is $UB = 8$ by Equation (1), the proposed bound is $UB_{dsb} = 6$ by Equation (2). Assume that $|incumbent| = 7$ during the search, we can prune this branch using UB_{dsb}, while McSplit cannot.

▶ **Example 8.** Figure 1(a) shows the partitioning created by the McSplit algorithm after matching $(0, a)$ and $(5, \perp)$. Matching $(5, \perp)$ means the vertex 5 is removed from further matching. This results in $P_{GH} = \{\langle 1, 2, 3, 4, 6, b, e, f, g, h, i\rangle, \langle 7, 8, c, d, h\rangle\}$. Let $V_1 = \{1, 2, 3, 4, 6\}$ and $U_1 = \{b, e, f, g, h, i\}$, $V_2 = \{7, 8\}$, and $U_2 = \{c, d, h\}$. The degree sequence of $G[V_1]$ is $Deg(G[V_1]) = (2, 2, 2, 3, 3)$, and $d(H[U_1]) = 4$. To satisfy the inequality $\sum_{i=1}^{k} d_G(v_i) - \sum_{j=k+1}^{n} d_G(v_j) > d(H)$ in Theorem 4, we need to select the first 4 elements from $Deg(G[V_1])$, making dividing number $k = 4$ the minimum value to satisfy the inequality. Then, $|M^*(G[V_1], H[U_1])| \leqslant 3$ and $\delta_{G[V_1]H[U_1]} = 2$. For $G[V_2]$ and $H[U_2]$, we have $\delta_{G[V_2]H[U_2]} = 0$. Therefore, according to Corollary 5, $UB_{dsb} = 6$, while the McSplit previous bound gives us $UB = 8$. At this point in the search, we have found the incumbent $M = \{(0, a), (5, c), (4, g), (3, e), (7, h), (6, i), (8, d)\}$ of size 7. Thus, this example demonstrates that the McSplit bound cannot prune this branch, whereas the proposed bound can successfully prune the branch.

5 Experiments

We use a 13th Gen Intel(R) Core(TM) i9-13900K server with 32 CPUs and 128GB of main memory. The implementation of our algorithm and other baselines are in C++ and compile all of them using g++ version 11.4.0 with C++17.

5.1 Benchmarks

We evaluate our approach on multiple standard benchmark datasets for maximum common subgraph detection. The Images-PR15 dataset [22] features a single large target graph (4,838 vertices) and 24 pattern graphs (4-170 vertices), creating 24 test pairs generated from segmented images. The more comprehensive Images CVIU11 dataset [5] offers 6,278 graph

pairs, combining 43 pattern graphs (22-151 vertices) with 146 target graphs (1,072-5,972 vertices). Its counterpart, Meshes CVIU11 [5], provides 3,018 pairs using 6 pattern graphs (40-199 vertices) and 503 target graphs (208-5,873 vertices). For larger-scale testing, the LV dataset [16] contains 2,352 pairs (49 patterns, 48 targets, 10-6,671 vertices), while the LargerLV extension [16] increases this to 3,430 pairs (49 patterns of 10-128 vertices, 70 targets of 138-6,671 vertices). We also include the Scalefree dataset [26, 21] (100 pairs with targets of 200-1,000 vertices and patterns at 90% target size) and the SI dataset [26, 21] (1,170 pairs with targets of 200-1,296 vertices and patterns at 20-60% target size). Finally, the Phase dataset [15] contributes 200 Erdos-Rényi graph pairs (pattern size 30, target size 150), providing a controlled random graph benchmark.

5.2 Solvers

We begin by evaluating our method against the state-of-the-art McSplit algorithm and its variants. For each of these baseline approaches McSplit [16], McSplit+RL [12], McSplit+LL [27], and McSplit+DAL [13], we develop enhanced versions (McSplit+DSB, McSplit+RL+DSB, McSplit+LL+DSB, and McSplit+DAL+DSB) by integrating our proposed bound calculation method. Since McSplit and its variations have shown a considerable advantage over other methods for solving the MCIS problem, such as maximum clique algorithms and constraint programming, we focused our experiments on McSplit and its extensions.

Our experimental evaluation uses all benchmark graph pairs described previously, with a strict 1800-second (30-minute) time limit per instance. We classify problem instances into three categories based on solution time: *easy* (solved within 10 seconds), *moderate* (solved between 10 seconds and 1800 seconds), and *hard* (unsolved within the time limit). This categorization enables detailed analysis of performance across different difficulty levels.

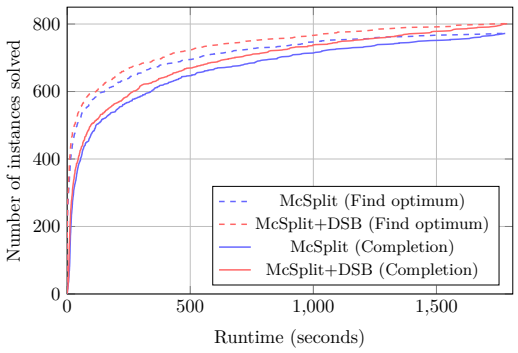


Figure 2 Cactus plot of McSplit and McSplit+DSB on time taken to complete all the enumerations and find the optimum.

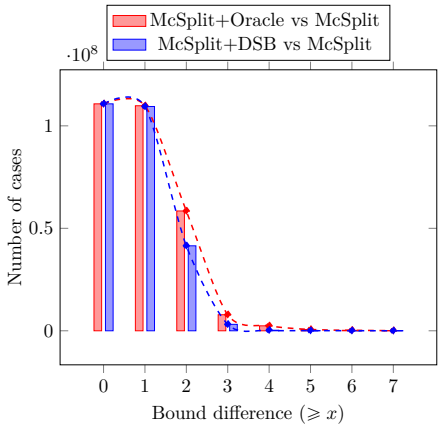


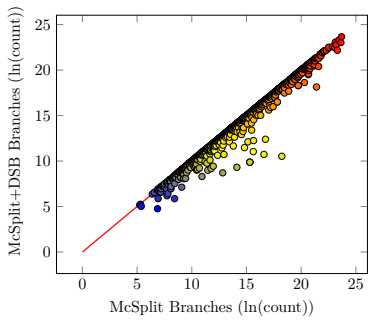
Figure 3 Comparison of the number of cases with bound differences $\geqslant x$ for two distributions: McSplit+Oracle vs. McSplit and McSplit+DSB vs. McSplit.

5.3 Comparison with McSplit

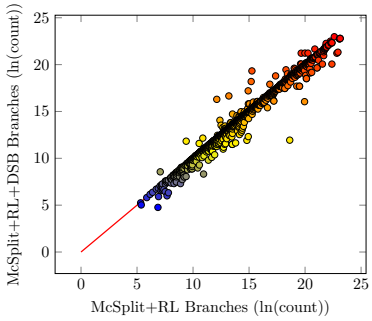
A comprehensive comparative analysis reveals that McSplit+DSB outperforms McSplit by successfully solving a greater number of moderate instances, achieving a notable 3.41% improvement. This enhancement is visually demonstrated in Figure 2, which compares

the performance of McSplit and McSplit+DSB over a 1800-second time frame. The y-axis represents the number of solved instances, while the x-axis denotes the time in seconds. The graph employs dotted lines to indicate the time taken to find the maximum common subgraph (i.e., the optimal solution) and solid lines to represent the time required to exhaustively explore the entire search space. The results clearly illustrate that the integration of the proposed bound significantly boosts the number of solved instances within the McSplit framework. Furthermore, the inclusion of this bound not only accelerates the discovery of the optimal solution but also reduces the overall time needed to verify and complete the search process. The 3.41% improvement refers to instances solved within a fixed time limit, rather than overall running time. Although it may seem modest, this consistent improvement across diverse instances highlights the generality of our approach.

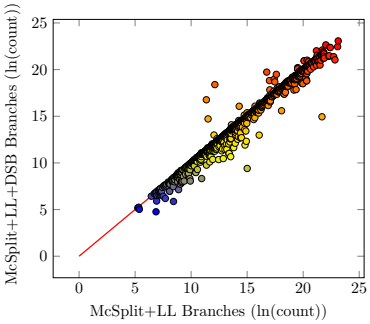
Additionally, Figure 4a provides a comparative analysis of the number of cases solved by McSplit and McSplit+DSB in logarithmic scale. The Y-axis shows the number of branches taken by the McSplit+DSB, while the X-axis shows number of branches taken by McSplit. Each point (y_1, x_1) represents a single instance, where y_1 is the branch count with DSB and x_1 is the count without it. A reference line $y = x$ is included in the plot for comparison. Points below this line indicate that our method reduced the number of branches needed, while points above suggest an increase. As seen in the plots, all the data points lie on or below the $y = x$ line, highlighting the overall efficiency of our method underscoring the computational advantages of the proposed bound. These findings collectively highlight the effectiveness of McSplit+DSB in enhancing both the speed and efficiency of the maximum common subgraph problem.



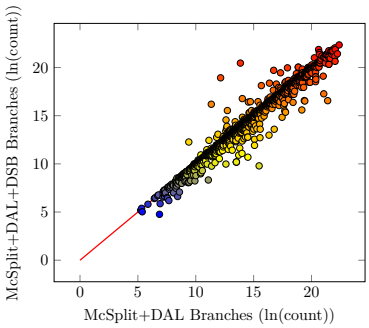
(a) McSplit vs McSplit+DSB.



(b) McSplit+RL vs McSplit+RL+DSB.



(c) McSplit+LL vs McSplit+LL+DSB.



(d) McSplit+DAL vs McSplit+DAL+DSB.

Figure 4 Comparison of the number of search branches for McSplit, McSplit+RL, McSplit+LL, and McSplit+DAL before and after applying the proposed method.

5.4 Comparison with McSplit+RL, McSplit+LL, and McSplit+DAL

Next, we assess the impact of integrating the proposed bound into the machine learning extensions of McSplit, namely McSplit+RL [12], McSplit+LL [27], and McSplit+DAL [13]. Although the bound leads to a similar number of solved instances across these methods, its true value lies in the substantial reduction of computational effort. As illustrated in Figure 4b, Figure 4c, and Figure 4d, the bound dramatically decreases the number of branches explored in moderate cases, with the majority of solved instances exhibiting a branch ratio greater than one. This indicates that the proposed bound calculation consistently explores fewer branches than its counterparts, enhancing efficiency.

However, Figure 4b, Figure 4c, and Figure 4d also reveals a few outlier points that falls above the $y = x$ line, meaning the original algorithm (without DSB) explores fewer branches. This phenomenon occurs because, in certain cases, early pruning disrupts the learning process of dynamic vertex selection models like McSplit+RL. These models rely on iterative refinement of vertex selection, and pruning initial branches can hinder this refinement, altering the selection process in subsequent iterations. In contrast, McSplit, which employs a static degree-based vertex selection method (as shown in Figure 4a), remains unaffected by such disruptions. This nuanced behavior highlights the trade-offs involved in integrating the bound with machine learning extensions, emphasizing its benefits in reducing search complexity while acknowledging its occasional interference with dynamic learning mechanisms.

5.5 Comparison with Oracle

In this experiment, we evaluate how closely our proposed bound approximates the theoretical limit of bound tightness based on bidomains alone. While even tighter bounds are possible in principle by considering relationships between vertices from different bidomains, such approaches run the risk of becoming too expensive to improve overall performance.

For a given bidomain $\langle V_l, U_l \rangle \in P_{GH}$, the exact upper bound or the ultimate bound reduction is computed by determining the maximum common subgraph of the induced subgraphs $G[V_l]$ and $H[U_l]$. We refer to this ideal bound calculation as the *oracle*. Our objective is to compare the deviations of both the proposed bound and the McSplit bound from this oracle. To facilitate this comparison, we introduce McSplit+Oracle, a variant of the McSplit algorithm that replaces its standard bound calculation with the oracle's method of computing the MCS for each matching partition as follows:

$$UB_{oracle} \leftarrow |M| + \sum_{\langle V_l, U_l \rangle \in P_{GH}} |McSplit(G[V_l], H[U_l])|, \tag{7}$$

where $McSplit(G[V_l], H[U_l])$ returns the maximum common subgraph of the induced subgraphs $G[V_l]$ and $H[U_l]$.

Figure 3 visually contrasts the bound differences between McSplit+Oracle and McSplit (represented by red bars) and McSplit+DSB and McSplit (represented by blue bars). The comparison between the red and blue bars allows us to measure the correlation between McSplit+DSB and McSplit+Oracle. Notably, the red and blue plots align closely at $x = 0$ and $x = 1$, indicating strong agreement in McSplit+DSB and McSplit+Oracle. However, as x increases beyond 1, both plots exhibit a noticeable decline.

Crucially it shows that in virtually all cases where Oracle improves on the basic McSplit bound, DSB does as well, and in most of those cases by the same amount. These results highlight the effectiveness of the proposed bound in bridging the gap toward the ultimate bound reduction, offering a significant improvement over the baseline McSplit algorithm.

Table 2 The Larger Common Subgraph Rate indicates the percentage of hard instances in which a given algorithm finds a larger common subgraph than its counterpart. The Δ column represents the difference between the success rates of the algorithm with the proposed bound and the baseline version without it.

Algorithm	Larger Common Subgraph Rate (%)	Δ (%)
McSplit	0.28	3.9
McSplit+DSB	**4.18**	
McSplit+RL	2.26	0.32
McSplit+RL+DSB	**2.58**	
McSplit+LL	1.91	1.73
McSplit+LL+DSB	**3.64**	
McSplit+DAL	5.09	0.18
McSplit+DAL+DSB	**5.27**	

5.6 Further Analysis

MCIS instances that remain unsolved by each algorithm and its branch sampling extension within the 1800-second time limit are categorized as hard instances. While these instances cannot be fully resolved within the given timeframe, identifying a larger common subgraph during this period remains highly valuable for practical applications [14]. Such partial solutions provide critical insights or approximations when exact solutions are out of reach, making them indispensable in real-world scenarios.

Table 2 compares our proposed methods against existing approaches on *hard* instances using the *Larger Common Subgraph Rate*, which quantifies the percentage of cases where our method identifies a larger common subgraph than its competitors. These larger subgraphs are subsequently used as lower bounds to prune more branches, significantly reducing computational time. The results in Table 2 demonstrate that our method consistently outperforms McSplit and its variants when the proposed bound calculation is applied to these difficult cases. In all other scenarios, both methods yield subgraphs of equal size. This performance gain highlights the proposed method's enhanced pruning efficiency and its ability to navigate the search space more effectively. These advantages establish it as a robust and practical solution for tackling complex MCIS problems, particularly under stringent time constraints. By providing larger subgraphs in *hard* cases, our method underscores the importance of integrating bound calculations into MCIS search algorithms.

The frequency with which the new bound is computed depends on both the structure of the input graph pairs and the behavior of the learned model. In our experiments, we found that the new bound was activated in approximately 1.64% of the cases, with a standard deviation of 7.99%.

6 Conclusion

In this research, we demonstrate how leveraging the structural properties of graphs can tighten the upper bound of the BnB algorithm used to solve the MCIS problem. Our reinforcement learning-driven approach strategically optimizes computations by focusing on the most significant areas of the search space. Extensive experiments show that our method consistently outperforms McSplit and its variants in both execution time and the

number of branches explored. Additionally, we establish that our proposed bound calculation closely approximates the maximum possible reduction in bounds achievable during the MCIS search. Further analysis reveals that our approach excels in challenging instances, successfully identifying larger common subgraphs for MCIS compared to existing methods.

Looking ahead, this work opens several promising directions for future research in combinatorial search problems. The principles behind our bound computation could be extended to related problems, such as subgraph isomorphism, to refine bounding strategies in other graph-based search algorithms. While this study focuses on structural properties like vertex degree and edge count to improve the upper bound in MCIS, future work could explore additional graph properties to further enhance bound calculations. Another promising direction is investigating alternative learning paradigms for more efficient application of these improved bounds, potentially leading to even greater performance gains in combinatorial search algorithms.

References

1 Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph-based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery*, 29, April 2014. doi:10.1007/s10618-014-0365-y.

2 Tatsuya Akutsu and Takeyuki Tamura. On the complexity of the maximum common subgraph problem for partial k-trees of bounded degree. In *Algorithms and Computation: 23rd International Symposium, ISAAC 2012, Taipei, Taiwan, December 19-21, 2012. Proceedings 23*, pages 146–155. Springer, 2012. doi:10.1007/978-3-642-35261-4_18.

3 Yunsheng Bai, Derek Xu, Yizhou Sun, and Wei Wang. Glsearch: Maximum common subgraph detection via learning to search. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 588–598. PMLR, July 2021. URL: http://proceedings.mlr.press/v139/bai21e.html.

4 Egon Balas and Chang Sung Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15(4):1054–1068, 1986. doi:10.1137/0215075.

5 Guillaume Damiand, Christine Solnon, Colin de la Higuera, Jean-Christophe Janodet, and Émilie Samuel. Polynomial algorithms for subisomorphism of nd open combinatorial maps. *Comput. Vis. Image Underst.*, 115(7):996–1010, 2011. Special issue on Graph-Based Representations in Computer Vision. doi:10.1016/j.cviu.2010.12.013.

6 Hans-Christian Ehrlich and Matthias Rarey. Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review. *WIREs Computational Molecular Science*, 1(1):68–79, 2011.

7 Ruth Hoffmann, Ciaran McCreesh, and Craig Reilly. Between subgraph isomorphism and maximum common subgraph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. doi:10.1609/aaai.v31i1.11137.

8 Xiuzhen Huang, Jing Lai, and Steven Jennings. Maximum common subgraph: Some upper bound and lower bound results. *BMC bioinformatics*, 7 Suppl 4:S6, February 2006. doi:10.1186/1471-2105-7-S4-S6.

9 Takeshi Kawabata and Haruki Nakamura. 3d flexible alignment using 2d maximum common substructure: Dependence of prediction accuracy on target-reference chemical similarity. *Journal of Chemical Information and Modeling*, 54(7):1850–1863, 2014. PMID: 24895842. doi:10.1021/ci500006d.

10 Ina Koch. Enumerating all connected maximal common subgraphs in two graphs. *Theoretical Computer Science*, 250(1-2):1–30, 2001. doi:10.1016/S0304-3975(00)00286-3.

11 Giorgio Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9(4):341–352, 1973.

12 Yanli Liu, Chu-Min Li, Hua Jiang, and Kun He. A learning based branch and bound for maximum common subgraph related problems. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, volume 34, pages 2392–2399. AAAI Press, April 2020. doi:10.1609/aaai.v34i03.5619.

13 Yanli Liu, Jiming Zhao, Chu-Min Li, Hua Jiang, and Kun He. Hybrid learning with new value function for the maximum common induced subgraph problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4):4044–4051, June 2023. doi:10.1609/aaai.v37i4.25519.

14 Simone Marini, Michela Spagnuolo, and Bianca Falcidieno. From exact to approximate maximum common subgraph. In *Graph-Based Representations in Pattern Recognition: 5th IAPR International Workshop, GbRPR 2005, Poitiers, France, April 11-13, 2005. Proceedings 5*, pages 263–272. Springer, 2005. doi:10.1007/978-3-540-31988-7_25.

15 Ciaran McCreesh, Patrick Prosser, Christine Solnon, and James Trimble. When subgraph isomorphism is really hard, and why this matters for graph databases. *Journal of Artificial Intelligence Research*, 61:723–759, 2018. doi:10.1613/jair.5768.

16 Ciaran McCreesh, Patrick Prosser, and James Trimble. A partitioning algorithm for maximum common subgraph problems. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 712–719. ijcai.org, 2017. doi:10.24963/ijcai.2017/99.

17 James J McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software: Practice and Experience*, 12(1):23–34, 1982. doi:10.1002/spe.4380120103.

18 Samba Ndojh Ndiaye and Christine Solnon. Cp models for maximum common subgraph problems. In Jimmy Lee, editor, *Principles and Practice of Constraint Programming – CP 2011*, pages 637–644, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-23786-7_48.

19 Jochem H. Rutgers, Pascal T. Wolkotte, Philip K.F. Hölzenspies, Jan Kuper, and Gerard J.M. Smit. An approximate maximum common subgraph algorithm for large digital circuits. In *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 699–705, 2010. doi:10.1109/DSD.2010.29.

20 Kim Shearer, Horst Bunke, and Svetha Venkatesh. Video indexing and similarity retrieval by largest common subgraph detection using decision trees. *Pattern Recognition*, 34(5):1075–1091, 2001. doi:10.1016/S0031-3203(00)00048-0.

21 Christine Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artif. Intell.*, 174(12-13):850–864, 2010. doi:10.1016/j.artint.2010.05.002.

22 Christine Solnon, Guillaume Damiand, Colin de la Higuera, and Jean-Christophe Janodet. On the complexity of submap isomorphism and maximum common submap problems. *Pattern Recognit.*, 48(2):302–316, 2015. doi:10.1016/j.patcog.2014.05.019.

23 Richard S. Sutton and Andrew G. Barto. *Reinforcement learning – An introduction, 2nd Edition*. MIT Press, 2018. URL: http://www.incompleteideas.net/book/the-book-2nd.html.

24 James Trimble. *Partitioning algorithms for induced subgraph problems*. PhD thesis, University of Glasgow, UK, 2023. doi:10.5525/GLA.THESIS.83350.

25 Philippe Vismara and Benoît Valéry. Finding maximum common connected subgraphs using clique detection or constraint satisfaction algorithms. In Le Thi Hoai An, Pascal Bouvry, and Pham Dinh Tao, editors, *Modelling, Computation and Optimization in Information Systems and Management Sciences, Second International Conference, MCO 2008, Metz, France – Luxembourg, September 8-10, 2008. Proceedings*, volume 14 of *Communications in Computer and Information Science*, pages 358–368. Springer, 2008. doi:10.1007/978-3-540-87477-5_39.

26 Stéphane Zampelli, Yves Deville, and Christine Solnon. Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15:327–353, July 2010. doi:10.1007/s10601-009-9074-3.

27 Jianrong Zhou, Kun He, Jiongzhi Zheng, Chu-Min Li, and Yanli Liu. A strengthened branch and bound algorithm for the maximum common (connected) subgraph problem. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 2022. doi:10.24963/ijcai.2022/265.