An integer program and new lower bounds for computing the strong rainbow connection numbers of graphs

Logan A. Smith | David Mildebrath | Illya V. Hicks

Department of Computational and Applied Mathematics, Rice University, Houston, 77005, Texas, USA

Correspondence
Logan A. Smith. Department of Computational and Applied Mathematics, Rice University, 6100 Main Street, MS 134, Houston, TX 77005, USA. Email: logan.smith@rice.edu

Funding information
This research was supported by the National Science Foundation, Grant/Award Number: DMS-1720225. U.S. Department of Defense.

Abstract
We present an integer programming model to compute the strong rainbow connection number, $src(G)$, of any simple graph $G$. We introduce several enhancements to the proposed model, including a fast heuristic, and a variable elimination scheme. Moreover, we present a novel lower bound for $src(G)$ which may be of independent research interest. We solve the integer program both directly and using an alternative method based on iterative lower bound improvement, the latter of which we show to be highly effective in practice. To our knowledge, these are the first computational methods for the strong rainbow connection problem. We demonstrate the efficacy of our methods by computing the strong rainbow connection numbers of graphs containing up to 379 vertices.

KEYWORDS
combinatorial optimization, graph, graph coloring, heuristic, integer programming, strong rainbow connection

1 BACKGROUND AND MOTIVATION

Let $G$ be a non-empty simple connected graph with unweighted edges, and let $c : E(G) \to \{1, \ldots, k\}$ for $k \in \mathbb{N}$ be a $k$-coloring of the edges of $G$. Note that $c$ is not necessarily a proper coloring, as adjacent edges may be the same color; throughout the paper we will consider colorings in this sense unless otherwise specified. We say that edge $e$ is color $k'$ if $c(e) = k'$. A path $P$ contained in $G$ is rainbow if $c$ maps each edge in $E(P)$ to a distinct color (i.e., for $e_1, e_2 \in E(P)$, $e_1 \neq e_2$ implies $c(e_1) \neq c(e_2)$). The graph $G$ is (strongly) rainbow connected with respect to $c$ if, for every pair of vertices $u, v \in V(G)$, there exists a (shortest) $(u, v)$-path $P$ contained in $G$ which is rainbow. The (strong) rainbow connection number $rc(G)$ ($src(G)$) is the minimum number of colors $k$ for which there exists a (strong) rainbow $k$-coloring of $G$. It is known that in general, $\text{diam}(G) \leq rc(G) \leq src(G) \leq m$, where $m = |E(G)|$ and $\text{diam}(G)$ is the diameter of $G$.

The concept of (strong) rainbow connection was first introduced by Chartrand et al. [7], and was originally intended to model the flow of classified information between government agencies in the aftermath of the terrorist attacks of September 11, 2001 [8]. Such agencies must communicate in a secure manner due to the sensitivity of their information, but messages must also traverse a complex network of intermediaries with established security protocols. Furthermore, while it is important to have enough security protocols to keep information secure, it is also desirable to have few enough to maintain reasonable simplicity. To address this concern, Chartrand et al. [7] originally posed rainbow connection as the following optimization problem: “What is the minimum number of passwords or firewalls needed that allows one or more secure paths between every two agencies so that the passwords along each path are distinct?” Rainbow connection has since been applied in other areas, including the routing of information over secure computer networks (e.g., rainbow paths arise in “onion routing” [29]). In addition to these applications, rainbow connection is of theoretical interest, and has recently garnered significant attention (see Li et al. [21] for a broad survey of rainbow connection and its many variants).
In spite of this growing attention, there does not currently exist a method for computing $src(G)$ in general graphs $G$. Previous works have proposed heuristic methods to produce rainbow colorings for variants of the strong rainbow connection problem [4,10,34]. However, none of these methods apply to the strong rainbow connection problem itself. Moreover, even in the event that these methods find an optimal solution, they are unable to provide certificates of optimality. In this work, we present the first exact solution method for computing $src(G)$ for general graphs. Our method relies on integer programming, which has a long history of solving difficult combinatorial problems (see, e.g., Nemhauser and Wolsey [23]). The critical component of our method is a practical technique for computing a strong lower bound for $src(G)$. In addition to its computational value, we believe this lower bound is also of theoretical interest.

The problem of computing $src(G)$ is known to be theoretically challenging. For certain classes of graphs—including trees, cycles, wheels, complete multipartite graphs [7], fan and sun graphs [32], stellar graphs [31], and block clique graphs [16] — $src(G)$ may be computed in polynomial time. However, in general, determining whether $src(G) \leq k$ for $k \geq 3$ is $\mathcal{NP}$-hard, even when $G$ is bipartite [1] (the same is true of $rc(G)$ [5]). Moreover, simply determining whether or not a given edge coloring strongly rainbow connects a graph is known to be $\mathcal{NP}$-complete in general graphs [33].

In this paper, we make the following contributions:

- We present the first exact solution method for computing $src(G)$ for general graphs $G$.
- We derive a novel lower bound for $src(G)$ for general graphs $G$. In addition to its value for computing the strong rainbow connection number, we believe that this bound may be of independent research interest.
- We introduce a series of computational enhancements, including a fast combinatorial heuristic and a variable elimination scheme, which significantly improve the performance of our method.
- We demonstrate the efficacy of our method by applying it to a collection of graphs containing up to 379 vertices.

The remainder of this paper is organized as follows. Section 2 presents notation and preliminaries used throughout the paper. Section 3 introduces a novel lower bound for the strong rainbow connection numbers of general graphs. Section 4 proposes an integer programming model to compute the strong rainbow connection numbers of general graphs, as well as a series of computational enhancements to improve the tractability of the integer programming model. Section 5 contains the results of several computational experiments, and concluding remarks are given in Section 6.

## 2 NOTATION AND PRELIMINARIES

Let $G$ be a graph with vertex set $V(G)$ and edge set $E(G)$. An edge $e = uv \in E(G)$ is said to be incident upon the vertices $u, v$. A graph $G$ is connected if there exists a path between every pair of vertices in $G$, and simple if both no vertex is adjacent to itself and no two distinct edges are incident upon the same pair of vertices. For the remainder of the paper, we assume that all considered graphs are simple, connected, and have at least one edge. If every pair of vertices are adjacent in $G$, then $G$ is called complete. The complete graph on $n$ vertices is denoted $K_n$. Additionally, a set of vertices which induces a complete graph is called a clique. The cardinality of the largest clique contained in a graph $G$ is called the clique number of $G$, and is denoted $\omega(G)$. A function $f : V(G) \rightarrow \mathbb{N}$ is called a proper coloring of the vertices of $G$, or simply a proper coloring of $G$, if for every pair of adjacent vertices $u, v \in V(G)$, $f(u) \neq f(v)$. The smallest $k \in \mathbb{N}$ for which there exists a proper coloring $f : V(G) \rightarrow \{1, \ldots, k\}$ is called the chromatic number of $G$ and is denoted $\chi(G)$. It is well known that $\chi(G) \geq \omega(G)$ holds in general graphs.

A graph $G$ with $\chi(G) \leq 2$ is called bipartite, and a complete bipartite graph is a graph whose vertices can be partitioned into two vertex sets $V_1, V_2$ such that no edge is incident upon two vertices within the same partition and no edge can be added to $G$ that is incident upon vertices in different partitions. The graph $K_{n_1,n_2}$ denotes the complete bipartite graph for which $|V_1| = n_1$ and $|V_2| = n_2$. The distance between vertices $u, v$, denoted $d(u,v)$, is the number of edges in any shortest $(u,v)$-path in $G$. A graph is called geodetic if every pair of distinct vertices in that graph are connected by a unique shortest length path. The union of two graphs $G_1, G_2$ is the graph whose vertex set and edge set are $V(G_1) \cup V(G_2)$ and $E(G_1) \cup E(G_2)$, respectively. Unless otherwise stated, the vertex and edge sets of $G_1, G_2$ are not necessarily assumed to be disjoint. Similarly, we define the intersection of $G_1, G_2$ as the graph with the vertex set $V(G_1) \cap V(G_2)$ and the edge set $E(G_1) \cap E(G_2)$.

Additional notation and well-known results in graph theory are provided by Bondy and Murty [3].

## 3 NEW LOWER BOUNDS FOR STRONG RAINBOW CONNECTION

In this section, we introduce a novel lower bound for the strong rainbow connection number $src(G)$. This lower bound and its construction will play a critical role in the computational methods introduced later. Additionally, we believe that this bound may be of independent theoretical interest.
By definition, a strong rainbow coloring connects each pair of vertices in $G$ with a rainbow shortest path. Consequently, we begin by studying the set $P$ of shortest paths in $G$. For each pair of vertices $u \neq v \in V(G)$, let $P_{uv}$ denote the set of shortest $(u,v)$-paths in $G$. To analyze the structure of the set $P$, it is useful to consider a set of directed graphs, $D(G) = \{D_{uv} : u \neq v \in V(G)\}$ where $D_{uv}$ is obtained by taking the graph union of each directed shortest $(u,v)$-path in $G$. This set of directed graphs will be used to construct an auxiliary graph which will provide a lower bound for the strong rainbow connection numbers of general graphs.

Additionally, we introduce a notion of separation to indicate edges and vertices which are always present in certain shortest paths. Vertices $u_1, u_2 \in V(G)$ are separated by a vertex $v \in V(G) \setminus \{u_1, u_2\}$ if $v \in V(P)$ for all $P \in P_{u_1u_2}$. Vertices $u_1, u_2 \in V(G)$ are separated by an edge $e \in E(G)$ if $e \in E(P)$ for all $P \in P_{u_1u_2}$. Equivalently, an edge $v_1 v_2$ separates vertices $u_1, u_2$ in $G$ if and only if there is a directed edge $e$ between $v_1, v_2$ in $D_{u_1u_2}$ and no directed $(u_1, u_2)$-path in $D_{u_1u_2} \setminus e$. Similarly, a vertex $v$ separates vertices $u_1, u_2$ in $G$ if and only if there is no directed $(u_1, u_2)$-path in $D_{u_1u_2} \setminus v$.

**Proposition 1.** Let $G$ be a graph and $f$ be an edge coloring of $E(G)$. Let $v_1, v_2, u \in V(G)$ such that $u$ separates $v_1$ and $v_2$. If there exists a rainbow shortest path from $v_1$ to $v_2$ in $G$ with respect to $f$, then there exist rainbow shortest paths in $G$ from $v_1$ to $u$ and from $u$ to $v_2$ with respect to $f$.

**Proof.** Let $P \in P_{v_1v_2}$ be rainbow with respect to $f$. Because $u$ separates $v_1, v_2$, by definition $u \in V(P)$. Let $P'$ be the $(v_1,u)$-subpath contained in $P$, and let $P''$ be the $(u,v_2)$-subpath contained in $P$. Because $P'$, $P''$ are each subpaths of a shortest path, they must themselves be shortest paths between their respective ends. Additionally, because $P$ is rainbow with respect to $f$, $f$ maps each of the edges in $E(P')$ and $E(P'')$ to distinct colors. Thus, there exist rainbow shortest paths in $G$ from $v_1$ to $u$ and from $u$ to $v_2$.

We now define an auxiliary graph $H(G)$ which encodes information about the set of separating edges in $G$. Let $H(G) = (V_H, E_H)$, where $V_H = E(G)$ and $E_H = \{e_1 e_2 \in E(G) \mid \text{there exists a pair of vertices } v_1, v_2 \in V(G) \text{ which are separated by both } e_1 \text{ and } e_2\}$. An example of a graph $G$ and its auxiliary graph $H(G)$ is shown in Figure 1.

---

**Figure 1** Sample graph $G$ (top left) and its auxiliary graph $H = H(G)$ (right). The bold edges $ab$, $cd$, $df$ and $hi$ in $E(G)$ correspond to the highlighted clique in $H$, and thus must be different colors in any edge coloring that strongly rainbow connects $G$ (Corollary 3). Two example directed graphs from $D(G)$ for the vertex pairs $(f,i)$ and $(a,i)$ are shown at bottom left. From the directed graph $D_{ui}$ we see that the edges $ab$ and $hi$ each separate vertices $a$ and $i$, and thus $ab$ and $hi$ are adjacent in $H$. Note that the four edges $jk, kl, mn, \text{ and } no$ each fail to separate any pairs of vertices in $G$. Their corresponding vertices in $H$ are thus each isolated.
To ease notation we will often denote $H(G)$ as simply $H$. Additionally, define $\omega'(G) := \omega(H)$ and $\chi'(G) := \chi(H)$.

Theorem 2 shows how $H$ can be used to provide lower bounds on the strong rainbow connection number of the graph $G$.

**Theorem 2.** For any graph $G$, $\text{src}(G) \geq \chi'(G) \geq \omega'(G)$. Additionally, if an edge coloring $f$ strongly rainbow connects $G$, then $f$ is a proper coloring of the vertices of $H$.

**Proof.** We first verify that the second statement of the theorem holds via its contrapositive. Let $G$ be a graph with edge coloring $f$ and assume that $f$ is not a proper coloring of the vertices of the auxiliary graph $H$. Since $f$ is not a proper coloring of the vertices of $H$, there must exist $e_1, e_2 \in V(H)$ such that $e_1, e_2$ are adjacent in $H$ and $f(e_1) = f(e_2)$. Since $e_1$ and $e_2$ are adjacent in $H$, there must exist a pair of vertices $v_1, v_2 \in V(G)$ such that $e_1$ and $e_2$ each separate $v_1$ and $v_2$ in $G$—that is, each $(v_1, v_2)$-shortest-path in $G$ must include both edges $e_1$ and $e_2$. Because $f(e_1) = f(e_2)$, we conclude that no shortest $(v_1, v_2)$-path in $G$ is rainbow. Thus, the edge coloring $f$ cannot strongly rainbow connect $G$.

Having established that any strong rainbow edge coloring $f$ of a graph $G$ must also be a proper coloring of the vertices of $H$, it follows that no edge coloring $f'$ of $G$ that uses fewer than $\chi'(G)$ colors can strongly rainbow connect $G$ as then $f'$ would also be a proper coloring of the vertices of $H$ despite having fewer than $\chi'(G)$ colors. We conclude then that $\text{src}(G) \geq \chi'(G)$. Additionally as $\chi'(G) \geq \omega'(G)$, the full statement $\text{src}(G) \geq \chi'(G) \geq \omega'(G)$ holds.

**Corollary 3.** Let $G$ be a graph and $f$ be a strong rainbow edge coloring of $G$. If $U \subseteq V(H) = E(G)$ is a clique in $H$, then $f(e_1) \neq f(e_2)$ for any $e_1, e_2 \in U$.

**Proof.** By Theorem 2, if $f$ strongly connects $G$ then $f$ must also be a proper coloring of $V(H)$. Let $U \subseteq V(H) = E(G)$ be a clique in $H$. Then every pair of edges $e_1, e_2 \in U$ are adjacent in $H$, and so $f(e_1) \neq f(e_2)$.

An illustration of Corollary 3 is shown in Figure 1.

**Theorem 4.** For any geodetic graph $G$, $\text{src}(G) = \chi'(G)$. Additionally, an edge coloring $f$ strongly rainbow connects $G$ if and only if $f$ is a proper coloring of the vertices of $H$.

**Proof.** From Theorem 2, if $f$ strongly rainbow connects $G$ then $f$ must be a proper coloring of the vertices of $H$ and $\text{src}(G) \geq \chi'(G)$. To verify the other direction of the theorem, we now assume that $f$ is a proper coloring of the vertices of $H$. Consider any pair of distinct vertices $u, v$ in $V(G)$. Because $G$ is geodetic, there exists a unique $(u, v)$-shortest-path in $G$—call it $P_{uv}$. Because $P_{uv}$ is unique, it follows that each edge $e \in E(P_{uv})$ separates $u$ from $v$, and thus every pair of edges in $E(P_{uv})$ are adjacent in $H$. Since $f$ is a proper coloring of $V(H)$, $f$ must map each pair of edges in $E(P_{uv})$ to different colors, thus the unique $(u, v)$-shortest-path in $G$ must be rainbow with respect to $f$. Because $u, v$ were chosen arbitrarily in $V(G)$ it follows that every pair of vertices in $G$ are connected by a rainbow shortest path, and thus that $f$ strongly rainbow connects $G$.

Because every proper coloring of the vertices of $H$ must strongly rainbow connect $G$, there must exist a proper vertex coloring $f'$ which colors the vertices of $H$ using exactly $\chi'(H)$ colors. Since the function $f'$ is also a strong rainbow coloring of $G$ we conclude that $\text{src}(G) \leq \chi'(G)$. Combining this with the earlier inequality given in Theorem 2, we have $\text{src}(G) = \chi'(G)$.

Despite the fact that $\chi'(G)$ is a theoretically tighter bound than $\omega'(G)$, in Section 5, we compute only $\omega'(G)$, not $\chi'(G)$. The reason for this choice is twofold. First, for many of the graphs we consider—particularly those describing social and infrastructure networks—the clique bound satisfies $\omega'(G) = \text{src}(G)$. In such situations, Theorem 2 implies that $\chi'(G) = \omega'(G)$, and thus computing $\chi'(G)$ is not necessary. Second, computing $\chi'(G)$ requires computing the chromatic number of $H$, while computing $\omega'(G)$ requires computing the clique number of $H$. Although computing both of these values is $\mathcal{NP}$-hard in general, computational experiments performed by Verma et al. [35] suggest that in sparse graphs, clique numbers can often be computed much more quickly than chromatic numbers. Verma et al. also propose and consider several lower bounds [9] for the chromatic numbers of sparse graphs, and found that the clique number provides the best lower bound in 45 of the 53 instances they consider. In all, we observe that computing $\omega'(G)$ sacrifices very little in terms of bound quality and is relatively inexpensive in terms of overall runtime.

Because of this observation, we devote the remainder of this section to studying the lower bound $\omega'(G)$. Specifically, we show that the lower bounds $\text{diam}(G)$ and $\omega'(G)$ for $\text{src}(G)$ are incomparable (Proposition 5) and may be simultaneously arbitrarily far from $\text{src}(G)$ (Proposition 6). In spite of these results, we demonstrate in Section 5 that the bound provided by $\omega'(G)$ is stronger than that of $\text{diam}(G)$ in many random graphs distributed by the Erdős–Rényi random graph model [11], as well as many graphs commonly used to describe infrastructure networks or social networks.
Proposition 5. For any \( k \in \mathbb{N} \), there exist graphs \( G \) and \( G' \) such that \( \text{diam}(G) - \omega'(G) = k \) and \( \omega'(G') - \text{diam}(G') = k \).

Proof. We first show that for any \( k \in \mathbb{N} \) there exists a graph \( G \) such that \( \text{diam}(G) - \omega'(G) = k \). Fix \( k \in \mathbb{N} \), and let \( G \) be the graph obtained by taking the union of \( k + 1 \) distinct \( K_4 \) graphs, where the vertices the \( i^{th} \) \( K_4 \) are labeled \( v_{2i-1} \) for \( j = 1, 2, 3, 4 \). The first few graphs in this sequence are illustrated in Figure 2. It can be observed that \( \text{diam}(G) = k + 1 \).

Additionally, each pair of non-adjacent vertices in \( V(G) \) are separated by at least two edge disjoint shortest paths, thus no pair of non-adjacent vertices in \( V(G) \) are separated by an edge. Since a pair of vertices in \( V(G) \) can only be separated by an edge which is incident upon both of them, no pair of edges can separate the same pair of vertices. \( E(H(G)) = \emptyset \), so \( \omega'(G) = 1 \) and \( \text{diam}(G) - \omega'(G) = k \) as desired.

To see that for any \( k \in \mathbb{N} \) there also exists a graph \( G' \) such that \( \omega'(G') - \text{diam}(G') = k \), fix some \( k \) and let \( G' = K_{1,k+2} \). Additionally, let \( H' = H(G') \) and note that \( |E(G')| = k + 1 \). Since each pair of edges in \( E(G') \) form the unique shortest path between the leaves those edges are incident upon, \( H' = K_{k+2} \). Thus \( \omega'(G') = k + 2 \). Since \( \text{diam}(G') = 2 \), we have that \( \omega'(G') - \text{diam}(G') = k \).

In the following proposition we make use of a formula shown by Chartrand et al. [7]: for integers \( s, t \) such that \( 1 \leq s \leq t \), \( \text{src}(K_{s,t}) = \lceil \sqrt{t} \rceil \).

Proposition 6. For any \( k \in \mathbb{N} \), there exist a graph \( G \) such that \( \text{src}(G) - \max(\text{diam}(G), \omega'(G)) \geq k \).

Proof. Fix \( k \in \mathbb{N} \), and let \( G = K_{2(k+2)} \). Then \( \text{diam}(G) = 2 \), and since there are at least two edge disjoint shortest paths between each pair of vertices in \( V(G) \), no pair of vertices in \( H(G) \) are adjacent and so \( \omega'(G) = 1 \). Thus \( \text{src}(G) - \max(\text{diam}(G), \omega'(G)) = \lceil \sqrt{(k + 2)^2} \rceil - 2 = k \).

4 | MODEL AND ENHANCEMENTS

In this section, we introduce an integer program for computing \( \text{src}(G) \) for general graphs \( G \). We then describe a series of enhancements which leverage the results of Section 3 in order to improve the efficiency of the model in practice.

The remainder of the section is organized as follows. We present the integer programming model in Section 4.1. Section 4.2 describes a practical method for constructing the auxiliary graph \( H \) introduced in Section 3. Section 4.3 explains a method which uses the auxiliary graph \( H \) to eliminate both variables and constraints from the integer programming model. In Section 4.4, we present the first heuristic for the (strong) rainbow connection problem, and explain how to use the heuristic to further reduce the size of the integer programming formulation. Section 4.5 describes a novel iterative scheme for computing \( \text{src}(G) \) by combining our integer programming formulation with the lower bound of Section 3. Finally, in Section 4.6 we propose several extensions of our model to different variants of the strong rainbow connection problem.

4.1 | An integer programming model

Let \( K = \{1, \ldots, K_0\} \) be a set of colors for some \( K_0 \leq m \). We introduce the following binary variables:

- \( x_{ek} = 1 \) if and only if edge \( e \in E(G) \) is color \( k \in K \).
- \( y_P = 1 \) if and only if path \( P \in P \) is rainbow (i.e., all edges \( e \in E(P) \) are different colors).
- \( \varepsilon_k = 1 \) if and only if there exists an edge \( e \in E(G) \) of color \( k \in K \).

The following integer program computes the strong rainbow connection number of \( G \), and a corresponding strong rainbow coloring.

\[
\varepsilon^*(G) = \min \sum_{k \in K} \varepsilon_k \tag{IP-1a}
\]
The objective (IP-1a) minimizes the total number of colors used in the strong rainbow coloring. The constraints (IP-1b) through (IP-1f) are summarized as follows:

- The constraints (IP-1b) require that each edge \( e \in E(G) \) is assigned exactly one color.
- The constraints (IP-1c) enforce the logical requirement that if \( y_P = 1 \) for some path \( P \in P \), then at most one edge in path \( P \) is assigned color \( k \in K \).
- The constraints (IP-1d) require that, for each pair of distinct vertices \( u, v \in V(G) \), there exists a \((u, v)\)-shortest-path \( P \) such that \( y_P = 1 \).
- The constraints (IP-1e) ensure that if there exists an edge \( e \in E(G) \) of color \( k \in K \), then \( z_k = 1 \) (i.e., color \( k \in K \) is counted in the objective value (IP-1a)).
- Constraints (IP-1f) are symmetry breaking constraints which require the number of colors used in the strong rainbow coloring to be consecutive starting at \( k = 1 \). While these constraints are not necessary to ensure a correct model, we find that they have significant computational benefits.

The following proposition certifies the correctness of the model (IP-1).

**Proposition 7.** For any graph \( G \), \( z^*(G) = src(G) \). Moreover, if \( (x, y, z) \) is a feasible solution to (IP-1), then the coloring \( f : E \to K \) with \( f(e) = \sum_{k \in K} k x_{ek} \) (i.e., which maps an edge \( e \) to the unique \( k \) with \( x_{ek} = 1 \)) strongly rainbow connects \( G \).

We note two relevant features of the integer program (IP-1). First, (IP-1) utilizes a total of \(|P| + (m + 1)|K|\) binary variables and \((|P| + m + 1)|K| + \binom{n}{2} + m - 1\) constraints. Unfortunately, \( O(2^n) \) is a tight upper bound for \(|P|\) in general graphs. Thus at worst, the integer program (IP-1) may have an exponential number of rows and columns. In spite of the potentially large size of (IP-1), we demonstrate in Section 5 that for graphs appearing in several applications, the cardinality of the set \( P \) is sufficiently small as to render (IP-1) computationally tractable. Second, (IP-1) is feasible if and only if \( K_0 \geq src(G) \). Consequently, building a feasible model requires a valid upper bound on the strong rainbow connection number \( src(G) \). While the naïve choice of \( K_0 = m \) is valid for any graph \( G \), we introduce a heuristic in Section 4.4 which computes a much tighter upper bound, thus reducing the size of integer program (IP-1) and improving its computational performance.

In addition, by replacing \( P \) with the set of all paths in \( G \) (rather than the set of shortest paths), (IP-1) can be used to compute the rainbow connection number \( rc(G) \). In general, the total number of paths in a graph \( G \) is much larger than the number of shortest paths in \( G \). Moreover, as we discuss throughout the paper, the set \( P \) of shortest paths contains structure which can be exploited for computation. Consequently, we do not consider the computation of \( rc(G) \) any further.

### 4.2 Constructing the auxiliary graph

In this section, we explain how to construct the auxiliary graph \( H \) introduced in Section 3. The techniques described in this section will be used throughout the remainder of the paper to improve the computational performance of the model (IP-1). The construction of \( H \) proceeds in three steps. First, for each \( u \in V(G) \) we construct a directed graph \( D_u \), which is related to the directed graph \( D_{mv} \) introduced in Section 3. Second, we use the set directed graphs \( \{D_u : u \in V(G)\} \) to compute the set of separating edges in \( G \). Third, we use the set of separating edges to construct the auxiliary graph \( H \). We will show how this construction also allows us to compute the set of separating vertices in \( G \), as well as \( r_{mv} = |P_{mv}| \), the number distinct shortest \((u, v)\)-paths in \( G \). These objects will be used in the computational enhancements introduced in Section 4.3 and Section 4.4, respectively.

Although they play a helpful role in the analysis of Section 3, it is not necessary to explicitly compute all \( n(n-1) \) directed graphs \( D_{mv} \). Instead, we compute a single auxiliary directed graph \( D_u \) for each \( u \in V(G) \), defined as follows. Let \( D_u = (V(G), E_D) \)
Algorithm 1. Algorithm to compute the separating edges and vertices for a fixed pair of distinct, non-adjacent vertices \( u \) and \( v \). If lines 7 and 8 are included, the algorithm also counts \( r_{uv} \), the number of distinct shortest \((u, v)\)-paths in \( G \).

Upon termination, for each \( j \in V(G) \), the value \( r_j \) equals the number of shortest \((j, v)\)-paths in \( G \). For each \( j \in V(G) \), the set \( N^+(j) \) (resp. \( N^-(j) \)) is the set of in-neighbors (resp. out-neighbors) of \( j \) in the directed graph \( D_u \).

**Data:** Distinct, non-adjacent vertices \( u, v \in V(G) \), and the directed graph \( D_u \).

**Result:** \( E_{sep} \) (the set of edges separating \( u \) and \( v \)); \( V_{sep} \) (the set of vertices separating \( u \) and \( v \)); \( r_{uv} \) (the number of distinct shortest \((u, v)\)-paths in \( G \)).

```plaintext
1 \( L \leftarrow \{v\} \)
2 \( r_v \leftarrow 1 \)
3 \( V_{sep} \leftarrow \emptyset \)
4 \( E_{sep} \leftarrow \emptyset \)
5 while \( u \notin L \)
6 \[ N \leftarrow \bigcup_{j \in L} N^+(j) \]
7 for each \( j \in N \)
8 \[ r_j \leftarrow \sum_{i \in L \cap N^-(j)} r_i \]
9 if \( |L| = |N| = 1 \)
10 \[ e' \leftarrow \text{the directed edge in } E(D_u) \text{ connecting the unique element of } N \text{ to the unique element of } L \]
11 \( E_{sep} \leftarrow E_{sep} \cup \{e'\} \)
12 if \( |N| = 1 \) and \( u \notin N \)
13 \( V_{sep} \leftarrow V_{sep} \cup N \)
14 \( L \leftarrow N \)
15 return \( V_{sep}, E_{sep}, r_{uv} \)
```

where the set of directed edges \( E_D = \{(v_1, v_2) : v_1v_2 \in E(G) \text{ and } d(u, v_1) + 1 = d(u, v_2)\} \). For fixed \( u \), the directed graph \( D_u \) can be constructed in \( O(n+m) \) time by modifying the breadth first search algorithm to allow a vertex to be discovered by multiple vertices of the previous depth. An example of the directed graphs \( D_u \) and their relation to the directed graphs \( D_{uv} \) is shown in Figure 3.

Given the directed graph \( D_u \) for some \( u \in V(G) \), Algorithm 1 computes the set of edges \( E_{sep} \) which separate \( u \) from any non-adjacent vertex \( v \in V(G) \setminus \{u\} \). As a byproduct, the algorithm also computes the set of separating vertices \( V_{sep} \), and the number \( r_{uv} = |P_{uv}| \) of shortest \((u, v)\)-paths in \( G \). For a fixed pair of vertices \( u \) and \( v \), the sets \( E_{sep} \) and \( V_{sep} \) can be computed in \( O(n+m) \) time using Algorithm 1 with lines 7 and 8 removed. Computing \( r_{uv} \) by adding lines 7 and 8 increases the time complexity of Algorithm 1 to \( O(n^2+nm) \).
Finally, we construct the graph $H$. For all $v_1 \neq v_2 \in V(G)$, let $E'_{v_1 v_2}$ denote the set of edges which separate $v_1, v_2$ in $G$ (computed using Algorithm 1 in $O(n + m)$ time). For each pair $e_1, e_2$ of distinct elements of $E'_{v_1 v_2}$, we add the undirected edge $e_1 e_2$ to $H$. For a fixed pair of vertices $v_1, v_2$, $|E'_{v_1 v_2}| = O(m^2)$, and thus $H$ can be constructed in $O(n^2 (n + m^2))$ time.

### 4.3 Variable and constraint elimination

The results introduced in Section 3 suggest two methods by which we can eliminate variables and constraints from the model (IP-1). First, by Corollary 3, for any single $U \subseteq V(H) = E(G)$ that is a clique in $H$, we may enforce a priori that each edge in $U$ is a different color—that is, we fix the variables $x_{ek}$ for all $e \in U$ and $k \in K$. Naturally, this technique eliminates the largest number of variables when $U$ is a maximum clique in $H$, corresponding to the bound $o'(G)$. In addition, this choice also helps to break symmetry inherent to the model.

Second, per Proposition 1, if a vertex $u$ separates a pair of vertices $v_1, v_2$ and at least one $(v_1, v_2)$-shortest-path is rainbow with respect to an edge coloring $f$, then there must exist shortest paths connecting both $v_1, u$ and $u, v_2$ which are rainbow with respect to $f$. Thus as long as it is enforced that the edge coloring indicated by the $x_{ek}$ variables strongly rainbow connects vertices $v_1, v_2$, it is not necessary to explicitly enforce that $v_1, u$ and $u, v_2$ are connected by rainbow shortest paths. Hence, we may eliminate the variables $y_{ip}$ and constraints (IP-1c) corresponding to paths $P \in P_{v_1 u} \cup P_{w_2}$, as well as the constraints (IP-1d) corresponding to the vertex pairs $v_1, u$ and $u, v_2$. In Section 5 we provide empirical evidence that this step substantially reduces the size of the formulation (IP-1) for some classes of graphs.

### 4.4 A fast random heuristic for (strong) rainbow connection

To the best knowledge of the authors, no heuristic method has been proposed for strong rainbow connection in general graphs. Polynomial heuristic methods have been proposed for several related problems, however, including the rainbow vertex connection problem [34] and the rainbow connection problem, both in general graphs [4] and in the special case of maximal outer planer graphs [10]. Because the upper bounds provided by heuristic methods can often be used to significantly assist with computational methods, it is desirable to also have a heuristic method for strong rainbow connection. Unfortunately, since the previously mentioned heuristics either solve related problems or determine upper bounds on $rc(G)$, which is a lower bound on $src(G)$, they are not immediately applicable to our study. We next propose a new fast random heuristic method for computing the strong rainbow connection numbers of general graphs.

The heuristic, presented in Algorithm 2, begins by randomly selecting a single shortest $(u, v)$-path $P_{uv} \in P_{uv}$ for all $u \neq v \in V(G)$ (in the coloring returned upon termination, each of the selected paths $P_{uv}$ will be a strong rainbow path). Throughout execution, the heuristic counts the number of colors $k$ used in the strong rainbow coloring (initially $k = 0$). The heuristic first randomly selects an uncolored edge $e \in E(G)$. It then checks whether edge $e$ can be colored using a color that has already been used previously (i.e., one of the colors \{1, ..., $k$\}), without causing any selected path to use the same color twice. If so, then $e$ is colored with such a color. Otherwise, $e$ is colored with a new color, and $k$ is increased by one. A new uncolored edge $e \in E(G)$ is selected, and the process repeats until all edges in $E(G)$ are colored. In order to produce better solutions, the entire randomized process can be repeated multiple times (controlled by the parameter maxIter).

This heuristic requires the selection of a path $P_{v_1 v_2} \in P_{v_1 v_2}$ for all $v_1 \neq v_2 \in V(G)$. We now describe a method to uniformly sample elements from $P_{v_1 v_2}$. Let $r_{v_1 v_2} = |P_{v_1 v_2}|$ equal the number of distinct shortest $(v_1, v_2)$-paths in $G$—in Section 4.2, we explained how these values can be quickly determined in $O(n^2 + nm)$ time. To sample a shortest $(v_1, v_2)$-path, we begin at vertex $v_2$ in the directed graph $D_{v_1}$, and move back to vertex $v_1$ as follows. Set $i_0 = v_2$ to be the initial vertex in the path. For each $i = 1, \ldots, d(v_1, v_2)$, the $i$th vertex $u_i$ in the path is sampled from the set $N^+(u_{i-1})$, which is chosen with probability $r_{v_1 u_i} / \sum_{u' \in N^+(u_{i-1})} r_{v_1 u'}$. An example of the labels $r_{uv}$ is shown in Figure 3.

Though not included in the provided pseudocode for readability, it is possible to improve the performance of Algorithm 2 by initially coloring an edge set $U \subseteq V(H) = E(G)$ which is a clique in $H$ with distinct colors and adjusting the initial values of fixed, free, $k$, and $f$ accordingly. Doing so creates a larger initial set of colors, providing additional flexibility in color choice and modestly improving the quality of solutions. Because the maximum cardinality clique in $H$ is already computed during the model preprocessing outlined in Section 4.3, it can easily be reused here. Alternatively if a worst-case polynomial time algorithm is desired, the heuristic could similarly be improved by initially coloring one of the fixed diam($G$) length paths chosen in line 4 instead.

It is also possible to extend the proposed heuristic to the standard rainbow connection problem. While any coloring generated by this heuristic must strongly rainbow connect $G$, and thus also be a valid rainbow coloring of $G$, additional rainbow connecting edge colorings may be generated by fixing randomly chosen $(u, v)$-paths for each $u \neq v \in V(G)$ rather than fixing randomly chosen $(u, v)$-shortest-paths.
The bottom-up approach

We refer to this method as the bottom-up approach. The method proceeds as follows: we first compute the lower bound \( \kappa \) presented in Section 3 is often very strong in practice and can be computed quite quickly despite its theoretical hardness. We now introduce a technique for computing \( \omega \). In particular, any existing max clique heuristic (e.g., [2, 27, 37]) could also be used to find a large clique in \( H \). Due to the relatively little computational time needed to solve max clique exactly for the problem instances we consider, in Section 5, we chose to compute \( \omega \) exactly.

Algorithm 2. Fast random strong rainbow coloring heuristic

Data: A simple connected graph \( G \), and an integer \( \text{maxIter} \).

Result: \( f \) (a valid strong rainbow coloring of \( G \)) and \( \text{best} \) (the number of colors used by \( f \)).

1. \( \text{best} \leftarrow m + 1 \)
2. for each \( i = 1, \ldots, \text{maxIter} \)
   3. \( k \leftarrow 0 \)
   4. Fix a random shortest \((u, v)\)-path \( P_{uv} \) for all \( u \neq v \) in \( V(G) \)
   5. \( \text{free} \leftarrow E(G) \)
   6. \( \text{fixed} \leftarrow \emptyset \)
   7. while \( \text{free} \neq \emptyset \)
      8. randomly select an edge \( e \) in \( \text{free} \)
      9. \( \text{free} \leftarrow \text{free} \setminus \{e\} \)
      10. \( K' \leftarrow \{1, \ldots, k\} \)
      11. for each \( u \neq v \in V(G) \)
         12. if \( K' = \emptyset \)
            goto line 17
         13. if \( e \in E(P_{uv}) \)
            14. for each \( e' \in E(P_{uv}) \cap \text{fixed} \)
               15. \( K' \leftarrow K' \setminus \{f(e')\} \)
         16. if \( K' \neq \emptyset \)
            17. \( f(e) \leftarrow \) randomly chosen element of \( K' \)
            18. \( k \leftarrow k + 1 \)
         19. else
            20. if \( k \geq \text{best} \)
               goto line 25
            21. \( f(e) = k \)
            22. \( \text{fixed} \leftarrow \text{fixed} \cup \{e\} \)
            23. \( \text{best} = \min(\text{best}, k) \)
      24. \( \text{return} f, \text{best} \)

4.5 The bottom-up approach

We now introduce a technique for computing \( \text{src}(G) \) which takes advantage of the fact that the lower bound \( \omega(G) \) presented in Section 3 is often very strong in practice and can be computed quite quickly despite its theoretical hardness. We refer to this method as the bottom-up approach. The method proceeds as follows: we first compute the lower bound \( \ell b = \max(\text{diam}(G), \omega'(G)) \) for \( \text{src}(G) \). Next, we solve (IP-1) with \( |K| = \ell b \) (i.e., with \( \ell b \) available colors). If the resulting model is infeasible, we increase \( |K| \) by 1, and repeat this process until we obtain a feasible instance. The first value of \( |K| \) for which (IP-1) is feasible equals \( \text{src}(G) \). This method clearly terminates in finite time.

The success of this method in practice relies on two factors. First, because (IP-1) has \( O(|P| + (m + 1)|K|) \) variables, \( O((|P| + m + 1)|K|) \) constraints, and \( |P| \) is relatively large, model size grows quickly with \( |K| \). To ensure feasibility of (IP-1), we require \( K_0 = \max(\text{diam}(G), \omega'(G)) \) is often very close to \( \text{src}(G) \), so the bottom-up approach requires very few iterations to converge. In Section 5 we demonstrate that this method is very effective in practice.

We note that the above technique does not require the exact computation of \( \omega'(G) \)—any lower bound on \( \text{src}(G) \) is sufficient. Since \( \omega'(G) \) often provides a particularly good bound on \( \text{src}(G) \) and can be quickly computed, we choose to utilize it here. Moreover, while larger cliques in \( H \) produce tighter bounds on \( \omega'(G) \) and thus on \( \text{src}(G) \), a clique of any size can in fact be used. In particular, any existing max clique heuristic (e.g., [2, 27, 37]) could also be used to find a large clique in \( H \). Due to the relatively little computational time needed to solve max clique exactly for the problem instances we consider, in Section 5, we chose to compute \( \omega'(G) \) exactly.
4.6 Model extensions for related problems

In this section, we describe how the integer program (IP-1) can be extended to solve several popular variants of the rainbow connection problem which have been introduced in the literature: the rainbow $k$-connection problem [8], rainbow vertex connection problem [18], the strong rainbow vertex connection problem [20], and the very strong rainbow connection problem [6] (see Li et al. [21] for a review of several of these variants and many of their known results).

Among these variants, perhaps most similar to the strong rainbow connection problem is the very strong rainbow connection problem. In this problem we also consider an edge coloring $f : E(G) \rightarrow \{1, \ldots, k'\}$ with $k' \in \mathbb{N}$, but for each pair of distinct vertices $u, v$, rather than requiring that at least one shortest $(u,v)$-path is rainbow with respect to $f$, we now require that every $(u,v)$-shortest-path is rainbow with respect to $f$. The smallest $k' \in \mathbb{N}$ for which there exists an $f$ that strongly rainbow connects $G$ is known as the very strong rainbow connection number of $G$. (IP-1) can be extended to this problem by simply requiring that $y_P = 1$ for all $P \in P$. Thus all but a polynomial number of variables $(x_{e,k}, z_k \forall e \in E(G), k \in K)$ can be eliminated, constraints (IP-1d) are necessarily satisfied and can therefore be eliminated, and constraints (IP-1c) reduce to set packing constraints.

The rainbow $k$-connection problem extends the standard rainbow connection problem by requiring that each pair of vertices are connected by at least $k$ internally vertex disjoint rainbow paths, with respect to an edge coloring $f : E(G) \rightarrow \{1, \ldots, k'\}$. The rainbow $k$-connectivity number of $G$ is then the smallest $k'$ for which there exists an edge coloring $f$ that rainbow $k$-connects $G$. As previously indicated, (IP-1) exactly solves the rainbow connection problem when the set $P$ is extended to the set of all paths in $G$. (IP-1) can be further extended to the rainbow $k$-connection problem by expanding each $P_u v$ to be the set of all $(u,v)$-paths in $G$, replacing the right-hand side of constraints (IP-1d) with $k$, and adding additional constraints of the form:

$$y_P + y_{P'} \leq 1, \quad \forall u \neq v \in V(G), \forall P \neq P' \in P_{uv} : V(P) \cap V(P') \setminus \{u,v\} \neq \emptyset.$$ 

Finally, the rainbow vertex connection problem and the strong rainbow vertex connection problems explore the notion of rainbow coloring in the case that vertices are colored rather than edges. In these settings we consider a vertex coloring $f : V(G) \rightarrow \{1, \ldots, k'\}$ with $k' \in \mathbb{N}$. A pair of vertices $v_1 \neq v_2 \in V(G)$ are said to be rainbow vertex connected if there exists a $(v_1, v_2)$-path $P$ such that for any $u_1 \neq u_2 \in V(P), f(v_1) \neq f(v_2)$. If every pair of vertices in $G$ is rainbow vertex connected, then $f$ is said to rainbow vertex connect $G$. If every pair of vertices in $G$ is rainbow vertex connected via a shortest path, then $f$ is also said to strongly rainbow connect $G$. (IP-1) can be modified for the strong rainbow vertex connection problem by replacing the $x_{e,k}$ variables with $x_{v,k}$ variables to indicate a vertex coloring rather than an edge coloring. The constraints (IP-1b), (IP-1c), and (IP-1e) can all be adjusted analogously to apply to vertex colorings rather than edge colorings. While this formulation solves the strong rainbow vertex connection problem, the standard rainbow vertex connection problem can also be solved by extending the set $P$ to be the set of all paths in $G$, as in the case of the standard rainbow connection problem.

5 COMPUTATIONAL RESULTS

In this section, we conduct a series of computational experiments to assess the performance of the methods we have introduced for calculating $src(G)$:

1 Naive model: In this test, we directly solve the integer program (IP-1). We do not use any of the computational enhancements introduced in Section 4.3 (no variable fixing, constraint elimination, etc.). This method uses the heuristic solution (Section 4.4) as the value for $K_0$.

2 Enhanced model: In this test, we directly solve (IP-1), this time using the computational enhancements of Section 4.3. The initial variable fixing was created with the maximum clique in $H$, computed using the Cliquer code [26]. This method also uses the heuristic solution (Section 4.4) as the value for $K_0$.

3 Bottom-up: This test computes $src(G)$ using the bottom-up approach introduced in Section 4.5. The initial value for $K_0$ is equal to max(diam($G$), $o'(G)$), where $o'(G)$ is again computed using Cliquer. This method also uses the computational enhancements of Section 4.3. Unlike the previous two methods, the bottom-up approach does not benefit from an initial feasible solution or upper bound, and thus the heuristic in Section 4.4 is not used.

To test these methods, we consider the following three categories of graphs:

1 Social and infrastructure graphs: We consider several graphs which represent real-world social and communication networks1 [12, 13, 17, 22, 24, 25, 28, 38], and a set of standard IEEE graphs used to represent infrastructure networks2.

---

1 Publicly available in the Koblenz Network Collection [19] and Network Repository [30].
2 Publicly available at the Power Systems Test Case Achieve: https://labs.ece.uw.edu/pstca/.
These graphs are similar to the communication networks which motivate the applications of the strong rainbow connection problem discussed in Section 1.

2 Sparse random graphs: We also consider two popular random graph models: the Erdős–Rényi random graph model [11] and Watt–Strogatz random graph model [36]. The Watts–Strogatz model is particularly relevant to the strong rainbow connection problem as these graphs exhibit the so called “small world” property often observed in social graphs – shortest paths are often quite short in length despite potentially large numbers of vertices.

3 Dense random bipartite graphs: As demonstrated in Table 2 in Section 5.2, it is often the case that the lower bound given by $\max(diam(G), \omega'(G))$ is tight in the first two categories of graphs we investigate. Consequently, the bottom-up approach of Section 4.5 generally outperforms the other methods we consider. In order to assess potential limitations of the bottom-up approach when this lower bound is not tight, we also examine dense random bipartite graphs. Although these graphs do not resemble the real-world graphs which motivate the strong rainbow connection problem, this choice is motivated by Proposition 6, which shows that the bound $\max(diam(G), \omega'(G))$ can be arbitrarily poor in complete bipartite graphs of the form $K_{2,n}$.

5.1 Implementation details

Each test was executed on the same hardware, a laptop with 32 GB of RAM running Ubuntu version 19.10 with a 2.60 GHz i7 processor. All tests were performed using a 1 hour timeout limit. We used Python 3.7.4 and the Networkx package [15] to implement the heuristic method and various preprocessing graph algorithms described in Section 4. Gurobi [14], version 9.0.2, was used to solve integer programs. Our code and the test instances we consider are publicly available at https://github.com/dtmildebrath/rainbow.

To compute cliques, we considered the publicly available max (weight) clique solver Cliquer [26] as well as a max weight independent set hybrid iterated local search heuristic (ILS-VND) proposed by Nogueira et al. [27]. As identifying max weight cliques is $\mathcal{NP}$-hard in general, we expected that exactly computing maximum cliques would be less computationally effective than using a heuristic method such as ILS-VND. In our experiments however, we found that due to the sizes of the subproblems which we encounter in our tests and the low edge densities of the corresponding auxiliary graphs, very little time is needed to solve these subproblems exactly. As such, the reported results use Cliquer to compute max cliques, and note that in the vast majority of our test instances less than 1% of the total runtime was spent on this task.

Some of the social network graphs we considered as test cases included directed edges, loops, or parallel edges. Since these were often used to indicate repeated communications, such as emails sent from one party and received by another party or messages sent from a party to back to that same party, these graphs were modified so that loops were removed, parallel edges were reduced to a single edge, and directed edges were made undirected. In the case that a graph had multiple connected components, only the component with the largest number of vertices is considered.

We also consider three random graph models. The first is the Erdős–Rényi random graph model, $G_{n,p}$, where the generated graph has $n$ vertices, and each pair of vertices is adjacent with probability $p$. The test instance labeled “ER$_{n_0,p_0}$” is then the $i$th random graph generated with $n = n_0$ and $p = p_0/100$. The second model is the Watts–Strogatz random graph model with parameters $n$, $k$, $p$. In this model, $n$ indicates a number of nodes initially organized into a cycle. These nodes are then connected to their $k$ nearest neighbors and finally each edge is removed and replaced by an edge not in the graph, with probability $p$. These test instances are labeled such that “WS$_{n_0,k_0,p_0}$” is the $i$th random graph generated with $n = n_0$, $k = k_0$, and $p = p_0/100$. The third model is the bipartite analogue of the Erdős–Rényi random graph model, $G_{n_1,n_2,p}$, where the generated graph is a bipartite graph with parts containing $n_1$ and $n_2$ vertices, and each pair of vertices which are not both in the same part are adjacent with probability $p$. Test instance “BER$_{n_a,n_b,p_0}$” is the $i$th random bipartite graph generated with $n_1 = n_a$, $n_2 = n_b$, and probability $p = p_0/100$.

Aside from the default parameters provided by the Gurobi solver, the only tunable parameter which appears in any of our methods is $\text{maxIter}$, the number of iterations for which the random heuristic introduced in Section 4.4 is run. Despite its simplicity, we found that when combined with the lower bound $\omega'(G)$ the random heuristic often provides a surprisingly tight optimality gap with $\text{maxIter} = \lceil n/5 \rceil$. As expected, runtime performance benefits diminish as $\text{maxIter}$ is increased and more search time needed for small improvements in solutions. In Table 1 we provide the solutions returned by the heuristic with $\text{maxIter} = \lceil n/5 \rceil$ and its total run time. For each test instance the heuristic was run once; the resulting solution was then used for all subsequent experiments, and the time needed for the heuristic was added to the overall runtimes reported in Table 1 as appropriate.
| Instance       | n  | src(G) | Init. LB | Heur. UB | Heur. Time | Naive       | Enhanced       | Bott. up       |
|---------------|----|--------|----------|----------|-----------|-------------|----------------|----------------|
| ER_80_3_2_0   | 80 | 19     | 19       | 27       | 0.521     | [14,21]     | 8.621          | 6.053          |
| ER_80_3_2_1   | 80 | 16     | 16       | 27       | 0.525     | 1512.083    | 25.884         | 9.226          |
| ER_80_3_2_2   | 80 | 21     | 21       | 26       | 0.510     | [16,21]     | 3.306          | 1.740          |
| ER_80_3_2_3   | 80 | 24     | 24       | 30       | 0.474     | [20,24]     | 3.180          | 1.584          |
| ER_80_3_2_4   | 80 | 25     | 25       | 29       | 0.489     | [24,25]     | 2.587          | 1.519          |
| ER_80_6_0     | 80 | 8      | 8        | 18       | 1.310     | 2319.224    | 1887.042       | 210.050        |
| ER_80_6_1     | 80 | 10     | 10       | 21       | 1.013     | 1287.704    | 888.659        | 79.950         |
| ER_80_6_2     | 80 | 8      | 8        | 18       | 1.444     | 1688.089    | 636.261        | 92.896         |
| ER_80_6_3     | 80 | 9      | 9        | 19       | 1.361     | 871.583     | 141.416        | 2.230          |
| WS_100_10_1_0 | 100| 8      | 8        | 17       | 1.689     | 670.346     | 510.330        | 69.112         |
| WS_100_10_1_1 | 100| 9      | 9        | 24       | 3.004     | [11,13]     | 3295.713       | [11, –]        |
| WS_100_10_1_2 | 100| –      | 8        | 25       | 3.186     | [10,16]     | [10,16]        | [10, –]        |
| WS_100_10_1_3 | 100| –      | 9        | 24       | 2.689     | [10,14]     | [10,14]        | [10, –]        |
| WS_100_10_1_4 | 100| –      | 7        | 22       | 2.959     | [10,12]     | [10,12]        | [10, –]        |
| WS_100_10_1_5 | 100| 4      | 4        | 13       | 16.986    | 673.842     | 475.729        | 9.915          |
| WS_100_10_1_6 | 100| 5      | 5        | 14       | 18.251    | [3,5]       | [4,5]          | 13.542         |
| WS_150_30_1_0 | 150| 4      | 4        | 14       | 17.357    | [4,6]       | [4,6]          | 14.775         |
| WS_150_30_1_1 | 150| 5      | 5        | 15       | 132.940   | [3,9]       | [3,9]          | 43.778         |
| WS_150_30_1_2 | 150| 4      | 4        | 14       | 129.283   | [3,7]       | [4,15]         | 49.698         |
| WS_150_30_1_3 | 150| 4      | 4        | 15       | 120.381   | [3,9]       | [4,14]         | 35.356         |
| WS_150_30_1_4 | 150| 4      | 4        | 15       | 122.179   | [3,8]       | [4,15]         | 46.268         |
| ieee30        | 30 | 10     | 10       | 12       | 0.019     | 0.252       | 0.106          | 0.090          |
| ieee57        | 57 | 14     | 14       | 22       | 0.115     | 54.571      | 4.893          | 1.470          |
| ieee118       | 118| 25     | 25       | 42       | 2.705     | [25,32]     | 269.386        | 225.037        |
| ieee300       | 300| (107)  | 106      | 125      | 127.574   | [–,125]     | [106,108]      | (107, –)       |
| karate        | 34 | 6      | 6        | 11       | 0.069     | 3.001       | 0.798          | 0.282          |
| surfers       | 43 | 3      | 3        | 6        | 0.401     | 1.470       | 1.145          | 0.394          |
| dolphins      | 62 | 10     | 10       | 17       | 0.524     | [9,10]      | 6.444          | 1.265          |
| lesmis        | 77 | 21     | 21       | 27       | 1.741     | 138.235     | 5.694          | 3.150          |
| adjnoun       | 112| 11     | 11       | 25       | 8.723     | [11,12]     | 1000.355       | 7.827          |
| football      | 115| (5)    | 5        | 16       | 14.334    | [5,8]       | [5,8]          | [5, –]         |
| enron-emails  | 143| 10     | 10       | 27       | 26.735    | [9,18]      | [10,17]        | 16.389         |
| rado-emails   | 167| 24     | 24       | 24       | 214.839   | [4,24]      | 309.058        | 58.173         |
| jazz-musicians| 198| –      | 6        | 22       | 416.739   | [5,22]      | [6,22]         | [6, –]         |
| ca-netscience | 379| 89     | 89       | 118      | 1213.924  | [–,–]       | [–,–]          | 1475.184       |

Notes: The lower bound max(diam(G), ω̃(G)) is indicated as Init. LB and the upper bound obtained by Algorithm 2 is indicated as Heur. UB. A 1 hour time limit was used. If a method timed out before a solution was found, the best lower and upper bounds obtained within the timeout period (if any) are given in brackets: [lb, ub]. The bottom-up approach does not produce an upper bound, and thus ub = “–” for these tests. The fastest method for each instance is indicated in bold. src values reported in parentheses were obtained using extended runtimes.
| Instance     | n   | m       | src(G) | src'(G) | diam(G) | dens(H) | |P| | |P| rem. | % rem. |
|--------------|-----|---------|--------|---------|---------|---------|----|----|------------|--------|
| ER_80_3_2_0  | 80  | 124     | 19     | 19      | 9       | 3.92    | 5655 | 2384 | 42.2       |
| ER_80_3_2_1  | 80  | 121     | 16     | 16      | 10      | 3.83    | 5276 | 2175 | 41.2       |
| ER_80_3_2_2  | 80  | 124     | 21     | 21      | 10      | 3.92    | 5708 | 2308 | 40.4       |
| ER_80_3_2_3  | 80  | 115     | 24     | 24      | 9       | 3.64    | 4906 | 1910 | 38.9       |
| ER_80_3_2_4  | 80  | 115     | 25     | 25      | 11      | 3.64    | 4621 | 1901 | 41.1       |
| ER_80_6_0    | 80  | 189     | 8      | 8       | 6       | 5.98    | 6670 | 5180 | 77.7       |
| ER_80_6_1    | 80  | 163     | 10     | 10      | 7       | 5.16    | 6327 | 4132 | 65.3       |
| ER_80_6_2    | 80  | 199     | 8      | 8       | 6       | 6.30    | 6990 | 5717 | 81.8       |
| ER_80_6_3    | 80  | 191     | 9      | 9       | 5       | 6.04    | 6524 | 5175 | 79.3       |
| ER_80_6_4    | 80  | 211     | 8      | 8       | 5       | 6.68    | 5848 | 5848 | 100.0      |
| ER_100_4_0   | 100 | 185     | 13     | 13      | 7       | 3.74    | 9221 | 5725 | 62.1       |
| ER_100_4_1   | 100 | 206     | 11     | 11      | 7       | 4.16    | 9571 | 6500 | 67.9       |
| ER_100_4_2   | 100 | 208     | –      | 10      | 8       | 4.20    | 10293| 6844 | 66.5       |
| ER_100_4_3   | 100 | 200     | –      | 10      | 6       | 4.04    | 9730 | 6005 | 61.7       |
| ER_100_4_4   | 100 | 218     | –      | 10      | 6       | 4.40    | 10580| 7580 | 71.6       |
| ER_100_8_0   | 100 | 378     | 7      | 7       | 4       | 7.64    | 13795| 13102| 95.0       |
| ER_100_8_1   | 100 | 402     | 7      | 7       | 4       | 8.12    | 14879| 14270| 95.9       |
| ER_100_8_2   | 100 | 378     | 6      | 6       | 4       | 7.64    | 13503| 12711| 94.1       |
| ER_100_8_3   | 100 | 409     | 6      | 6       | 4       | 8.26    | 15511| 14896| 96.0       |
| ER_100_8_4   | 100 | 408     | 6      | 6       | 4       | 8.24    | 15119| 14531| 96.1       |
| WS_100_10_1_0| 100 | 500     | 8      | 8       | 8       | 10.10  | 51208| 47253| 92.3       |
| WS_100_10_1_1| 100 | 500     | 9      | 9       | 8       | 10.10  | 97045| 93088| 95.9       |
| WS_100_10_1_2| 100 | 500     | –      | 8       | 8       | 10.10  | 92757| 88095| 95.0       |
| WS_100_10_1_3| 100 | 500     | 9      | 9       | 8       | 10.10  | 80985| 77201| 95.3       |
| WS_100_10_1_4| 100 | 500     | –      | 7       | 7       | 10.10  | 40077| 37174| 92.8       |
| WS_150_30_1_0| 150 | 2250    | 4      | 4       | 4       | 20.20  | 45194| 44518| 98.5       |
| WS_150_30_1_1| 150 | 2250    | 4      | 4       | 4       | 20.20  | 52548| 51895| 98.8       |
| WS_150_30_1_2| 150 | 2250    | 4      | 4       | 4       | 20.20  | 59160| 58435| 98.8       |
| WS_150_30_1_3| 150 | 2250    | 4      | 4       | 4       | 20.20  | 186169|185445| 99.6      |
| WS_150_30_1_4| 150 | 2250    | 4      | 4       | 4       | 20.20  | 154201|153663| 99.7      |
| ieee30       | 30  | 41      | 10     | 10      | 6       | 9.43   | 484  | 201  | 41.5       |
| ieee57       | 57  | 78      | 14     | 14      | 12      | 4.89   | 2147 | 699  | 32.6       |
| ieee118      | 118 | 179     | 25     | 25      | 14      | 2.59   | 15600| 6237 | 40.0       |
| ieee300      | 300 | 409     | 107    | 106     | 24      | 0.91   | 125782|36954 | 29.4       |
| karate       | 34  | 78      | 6      | 6       | 5       | 13.90  | 1478 | 1315 | 89.0       |
| surfers      | 43  | 336     | 3      | 3       | 3       | 37.21  | 3470 | 3470 | 100.0      |
| dolphins     | 62  | 159     | 10     | 10      | 8       | 8.41   | 5513 | 3339 | 60.6       |
| lesmis       | 77  | 254     | 21     | 21      | 5       | 8.68   | 6581 | 5381 | 81.8       |
| adjourn      | 112 | 425     | 11     | 11      | 5       | 6.84   | 23555| 19083| 81.1       |
| football     | 115 | 613     | 5      | 5       | 4       | 9.35   | 22150| 21114| 95.3       |
| enron-emails | 143 | 623     | 10     | 10      | 8       | 6.14   | 42979| 36048| 83.9       |
| rado-emails  | 167 | 3251    | 24     | 24      | 5       | 23.45  | 137009|108349| 79.1      |
| jazz-musicians|198 | 2742    | –      | 6       | 6       | 14.06  | 186618|172595| 92.5      |
| ca-netscience| 379 | 914     | 89     | 89      | 17      | 1.28   | 189790|131535| 69.3       |

*Notes:* For each graph G (from left to right) the order, size, src(G), lower bound src'(G), diam(G), edge density of H(G) \( \text{dens}(H) := |E(H)|/\text{(density)} \), number of shortest paths in G, number of shortest paths in G remaining after variable elimination, and the percentage of shortest paths in G remaining after variable elimination are indicated.
Results and discussion

A comparison of all three computational methods for the social/infrastructure graphs and the sparse random graphs is given in Table 1, and additional details for these graphs are provided in Table 2. The same information for the dense bipartite random graphs is given in Table 3 and Table 4, respectively. For all instances, the reported runtimes for the naïve model and the enhanced model both include the time to compute the heuristic solution; the bottom-up approach does not require a heuristic solution, and thus the bottom-up times do not include the heuristic runtime.

When the naïve and enhanced models time out, we report the best lower and upper bounds obtained by Gurobi within the timeout period (1 hour) in brackets: [lb, ub]. In three instances (ieee300, enron-emails and ca-netscience), we fail to solve the root-node LP relaxation of the naïve model within the timeout period, and thus we report a best lower bound of “...”. When the bottom-up approach times out, we report the best lower bound, that is, the largest value of \( K_0 \) attained within the timeout period. The bottom-up method does not benefit from the use of a heuristic, and does not otherwise produce an upper bound for \( src(G) \). Consequently, we report the best upper bound for the bottom-up approach as “...”. For two instances (ieee300 and football), we were able to compute \( src(G) \) using an extended time limit and additional ad hoc analysis—these values are reported for completeness in Table 1, and are given in parentheses.

Social/infrastructure and sparse random graphs

We first consider the social/infrastructure graphs and sparse random graphs (Table 1). In all but one of these instances (excepting those for which all three methods timed out), the bottom-up approach is the fastest method, often by a wide margin. This is likely due to the combination of strong, and often tight, lower bounds given by \( o'(G) \) and a robust heuristic solution search provided by Gurobi. Because the initial lower bound \( \max(diam(G), o'(G)) \) is often equal to \( src(G) \), in most cases optimality can be verified as soon as a single feasible solution is found. There are a few cases however, in which the bottom-up approach proves the bound \( \max(diam(G), o'(G)) \) to be slack. Because proving infeasibility is often computationally expensive, we expected the bottom-up approach to be outperformed by the enhanced model on these instances. However, none of the proposed methods are able to solve these instances within the hour timeout period, and the solution of the enhanced model often still terminates with a fairly large optimality gap.

In 21 of the 49 instances in Table 1, both the naïve model and the enhanced model timed out. In 25 of the remaining 28 instances, the enhanced model was faster than the naïve model, often by large margins. This result indicates the effectiveness of the variable and constraint elimination strategy introduced in Section 4.3 for sparse graphs. Indeed, as shown in Table 2, the variable elimination strategy is able to reduce the number of shortest paths in the formulation by as much as 70%. In general, the more sparse the auxiliary graph \( H \) is, the larger the number of paths that our method is able to eliminate.

One surprising observation is that despite its large number of shortest paths, and thus model sizes, the \( n = 150 \) Watts–Strogatz instances can often be solved quite quickly. In comparison, the \( n = 100, p = 0.04 \) Erdős–Rényi instances are much more difficult to compute despite having approximately 20 times fewer shortest paths. This result indicates that instance difficulty may be more closely tied to graph features other than \( |P| \), and partially justifies the exponential formulation of (IP-1). We also note that the edge density of \( H \) is often quite low, and thus the max clique subproblems used to compute \( o'(G) \) are small and particularly easy to solve in practice.

Dense random bipartite graphs

Results and statistics for the dense random bipartite graphs with \( n = 25 \) are shown in Tables 3 and 4, respectively. In contrast with the much larger instances in Table 1, we see that these instances are very challenging for their size. This is particularly pronounced for the most dense instances (first 5 rows of Table 3). As suggested by Proposition 6, these graphs have relatively large gaps between \( src(G) \) and \( \max(diam(G), o'(G)) \)—in these instances the bound is off by as much as 50%, much larger than the sparse instances of Table 1.

Because of the comparatively poor quality of the bound \( \max(diam(G), o'(G)) \) for these instances, we see that the bottom-up method is often bested by the enhanced model. This degradation of performance in the bottom-up method is likely due to the larger number of iterations needed to reach a value of \( K_0 \) which produces a feasible solution. This trend fades as the graphs becomes slightly less dense; even at 80% edge density (last 5 rows of Table 3) the bottom-up method uniformly outperforms the enhanced model. We note that within the context of the applications of the strong rainbow connection problem, the dense bipartite instances are somewhat artificial. They are not representative of most social/infrastructure networks which arise in practice, or even of real-world affiliation networks, as they have a part consisting of two vertices yet are highly dense. However, they do provide valuable insights into the limits of our methods when a strong lower bound on \( src(G) \) is not available—particularly for the bottom-up approach.
### TABLE 3  Performance comparison of proposed methods

| Instance | $n$ | $src(G)$ | Init. LB | Heur. UB | Heur. Time | Naive | Enhanced | Bott. up |
|----------|-----|----------|----------|----------|------------|-------|-----------|----------|
| BER_2_25_95_0 | 27 | 6 | 3 | 9 | 0.028 | [5,6] | 37.286 | 581.085 |
| BER_2_25_95_1 | 27 | 5 | 2 | 8 | 0.029 | 77.796 | 23.364 | 40.104 |
| BER_2_25_95_2 | 27 | 8 | 6 | 10 | 0.027 | [7,8] | 2788.336 | [7,–] |
| BER_2_25_95_3 | 27 | 7 | 4 | 9 | 0.027 | [6,7] | 44.861 | [6,–] |
| BER_2_25_95_4 | 27 | 7 | 4 | 10 | 0.027 | [6,7] | 114.553 | |
| BER_2_25_90_0 | 27 | 7 | 4 | 9 | 0.025 | [6,7] | 1873.748 | 61.777 |
| BER_2_25_90_1 | 27 | 7 | 4 | 9 | 0.025 | [6,7] | 304.780 | 935.404 |
| BER_2_25_90_2 | 27 | 6 | 4 | 9 | 0.027 | 111.781 | 2781.163 | 104.679 |
| BER_2_25_90_3 | 27 | 6 | 4 | 9 | 0.026 | [5,6] | 124.957 | 196.738 |
| BER_2_25_90_4 | 27 | 7 | 4 | 9 | 0.025 | [6,7] | 208.203 | 379.667 |
| BER_2_25_85_0 | 27 | 8 | 6 | 9 | 0.027 | 787.454 | 1.334 | 0.606 |
| BER_2_25_85_1 | 27 | 9 | 8 | 10 | 0.027 | 14.375 | 0.770 | 0.824 |
| BER_2_25_85_2 | 27 | 8 | 6 | 10 | 0.027 | [7,8] | 11.767 | 6.340 |
| BER_2_25_85_3 | 27 | 8 | 6 | 10 | 0.027 | [7,8] | 145.622 | 214.679 |
| BER_2_25_85_4 | 27 | 8 | 7 | 9 | 0.027 | 23.749 | 1.022 | 0.695 |
| BER_2_25_80_0 | 27 | 9 | 8 | 12 | 0.024 | 25.743 | 0.764 | 0.646 |
| BER_2_25_80_1 | 27 | 9 | 8 | 13 | 0.025 | 422.235 | 0.817 | 0.734 |
| BER_2_25_80_2 | 27 | 9 | 9 | 11 | 0.039 | 15.630 | 0.210 | 0.160 |
| BER_2_25_80_3 | 27 | 8 | 6 | 10 | 0.026 | 2383.839 | 20.677 | 7.788 |
| BER_2_25_80_4 | 27 | 8 | 7 | 11 | 0.025 | 25.990 | 0.936 | 0.599 |

Notes: The lower bound $\max(\text{diam}(G), \omega'(G))$ is indicated as Init. LB and the upper bound obtained by Algorithm 2 is indicated as Heur. UB. A 1 hour time limit was used. If a method timed out before a solution was found, the best lower and upper bounds obtained within the timeout period (if any) are given in brackets: [lb, ub]. The bottom-up approach does not produce an upper bound, and thus $\text{ub} = "-"$ for these tests. The fastest method for each instance is indicated in bold.

### TABLE 4  Additional information for the graphs considered in Table 3

| Instance | $n$ | $m$ | $src(G)$ | $\omega'(G)$ | diam($G$) | $\text{dens}(H)$ | $|P|$ | $|P| \text{ rem.}$ | $\% \text{ rem.}$ |
|----------|-----|-----|----------|-------------|-----------|-----------------|--------|-----------------|-----------------|
| BER_2_25_95_0 | 27 | 48 | 6 | 3 | 3 | 13.68 | 622 | 599 | 96.30 |
| BER_2_25_95_1 | 27 | 50 | 5 | 2 | 2 | 14.25 | 625 | 625 | 100.00 |
| BER_2_25_95_2 | 27 | 44 | 8 | 6 | 4 | 12.54 | 766 | 633 | 82.64 |
| BER_2_25_95_3 | 27 | 46 | 7 | 4 | 4 | 13.11 | 695 | 590 | 84.89 |
| BER_2_25_95_4 | 27 | 47 | 7 | 4 | 3 | 13.39 | 619 | 597 | 96.45 |
| BER_2_25_90_0 | 27 | 47 | 7 | 4 | 3 | 13.39 | 619 | 597 | 96.45 |
| BER_2_25_90_1 | 27 | 47 | 7 | 4 | 3 | 13.39 | 661 | 573 | 86.69 |
| BER_2_25_90_2 | 27 | 48 | 6 | 4 | 2 | 13.68 | 644 | 575 | 89.29 |
| BER_2_25_90_3 | 27 | 48 | 6 | 4 | 2 | 13.68 | 644 | 575 | 89.29 |
| BER_2_25_90_4 | 27 | 47 | 7 | 4 | 3 | 13.39 | 661 | 573 | 86.69 |
| BER_2_25_85_0 | 27 | 44 | 8 | 6 | 4 | 12.54 | 694 | 561 | 80.84 |
| BER_2_25_85_1 | 27 | 42 | 9 | 8 | 4 | 11.97 | 781 | 628 | 80.41 |
| BER_2_25_85_2 | 27 | 44 | 8 | 6 | 4 | 12.54 | 748 | 615 | 82.22 |
| BER_2_25_85_3 | 27 | 44 | 8 | 6 | 4 | 12.54 | 766 | 633 | 82.46 |
| BER_2_25_85_4 | 27 | 43 | 8 | 7 | 4 | 12.25 | 767 | 623 | 81.23 |
| BER_2_25_80_0 | 27 | 42 | 9 | 8 | 4 | 11.97 | 781 | 628 | 80.41 |
| BER_2_25_80_1 | 27 | 42 | 9 | 8 | 4 | 11.97 | 781 | 628 | 80.41 |
| BER_2_25_80_2 | 27 | 41 | 9 | 9 | 4 | 11.68 | 850 | 690 | 81.18 |
| BER_2_25_80_3 | 27 | 44 | 8 | 6 | 4 | 12.54 | 748 | 615 | 82.22 |
| BER_2_25_80_4 | 27 | 44 | 8 | 7 | 4 | 12.25 | 767 | 623 | 81.23 |

Notes: For each graph $G$ (from left to right) the order, size, $src(G)$, lower bound $\omega'(G)$, diam($G$), edge density of $H(G)$ ($\text{dens}(H) := |E(H)|/\binom{|V(H)|}{2}$), number of shortest paths in $G$, number of shortest paths in $G$ remaining after variable elimination, and the percentage of shortest paths in $G$ remaining after variable elimination are indicated.
6  |  CONCLUSIONS AND FUTURE WORK

In this paper, we propose the first exact computational methods for strong rainbow connection in graphs as well as a novel lower bound on the strong rainbow connection numbers of general graphs. We demonstrate that the bound we provide, especially when combined with the graph diameter lower bound, is often equal to the strong rainbow connection number in sparse graphs. We illustrate this phenomenon and compare the effectiveness of our proposed method in computational experiments which consider several random graph models and real world social and infrastructure networks. The method we propose is an integer program, for which we provide several computational enhancements including a random heuristic and variable/constraint elimination strategies.

Several branches of future work remain. First, it would be desirable to find an integer program formulation for computing $src(G)$ (or its variants) that contains fewer variables and/or constraints than the formulation (IP-1). Additionally, as observed in Section 5, the lower bound $\max(diam(G), 1)$ often exactly equals $src(G)$, in both instances generated by popular random graph models and real world networks commonly seen in the literature. As of yet, the only graphs which we have found with large $src(G) - \max(diam(G), 1)$ are a set of complete bipartite graphs. It may be interesting to identify a set of graphs arising from other applications or random graph models in which $src(G) - \max(diam(G), 1)$ is also large. In such graphs, it is likely that the performance of the proposed bottom-up approach would be significantly degraded due to the lack of a strong initial lower bound.

ACKNOWLEDGMENTS

This work was supported by National Science Foundation grant number DMS-1720225. Additionally, D. Mildebrath was supported by the United States Department of Defense through the National Defense Science and Engineering Graduate Fellowship program. The authors would like to thank Zachary Kingston of Rice University for helpful comments and conversations.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in Network Repository at http://networkrepository.com, reference number [31].

ORCID

Logan A. Smith @ https://orcid.org/0000-0002-6919-8217
Illya V. Hicks @ https://orcid.org/0000-0002-7815-796X

REFERENCES

[1] P. Ananth and M. Nasre, New hardness results in rainbow connectivity, 2011, arXiv:1104.2074 [cs.CC].
[2] U. Benlic and J. K. Hao, Breakout local search for maximum clique problems, Comput. Oper. Res. 40 (2013), 192–206.
[3] J. A. Bondy and U. S. R. Murty, Graph theory with applications, Vol 290, Macmillan, London, 1976.
[4] J. Cervantes-Ojeda, M. Gómez-Fuentes, D. González-Moreno, and M. Olsen, Rainbow connectivity using a rank genetic algorithm: Moore cages with girth six, J. Appl. Math. 2019 (2019), 1–7.
[5] S. Chakraborty, E. Fischer, A. Matsliah, and R. Yuster, Hardness and algorithms for rainbow connectivity, J. Comb. Optim. 21 (2011), 330–347.
[6] L. S. Chandran, A. Das, D. Issac, and E. J. van Leeuwen, Algorithms and bounds for very strong rainbow coloring, Latin American Symposium on Theoretical Informatics, 2018, pp. 625–639.
[7] G. Chartrand, G. L. Johns, K. A. McKeon, and P. Zhang, Rainbow connection in graphs, Math. Bohem. 133 (2008), 85–98.
[8] G. Chartrand, G. L. Johns, K. A. McKeon, and P. Zhang, The rainbow connectivity of a graph, Networks 54 (2009), 75–81.
[9] J. C. Cullerson and F. Luo, Exploring the $k$-colorable landscape with iterated greedy, in Cliques, coloring, satisfiability: Second DIMACS implementation challenge, Vol 26, Providence, Rhode Island: American Mathematical Society; 1996, 245–284.
[10] X. C. Deng, K. N. Xiang, and B. Wu, Polynomial algorithm for sharp upper bound of rainbow connection number of maximal outerplanar graphs, Appl. Math. Lett. 25 (2012), 237–244.
[11] P. Erdős and A. Rényi, On the evolution of random graphs, Publ. Math. I Hungar. Acad. Sci. 5 (1960), 17–60.
[12] L. C. Freeman, S. C. Freeman, and A. G. Michaelson, On human social intelligence, J. Soc. Biol. Struct. 11 (1988), 415–425.
[13] P. M. Gleiser and L. Danon, Community structure in jazz, Adv. Complex Syst. 6 (2003), 565–573.
[14] Gurobi Optimization, LLC, Gurobi optimizer reference manual, 2020.
[15] A. Hagberg, P. Swart, and D. A. Chult, Exploring network structure, dynamics, and function using NetworkX, Tech. Report, Los Alamos National Lab. (LANL), Los Alamos, NM (United States), 2008.
[16] M. Keranen and J. Lauri, Computing minimum rainbow and strong rainbow colorings of block graphs, Discrete Math. Theor. Comput. Sci. 20 (2018), 1–13.
[17] D. E. Knuth, The Stanford GraphBase: A platform for combinatorial computing, ACM Press, New York, 1993.
[18] M. Krivelevich and R. Yuster, The rainbow connection of a graph is (at most) reciprocal to its minimum degree, J. Graph Theory 63 (2010), 185–191.
[19] J. Kunegis, Konect: The Koblenz network collection, Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 1343–1350.
[20] S. Li, Y. Mao, and Y. Shi, *The strong rainbow vertex-connection of graphs*, 2012, arXiv:1201.1541 [math.CO].

[21] X. Li, Y. Shi, and Y. Sun, *Rainbow connection of graphs: A survey*, Graphs Combin. 29 (2013), 1–38.

[22] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, *The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations*, Behav. Ecol. Sociobiol. 54 (2003), 396–405.

[23] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*, in *Interscience Series in Discrete Mathematics and Optimization*, New York: Wiley, 1988.

[24] M. E. J. Newman, *Clustering and preferential attachment in growing networks*, Phys. Rev. E 64 (2001), 025102.

[25] M. E. J. Newman, *Finding community structure in networks using the eigenvectors of matrices*, Phys. Rev. E 74 (2006), 036104.

[26] S. Niskanen and P. R. J. Östergård, *Cliquer user’s guide*, Espoo: Helsinki University Technology, 2003.

[27] B. Nogueira, R. G. S. Pinheiro, and A. Subramanian, *A hybrid iterated local search heuristic for the maximum weight independent set problem*, Optim. Lett. 12 (2018), 567–583.

[28] M. Nurek and R. Michalski, *Combining machine learning and social network analysis to reveal the organizational structures*, Appl. Sci. 10 (2020), 1699.

[29] M. G. Reed, P. F. Syverson, and D. Goldschlag, *Anonymous connections and onion routing*, IEEE J. Sel. Areas Commun. 16 (1998), 482–494.

[30] R. A. Rossi and N. K. Ahmed, *The network data repository with interactive graph analytics and visualization*, Austin: AAAI Press, 2015.

[31] M. A. Shulhany and A. N. M. Salman, *The (strong) rainbow connection number of stellar graphs*, AIP Conference Proceedings, Vol. 1708, 2016.

[32] S. Sy, G. H. Medika, and L. Yulianti, *The rainbow connection number of fan and sun*, Appl. Math. Sci. 7 (2013), 3155–3160.

[33] K. Uchizawa, T. Aoki, T. Ito, A. Suzuki, and X. Zhou, *On the rainbow connectivity of graphs: Complexity and FPT algorithms*, Algorithmica 67 (2013), 161–179.

[34] O. Ugurlu and F. Nuriyeva, *A new heuristic algorithm for rainbow vertex connection, theoretical and applied aspects of cybernetics*, Proceedings of the 2nd International Scientific Conference of Students and Young Scientists, 2012, pp. 101.

[35] A. Verma, A. Buchanan, and S. Butenko, *Solving the maximum clique and vertex coloring problems on very large sparse networks*, INFORMS J. Comput. 27 (2015), 164–177.

[36] D. J. Watts and S. H. Strogatz, *Collective dynamics of “small-world” networks*, Nature 393 (1998), 440–442.

[37] Q. Wu, J. K. Hao, and F. Glover, *Multi-neighborhood tabu search for the maximum weight clique problem*, Ann. Oper. Res. 196 (2012), 611–634.

[38] W. W. Zachary, *An information flow model for conflict and fission in small groups*, J. Anthropol. Res. 33 (1977), 452–473.

How to cite this article: Smith LA, Mildebrath D, Hicks IV. An integer program and new lower bounds for computing the strong rainbow connection numbers of graphs. Networks. 2022;79:3–19. https://doi.org/10.1002/net.22031