2006

Graph reconstruction numbers

Brian McMullen

Rochester Institute of Technology
Master's Project Report
Computer Science

Graph Reconstruction Numbers

Brian McMullen

June 28, 2005

Committee:
Stanisław Radziszowski, Chair
Chris Homan, Reader
Darren Narayan, Observer

Abstract

The Reconstruction Conjecture is one of the most important open problems in graph theory today. Proposed in 1942, the conjecture posits that every simple, finite, undirected graph with three or more vertices can be uniquely reconstructed up to isomorphism given the multiset of subgraphs produced by deleting each vertex of the original graph. Although proven to be true when restricted to several classes of graphs, the general problem remains unsolved today.

Related to the Reconstruction Conjecture, reconstruction numbers concern the minimum number of vertex deleted subgraphs required to uniquely identify a graph up to isomorphism. During the summer of 2004 at the Rochester Institute of Technology, Jennifer Baldwin completed an MS project regarding reconstruction numbers. In it, she calculated reconstruction numbers for all graphs G where $3 \leq |V(G)| \leq 8$.

This project expands the computation of reconstruction numbers up to all graphs with ten vertices and a specific class of graphs with eleven vertices. Whereas Jennifer's project focused on a statistical analysis of reconstruction number results, we instead focus on theorizing the causes of high reconstruction numbers. Accordingly, this project establishes the reasons behind the high existential reconstruction numbers identified for all graphs G where $3 \leq |V(G)| \leq 10$ and identifies new classes of graphs that have large reconstruction numbers. Finally, we consider 2-reconstructibility – the ability to reconstruct a graph G from the multiset of subgraphs produced by deleting each combination of two vertices from G. The 2-reconstructibility of all graphs with nine or less vertices was tested, identifying two graphs in this range with five vertices as the highest order graphs that are 2-nonreconstructible.

Contents

1 Background

1.1 Reconstruction Conjecture

In this report, all graphs are assumed to be simple, finite and undirected. To distinguish between sets and multisets, [., ..., .] denotes a multiset. When evaluating the equivalence of two multisets, repetition of elements is taken into consideration. Given graph G, $V(G)$ is the set of vertices of G and $|V(G)|$ is the order of G. Also, $E(G)$ is the set of edges of G and $|E(G)|$ is the edge total. Two graphs, F and G are *complementary iff* $|V(F)| = |V(G)| = n$, $E(F) \cup E(G) = E(K_n)$ and $E(F) \cap E(G) = \emptyset$. If v is a vertex of G, then $G - v$ is the graph obtained from G by deleting vertex v and its incident edges – a vertex-deleted subgraph of G. The *deck* of G, $D(G)$, is the multiset of vertex deleted subgraphs of G defined by $[G - v_0, ..., G - v_{n-1}]$ where $\{v_0, ..., v_{n-1}\} \in V(G)$ and $n = |V(G)|$. Each member of $D(G)$ is referred to as a *card*. An example deck of a graph is given in Figure 1.

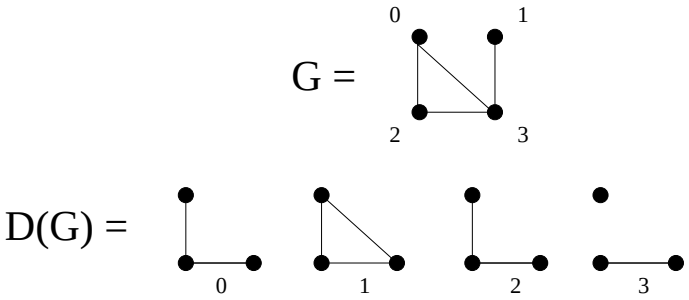


Figure 1: Deck of graph G

In 1942, Kelly and Ulam proposed the Reconstruction Conjecture [4]. It states that every graph with three or more vertices is reconstructible up to isomorphism given its deck. In other words, given graphs G and H, $D(G) = D(H)$ *iff* G and H are isomorphic. If true, then any graph with three or more vertices can be recreated simply by looking at its deck.

Although unproven in the general case, the Reconstruction Conjecture has been proven to hold true for several classes of graphs. Thanks to Kelly's Lemma, developed in 1957, the reconstructibility of regular graphs (graphs where each vertex has the same number of incident edges, or degree), disconnected graphs, and trees were established in the same year [4, 16]. It was also at this time that Kelly introduced the topic of k-reconstructibility, to be discussed in Section 1.2.

In 1964, Harary formulated the related topic of edge reconstruction. This entails reconstructing a graph from its deck of *edge*-deleted subgraphs [4]. With it comes the Edge Reconstruction Conjecture. It states that all finite simple graphs with four or more edges are edge reconstructible. Like the original Reconstruction Conjecture, it has been proven to hold true for all regular graphs, disconnected graphs and trees. Unless otherwise specified, the reader should

assume that any mention of the Reconstruction Conjecture in this report refers to reconstruction from vertex-deleted subgraphs.

It is known that the Reconstruction Conjecture does not hold for directed graphs. In 1977, Stockmeyer proved that there exists arbitrarily large directed graphs that are not reconstructible [4]. In the same year, Brendan McKay verified that the Reconstruction Conjecture holds for all simple, finite undirected graphs with nine or fewer vertices [4].

A more detailed survey of results regarding graph reconstruction can be found in [4, 5, 16]. Below is a detailed account of results regarding some classes of reconstructible graphs.

One important tool used to analyze the reconstructibility of graphs is Kelly's Lemma. Given graphs F and G, let the number of subgraphs of G isomorphic to F be denoted by $s(F, G)$.

Kelly's Lemma. *For any two graphs F and G where $|V(F)| < |V(G)|$, $s(F, G)$ can be determined given $D(G)$ [9].*

Proof.

$$s(F, G) = \frac{1}{|V(G)| - |V(F)|} \sum_{v \in V(G)} s(F, G - v) \tag{1}$$

□

The lemma results from the fact that the same occurrences of graph F in the subgraphs of G will be repeated $|V(G)| - |V(F)|$ times in $D(G)$. With this established, we can apply Kelly's Lemma to reconstruct the number of edges in G from its deck.

Corollary 1. *Given $D(G)$ of graph G, $|E(G)|$ is reconstructible [9].*

Proof. To find $|E(G)|$, we simply need to calculate $s(K_2, G)$. Obviously, the number of members in multiset $D(G)$ is equivalent to $|V(G)|$. By applying Kelly's Lemma, we can find $|E(G)|$ simply by finding the total number of edges (equivalent to K_2's) in the subgraphs of $D(G)$ and dividing this total by $|V(G)| - 2$. □

With further application of Kelly's Lemma, it is possible to reconstruct the *degree sequence* of G. The degree sequence is the number of edges incident to each vertex of G.

Corollary 2. *Given $D(G)$, the degree sequence of G is reconstructible [9].*

Proof. The number of incident edges of some vertex v_a, where $v_a \in V(G)$ and $G - v_a$ is a card of $D(G)$, is equivalent to $s(K_2, G) - s(K_2, G - v_a)$. The degree sequence of G is completed by finding the number of incident edges of the remaining elements of $V(G)$ in the same manner. □

With these theorems established, it is possible to prove the reconstructibility of all regular graphs. When proving the reconstructibility of a class of graphs, it is customary to do so by proving that the class is both *recognizable* and *weakly reconstructible* [4].

Definitions 1. A class of graphs, C, is *recognizable* if, for each graph G that is a member of C, every reconstruction of G is in C. A class of graphs, C, is *weakly reconstructible* if, for each G that is a member of C, every reconstruction of G that belongs to C is isomorphic to G [4].

Remark 1. Given the definitions, it follows that a class of graphs is reconstructible *iff* it is both recognizable and weakly reconstructible.

Lemma 1. *Regular graphs are recognizable [9].*

Proof. The degree sequence of graph G is reconstructible given $D(G)$ by Corollary 2. The number of incident edges to each element of $V(G)$ are all equal *iff* G is a regular graph. □

Lemma 2. *Regular graphs are weakly reconstructible [9].*

Proof. If we determine that graph G is a regular graph where each vertex has degree n, then by taking any card from $D(G)$, adding a vertex and connecting the new vertex to every vertex of the card with degree $n-1$ the original graph is reconstructed. □

From this, we can conclude that all regular graphs are reconstructible [4]. The proof that disconnected graphs are reconstructible follows in a similar manner.

Lemma 3. *Disconnected graphs are recognizable [9].*

Proof. Graph G, where $|V(G)| \geq 3$, is disconnected *iff* at most one card from $D(G)$ is connected. □

Lemma 4. *Disconnected graphs are weakly reconstructible [9].*

Proof. Take the largest order connected component C from the cards of $D(G)$ where G is a disconnected graph. We know that this component must be a connected component of G. A vertex, v, is known as a *cutvertex* of G *iff* there are more disconnected components in $G - v$ than G. Choose noncutvertex v_0 where $v_0 \in V(C)$. Find the set of cards in $D(G)$ with the least occurrences of C as their components. From this set, choose the card with the most occurrences of $C - v_0$ as its components. By taking this card and replacing one of the $C - v_0$ components with C, the graph G is reconstructed. □

Since this proves that disconnected graphs are weakly reconstructible and we've already seen that disconnected graphs are recognizable, we know that disconnected graphs are reconstructible [11].

1.2 k-Reconstructibility

Kelly proposed that the Reconstruction Conjecture can be generalized by considering the reconstructibility of graphs from subgraphs created by deleting some number k of vertices instead of just one [9]. The deck of graph G where each card is created by deleting a unique combinations of k vertices is denoted as $D_k(G)$. Unless otherwise specified, references to the deck of G in this report should be assumed to be regarding the singly vertex-deleted deck, or $D_1(G)$.

Definition 2. Graph G is said to be *k-reconstructible iff* it is uniquely reconstructible up to isomorphism given $D_k(G)$.

As would be expected, given set S of all graphs of order n, as k becomes larger in comparison to n it is more likely that there exist some $G \in S$ that is not k-reconstructible. In fact, for any $k > 0$, there must exist some graphs G where $|V(G)| = 2k$ that are not k-reconstructible [4]. Determining the minimum value for $f(k)$ such that all graphs G where $|V(G)| \geq f(k)$ are k-reconstructible is still an open question that is much more difficult to resolve than the Reconstruction Conjecture.

1.3 Graph Reconstruction Numbers

An issue related to the Reconstruction Conjecture is that of *reconstruction numbers*. While the Reconstruction Conjecture is concerned with the possibility of reconstructing graphs from a deck, reconstruction numbers can be considered a measure of how easily a graph can be reconstructed from a deck. In most cases, a given graph G is reconstructible from a small subset of cards from *D(G)*. In fact, Bollobás proved probabilistically that almost all graphs can be reconstructed with only three cards of a deck [3].

This project is concerned with two types of reconstruction numbers – the *existential reconstruction number* and the *universal reconstruction number*.

Definitions 3. The *existential reconstruction number* of G, $\exists rn(G)$, is the minimum number of vertex-deleted subgraphs of G required to uniquely reconstruct G up to isomorphism. The *universal reconstruction number*, $\forall rn(G)$, is the minimum number n for which all multisubsets of $D(G)$ of size n uniquely reconstruct G up to isomorphism.

Where a value for $\exists rn(G)$ is found for graph G, we know that G is reconstructible. In case there exists a graph G that is not reconstructible, we say that $\exists rn(G) = \infty$.

This report is only concerned with graphs with three or more vertices. The minimum existential reconstruction number for all of these graphs is three [2, 7].

Proposition 1. *Given graph G, where $|V(G)| \geq 3$, $\exists rn(G) > 2$.*

Proof. For graph G to have two for an existential reconstruction number, there must be two vertices that can be chosen whose deletion will create a subset of $D(G)$ unique to the automorphism class of G. However, given any two vertices of

G, v_a and v_b, we can produce a graph H that shares the subdeck $\{G-v_a, G-v_b\}$ by copying graph G and inverting the edge between v_a and v_b. □

Much has been proven regarding the existential reconstruction numbers of disconnected graphs thanks to work done by Wendy Myrvold and others [14, 15].

Theorem 1. *Given disconnected graph G where not all components are isomorphic, $\exists rn(G) = 3$ [15].*

This was done by providing algorithms for choosing three vertices for disconnected graphs where not all components have the same order and disconnected graphs with all components of the same order but not all isomorphic, and arguing that the deletion of these vertices result in subdecks unique to the original graphs. Problems with this proof were identified and corrected by Molina [14], but the results remain correct.

Theorem 2. *Given disconnected graph G composed of isomorphic components, each component of order c, $\exists rn(G) \leq c + 2$ [15].*

With the use of the above theorem, Myrvold found a method for calculating $\exists rn$ for any disconnected graph composed entirely of isomorphic, complete components.

Theorem 3. *Given disconnected graph G of the form pK_c, $\exists rn(G) = c + 2$ [15].*

Proof. Given disconnected graph G of the form pK_c, $D(G)$ consists of $|V(G)|$ elements of subgraph $(p-1)K_c \cup K_{c-1}$. Consider disconnected graph H of the form $K_{c+1} \cup (p-2)K_c \cup K_{c-1}$. By deleting each vertex of K_{c+1} from H we recreate the only multisubdeck of G of size $c+1$, meaning $\exists rn(G) > c+1$. Since applying Theorem 2 reveals that $\exists rn(G) \leq c+2$, we can conclude that $\exists rn(G) = c+2$. □

Using graph $3K_3$ as an example, the above proof is illustrated in Figure 2. Removing each numbered vertex from graph H, we recreate $c+1$ cards from $D(G)$, pushing $\exists rn(G)$ to five. As recently as 2002, Asciak and Lauri proved that this is the only class of disconnected graphs with such a high $\exists rn$ value.

Theorem 4. *Given disconnected graph G of the form pC where C is not a complete subgraph, $\exists rn(G) \leq c$ [1].*

2 General Project Description

Jennifer Baldwin first composed a project on calculating and analyzing existential and universal reconstruction numbers for graphs. The calculations were performed for all graphs G where $3 \leq |V(G)| \leq 8$ and some graphs where $|V(G)| = 9$ – a total of approximately 26,000 graphs [2]. This project expands the calculation of existential and universal reconstruction numbers for all graphs

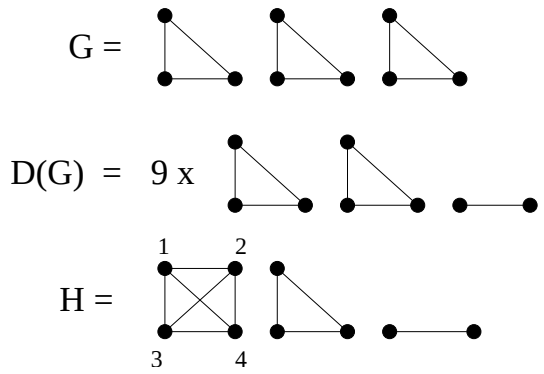


Figure 2: Disconnected graph composed of 3 K_3 components with $\exists rn(G) = 5$

up to order ten and for a class of graphs of order eleven – raising the number of graphs analyzed to approximately 12,294,000. Also, in this project I focus on the reasons certain graphs have unusually high reconstruction numbers and attempt to organize these graphs into classes to identify graphs of all orders with high reconstruction numbers.

Although no source code was used from Baldwin's original implementation, the same basic algorithm was used with many enhancements for the sake of efficiency. Both the algorithm and the enhancements are outlined here. In the proposal, it was hoped that reconstruction numbers for all graphs with up to eleven vertices would be calculated. This report argues this is not currently feasible given constraints of time and processing power.

Lastly, calculations were performed to find non-trivial examples of graphs that are not 2-reconstructible. This was done in response to an email sent by Geoff Exoo of Indiana State University to Stanisław Radziszowski of the Rochester Institute of Technology on October of 2004. It states in part:

> Specifically, I'd like to know: What are the largest known graphs that are not (set) k-reconstructible, for $k > 1$. One can search through all small graphs and find examples: there are graphs of order 11 that are not set 4-reconstructible, graphs of order 9 that are not set 3-reconstructible, etc. But non-trivial examples for $k = 2$ appear to be rare.

It was confirmed that these graphs are extremely rare, and the results suggest that all graphs with more than five vertices are 2-reconstructible. Put in more general terms, the Reconstruction Conjecture assumes that the maximum order of graph G that is not 1-reconstructible is two, while it would seem that this maximum order for 2-nonreconstructibility is five.

3 Graph Reconstruction Calculations

3.1 Algorithm Overview

The algorithm for calculating Graph Reconstruction Numbers involves comparing the decks of graph G to the decks of a select set of graphs to determine minimal reconstruction numbers for G. Here the set of graphs considered is the set of all *single vertex extensions* of the cards of $D(G)$. Pseudocode for finding $\exists rn(G)$ of input graph G is given in Algorithm 1. Details regarding extension graphs and evaluating all possible subdecks of a given size are below.

Algorithm 1 General algorithm for computing $\exists rn(G)$

```
1: Input graph G
2: Acquire D(G)
3: Acquire all 1-vertex extensions of each card of D(G)
4: Acquire the deck of each extension graph
5: for subdeck_size = 2 to n − 1 do
6:    for each subdeck of G of size subdeck_size do
7:       found = false
8:       for each extension graph do
9:          if extension deck contains current subdeck of G then
10:            found = true
11:            break out of innermost for loop
12:         end if
13:      end for
14:      if found = false then
15:         break out of outermost for loop
16:      end if
17:   end for
18: end for
19: The value of subdeck_size is ∃rn(G)
```

The n *vertex extensions* of a given graph G is the set of all graphs H where there exists a subset of n vertices from $V(H)$ such that $H - v_1 - ... - v_n \cong G$. So the order of every n vertex extension of G is $|V(G)| + n$. Given graph G, the set of all *single* vertex extensions to each card from $D(G)$ is, by definition, the set of all graphs sharing at least one card in their decks with that of $D(G)$. The maximum number of possible single vertex extensions of given graph F is $2^{|V(F)|}$, but the actual number is usually much less given isomorphism.

In Figure 3, where reconstruction numbers are to be calculated for graph G, extension graphs generated from $D(G)$ are shown. The graphs listed down the leftmost column of the figure is the set (not multiset) of cards from $D(G)$. Listed next to each of these are the corresponding extension graphs. Enclosed in squares is the set of extension graphs to be considered when calculating reconstruction numbers for G. Of course, graph G itself is not used as an extension graph.

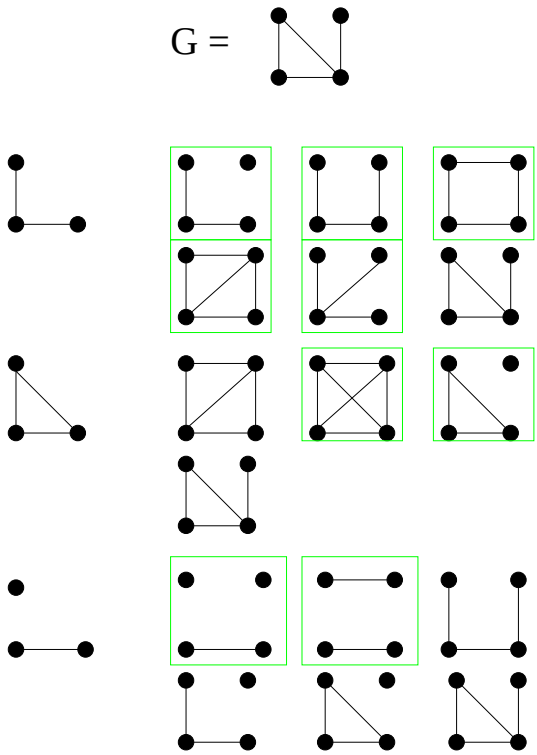


Figure 3: Unique extensions of $D(G)$

After finding the unique extension graphs for each card of $D(G)$, matches between the cards of $D(G)$ and the cards of extension graphs' decks are represented by a *relation matrix*. An example is given by Figure 4. The first row shows graph G and its deck, while following rows show extension graphs and unique matches marked between their cards and the cards of $D(G)$. It is important to put isomorphically equivalent cards from $D(G)$ adjacent to each other as shown in columns two and three of Figure 4. Also, matches to repeated cards of $D(G)$ from an extension deck must be filled in from left to right, as illustrated in row one of Figure 4. Otherwise, it would be difficult to determine the set of all possible subdecks of G of a given size.

With the relation matrix in place, the existential reconstruction number of G can be found. We begin by assigning variable *subdeck_size* a value of 2. Then every subset of $D(G)$ of size *subdeck_size* is examined to determine if there exists a multisubdeck of G of size *subdeck_size* that is not contained in any of the decks of the extension graphs. If so, then the current value of *subdeck_size* is equivalent to $\exists rn(G)$. Otherwise, *subdeck_size* is incremented and the process is repeated. As long as G is reconstructible, an existential reconstruction number will be found.

In the example diagram, we can see that no matter what two unique cards we choose from $D(G)$, by reading down the columns in question we come up with at least one instance of two matching cards from an extension graph deck. Specifically, the unique subdecks evaluated in order are from columns $\{1, 2\}$, $\{1, 4\}$, $\{2, 3\}$ and $\{2, 4\}$ – each revealing at least one extension graph with a matching subdeck. Therefore, $\exists rn(G) > 2$. Incrementing the subdeck size to three, we first evaluate the subdeck from columns $\{1, 2, 3\}$ and find a match at row four. However, we next find a unique subdeck of G when we come to columns $\{1, 2, 4\}$. $\exists rn(G) = 3$. If there had been a matching subdeck to $\{1, 2, 4\}$, we would have next examined rows $\{2, 3, 4\}$ before evaluating the full deck of G. Given these examples, the general algorithm for evaluating an exhaustive list of subdecks of a given size should be obvious.

Because the old and new algorithms for finding universal reconstruction numbers were fundamentally different, these are discussed in detail separately in Sections 3.3 and 3.5.

3.2 Software Tools

There were two tools used to assist in the calculation of reconstruction numbers. One of these, the Condor package [6], which is installed on several machines in labs at RIT, distributes the computation workload among several machines according to which machines have the most available processing time. This provided parallel processing that was essential for completing calculations in reasonable time.

The second tool used was the **nauty** software package written by Brendan McKay [13]. **Nauty** offers functionality for isomorph-free exhaustive generation of graphs, a compact and readable representation of simple graphs, canonization of graphs according to isomorphic equivalence as well as other basic graph

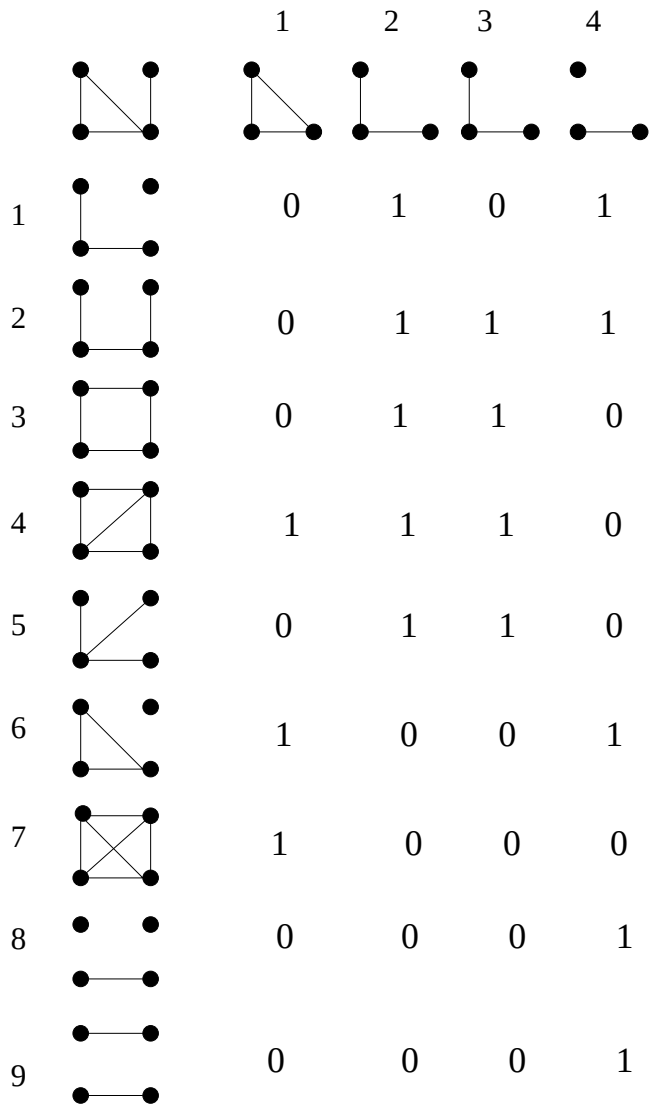


Figure 4: Relation matrix of G

manipulation functions. This is currently the most efficient software package available for identifying isomorphisms between graphs.

Graphs manipulated in the **nauty** package were commonly presented in what is known as **graph6** implementation. This is a string of ASCII characters used to denote the graph's order with the first character and a packed adjacency matrix with all characters following. As an example, the graph referred to as G in Figure 1 is CN in **graph6** notation. Each valid **graph6** representation of a graph has a single canonized labeling, also represented in **graph6**. Two graphs have the same canonical labeling *iff* they are isomorphs.

3.3 Old Implementation Details

Jennifer Baldwin's method for calculating reconstruction numbers depends heavily on files to avoid repeating calculations and save on processing time. In her implementation, the first step in finding reconstruction numbers for a set of graphs is to calculate all single vertex extended graphs and single vertex deleted graphs that will be needed. The results for these calculations are stored in two different files – one for extensions and the other for deletions.

Given an input file of graphs in **graph6** format, programs *shrink* and *expand* produce canonized graphs in **graph6** format for single vertex deletions and extensions, respectively. The output for the two programs are saved to two different files. The program *reconstruct* uses these two files to find extension files and decks needed by input graphs to calculate their reconstruction numbers. *Reconstruct* cannot find the reconstruction number of an input graph G unless $D(G)$ and the extension graphs of the cards of $D(G)$ are present in these files.

For each input graph G, *reconstruct* first scans the given *shrink* file from the beginning, looking for the entry for $D(G)$. Once found, the program takes the unique cards of $D(G)$ and searches for the set of extension graphs of each one from the beginning of the *expand* file. Once the set of appropriate extension graphs are identified, each one of their decks is found by scanning the *shrink* file from the beginning. Finally, an entry is made in the relation matrix for each extension graph so that the cards of its deck are related to $D(G)$.

With the relation matrix formed, the existential reconstruction number of G is computed as specified in Section 3.1. Looking at Algorithm 1, Jennifer's algorithm for acquiring $D(G)$, extensions of cards from $D(G)$ and their decks entails scans of the pre-generated files.

It is obvious that the universal reconstruction number cannot be smaller than the existential reconstruction number. As a result, the calculated $\exists rn(G)$ value is used as the first value to test as being $\forall rn(G)$. After $\exists rn(G)$ is assigned to variable $curr\_sz$, the program checks whether all subdecks of $D(G)$ of size $curr\_sz$ are unique to $D(G)$ by looking down the proper columns of the relation matrix. If an extension deck is found with the same subdeck being examined then $curr\_sz$ is incremented, and the process is repeated. Otherwise, $\forall rn(G) = curr\_sz$. A universal reconstruction number will be found as long as G is reconstructible. Details in pseudocode are given in Algorithm 2.

Algorithm 2 Old algorithm for computing $\forall rn(G)$

```
1: for subdeck_size = ∃rn(G) to n − 1 do
2:    found = false
3:    for each unique subdeck of G of size subdeck_size do
4:       for each extension graph from ∃rn(G) algorithm do
5:          if extension deck contains current subdeck of G then
6:             found = true
7:             break for on line 3
8:          end if
9:       end for
10:   end for
11:   if found = false then
12:      break for
13:   end if
14: end for
15: The value of subdeck_size is ∀rn(G)
```

3.4 Comments on Old Implementation

Using the old implementation, Jennifer was able to complete reconstruction number calculations for all graphs with eight or less vertices. However, when the program was applied to graphs with nine vertices, the calculations for a single graph would take anywhere from one second to a full minute to complete [2]. Given that there are 274,688 graphs with nine vertices, completing the calculations would be too time-consuming even considering the gains achieved by Condor.

In Jennifer's project, it was mentioned that file I/O was the major bottleneck. Looking at the implementation, it is easy to see why. The calculations required for the reconstruction numbers of a single graph G entails sequential scans through text files for $D(G)$, the extensions of the cards of $D(G)$ and the decks for each extension. As the order of graphs we want to analyze increases, the number of possible graphs we need to consider increases rapidly. This leads to much larger files and, consequently, much longer seek times to find the data we need. To split up the sets of graphs we want to consider to reduce the size of extension and shrink files is difficult to manage. Indeed, as the order of graphs grows the amount of time spent on file I/O is much more of a performance hit than what we are trying to save from processing time.

This and other considerations form the basis for improvements in efficiently determining reconstruction numbers of graphs. Adjustments to the old algorithm as stated below made it possible to analyze a much larger data set of graph reconstruction numbers.

3.5 Current Implementation Overview

Whereas the old implementation depended on extension graphs and vertex deleted subgraphs being calculated beforehand and placed into files, the current implementation simply calculated these graphs as they were needed. Although the calculations of extensions and decks of the same graphs are repeated in this scenario, the more severe time-loss to CPU waits on file I/O were eliminated.

Although it is conceivable that the extension graphs and deck of each graph can be stored in memory when first needed and retrieved again for the reconstruction number calculations of each subsequent graph that needs it, this is simply not feasible without very sophisticated methods. There are over 12,000,000 graphs with ten vertices and over 1,000,000,000 graphs with eleven vertices. This is especially sobering when we consider that there can be up to $2^{|V(G)|}$ possible single vertex extensions of graph G. Also, the calculations of the deck of a graph take up so little processing time that they are hardly worth the effort. Lastly, since the reconstruction number calculations will be done by many programs running on Condor, it makes sense to keep their memory footprints reasonably small since they will need to be migrated as other users log onto the machines in the lab.

Another big issue to consider in the old implementation is the algorithm for finding universal reconstruction numbers. Although it is guaranteed to obtain the correct result, the old method performs far more work than is needed. The only information needed to find $\forall rn(G)$ is the maximum number of matches between the cards of $D(G)$ and the extension graphs' decks. This information can be easily obtained as the relation matrix is set up. Adding one to this maximal value yields the universal reconstruction number since this is the smallest subdeck size, s, where all possible subdecks of $D(G)$ of size s are unique when compared to the decks of relevant extension graphs.

We can also improve the efficiency of our calculations when we observe that universal and existential reconstruction number values are always the same between two complementary graphs [7]. Knowing this we can calculate reconstruction numbers for approximately half of all graphs of a specified order, and simply assign these values to their complements. This is a little more difficult than it seems since we need to generate input graphs of a specified order n such that they and their complements form the set of all graphs of order n and no generated graph is the complement of another generated graph. It is also important to note that some graphs are *self-complementary* – that is graphs G such that $G \cong \overline{G}$. If we blindly assign the reconstruction numbers of a self-complementary graph to its complement, we would be counting the same graph twice. Information on how input graphs are generated and assignment to complementary graphs are done properly is explained in the Section 3.6.

So as each graph G is input for calculation, $D(G)$ is calculated. As we calculate each extension graph of the cards of $D(G)$, its deck is in turn calculated and matched up with $D(G)$. Since we know that for every G we consider $\exists rn(G) \geq 3$, only extension decks with three or more matching cards are placed in the relation matrix. As extension graphs are added to the relation matrix, we keep

track of the maximal number of matches between extension decks and $D(G)$. After the relation matrix is complete, incrementing this maximal value gives us $\forall rn(G)$. If no extension graph entries end up in the matrix then $\exists rn(G) = 3$.

If there are extension graph entries in the relation matrix at this point, we begin looking at subgraphs of $D(G)$ as specified earlier to find the smallest one possible that is unique among the extension decks. Whereas each row in Jennifer's implementation is an array of values, the recent implementation simply uses a simple bitmap to represent the boolean match/non-match values. This makes calculation quicker as using a bitmask to represent the subgraph of $D(G)$ we are looking at and performing a "bitwise and" with the extension deck in question quickly determines whether or not we have found a unique subdeck. Once we have found $\exists rn(G)$ and $\forall rn(G)$, these values are assigned to $\overline{G}$ if necessary. Situations in which we don't assign these values to the complement are specified in Section 3.6.

Finally it must be noted that in the proposal for this project it was stated that we would use information we already know about reconstruction numbers for certain classes of disconnected graphs to save calculations. However, this was decided against since identifying disconnected graphs, their components and isomorphic equivalence between components may entail more calculations than they save. Also, letting the algorithm work on graphs with known reconstruction numbers helps test that it is functioning correctly.

3.6 Current Implementation Details

The generation of input graphs was performed using the *geng* program from Brendan McKay's *nauty* package. It is able to efficiently generate an exhaustive list of graphs of a specified class. Two properties of graphs that *geng* can specify are the number of vertices and minimum and maximum number of edges. The maximum number of edges possible for graph G is $\frac{n(n-1)}{2}$ where $n = |V(G)|$. Knowing this, we can generate all proper input graphs of order n by telling *geng* to create all graphs of order n with a maximum of $\lfloor \frac{n(n-1)}{4} \rfloor$ edges. Then we can safely calculate reconstruction numbers for each input graph and assign them to each corresponding complement. We need to be careful when we are doing calculations for graphs with an even number of maximum edges, however, as there exists graphs that have the same number of edges as their complements. So to avoid counting reconstruction numbers for the same graph more than once, the reconstruction numbers calculated for all graphs of order n with exactly $\frac{n(n-1)}{4}$ edges are not assigned to their complements. The reconstruction numbers of these graphs and their complements must be computed separately.

There are $2^{|V(G)|}$ possible extension graphs of G when isomorphism is not taken into consideration. To avoid placing the same extension graph in the relation matrix of G more than once, a hash table was used. The hash table was cleared before a new input graph was read in. As extension graphs were generated, each one was canonized and placed both in the hash table and relation matrix when first encountered. If already seen for the input graph, it was not

Algorithm 3 Current algorithm for computing $\exists rn(G)$ and $\forall rn(G)$

```
1: Program parameter order specifies order of input graphs
2: totaledges = order(order−1)/2, maxedges = ⌊totaledges/2⌋
3: Clear hash_table
4: Clear list rel_extensions
5: max_matches = 2
6: Input graph G where |E(G)| ≤ maxedges
7: Compute D(G)
8: Compute all 1-vertex extensions of each card of D(G)
9: for each extension graph H computed do
10:   if H is not already in hash_table then
11:     Add H to hash_table
12:     Compute D(H)
13:     Set matches to the number of cards from D(H) matching cards of D(G)
14:     if matches ≥ 3 then
15:       Add H to rel_extensions
16:       max_matches =max(matches, max_matches)
17:     end if
18:   end if
19: end for
20: for subdeck_size = 3 to n − 1 do
21:   for each subdeck of G of size subdeck_size do
22:     found = false
23:     for each extension graph I in rel_extensions do
24:       if D(I) contains current subdeck of G then
25:         found = true
26:         break out of innermost for loop
27:       end if
28:     end for
29:     if found = false then
30:       break out of outermost for loop
31:     end if
32:   end for
33: end for
34: The subdeck_size is ∃rn(G) and max_matches + 1 is ∀rn(G)
35: if not(2|totaledges) or |E(G)| ≠ maxedges then
36:   Reconstruction numbers are assigned to Ḡ
37: end if
```

necessary to write it to the relation matrix. This is not only important to save on memory used for the relation matrix but on processing time that would have been spent creating and canonizing the deck of an extension graph already available for the input graph. Details in pseudocode of the new algorithm for finding $\exists rn(G)$ and $\forall rn(G)$ for graphs of specified order are given in Algorithm 3.

The last important implementation detail regarding reconstruction number calculations has to do with Condor. The calculation of reconstruction numbers for all graphs with seven or less vertices is quick enough that the parallel processing power of Condor is not necessary. For these cases, the single *reconstruction* executable is run with the number of vertices specified as a command line parameter. This quickly generates output files containing reconstruction number counts and graphs with notably high reconstruction numbers.

For sets of graphs with more vertices, Condor is helpful. To use Condor, the calculation of reconstruction numbers were split up into three separate modules: one to generate input graphs, another to calculate reconstruction numbers for a set of graphs and a module to process reconstruction number output for sets of graphs as they are created. The proper invocation of these programs is coordinated by the *run_recon* shell script. Parameters within the shell script can be changed to affect the maximum number of input files that can be present at the same time and the number of graphs within each input file, among other settings. Communication between these programs, which were scattered across several machines, is achieved with files stored on an NFS.

A specialized module was created to generate input since storing all input graphs into files at the same time would take up too much disk space. Instead, program *createInput* creates input files as they are needed by reconstruction number calculators assigned as Condor jobs. The *createInput* program creates a specified number of input files (as dictated by *run_recon*) for *reconstructNums* jobs to work on in parallel. When a *reconstructNums* job is finished with its input file, it deletes it. When *createInput* sees it needs to generate more input files it simply creates new ones containing input graphs from the point it left off. By specifying how many input files *createInput* must keep present at a time, we are specifying how many Condor jobs are calculating in parallel.

A *reconstructNums* Condor job is submitted by *run_recon* whenever a new input file is generated. Each of these jobs calculates reconstruction numbers for graphs in its assigned input file, deletes the input file, then generates one reconstruction number output file for existential data and another for universal data. It is these jobs that are utilized for parallel processing.

Sample output files generated by *reconstructNums* for a file of input graphs generated by *createInput* is given in Appendix A.1. This particular run of *reconstructNums* was performed on a set of 300 input graphs (the number specified by *run_recon*) of order eight, beginning with the 600th generated graph. The sequential number of the first processed graph and the order of graphs being processed are shown within the output file names.

The file **graphs8e600.out** is a summary of $\exists rn$ calculations of the input graphs. First is listed the exceptions – all graphs found to have high $\exists rn$ values. In this case one graph with **graph6** notation G‘?G?C and its complement are

found to have a $\exists rn$ value of four. Lastly come the $\exists rn$ count totals – 569 graphs G where $\exists rn(G) = 3$ and and two graphs with a $\exists rn$ value of four. The reason why results for more than 300 graphs were generated was that results for input graphs were also assigned to their complements when possible. Remember, assignment to complements is not done for input graphs G in cases where $|E(G)| = \frac{|V(G)|\times(|V(G)|-1)}{4}$. This explains why in the example the number of graphs assigned $\exists rn$ values is less than 600.

The file **graphs8u600.out** is a summary of $\forall rn$ calculations for the same set of graphs. First the exceptions given are the set graphs within the input set found with the highest $\forall rn$ values – in this case, six. These results are only kept for the final output of all graphs of order eight if no graphs of that order are found with higher $\forall rn$ values. If higher values are found, *processOutput* throws out these results in favor of the graphs with the highest $\forall rn$ values. Finally come the $\forall rn$ count totals.

The last module, *processOutput* simply sequentially processes the existential and universal reconstruction number output files generated by all *reconstructNums*, deleting them as it finishes each. When it has completed processing all of these files, it simply outputs two final output files, summarizing relevant reconstruction number data for existential and universal values.

The final output files generated by *processOutput* for graphs with eight vertices is given in Appendix A.2. As expected, **graphs8e.out** is a summary of $\exists rn$ values for all graphs of order eight and **graphs8u.out** is a summary of their $\forall rn$ values. The existential reconstruction number summary shows all graphs G of order eight where $\exists rn(G) > 3$ and the $\exists rn$ count totals for all graphs of order eight. Files **graphs8u.out** shows the set of all graphs of order eight with the highest $\forall rn(G)$ value and a $\forall rn$ count summary of all graphs of order eight. Notice that it threw out the exception results of **graphs8u600.out** since graphs were found with a $\forall rn$ value of seven.

3.7 2-Reconstructible Calculations

The algorithm for determining whether or not a graph is 2-reconstructible is very similar to that used to calculate the reconstruction number of a graph. First, given graph G, $D_2(G)$ is found and each of the $\binom{n}{2}$ deck's cards is canonized. Next, for every card of $D_2(G)$, all possible 2-vertex extensions are found and canonized.

From card F of order n, there are 2^{2n+1} possible extension graphs where isomorphism is not taken into consideration. Similar to the new algorithm for finding reconstruction numbers, a hash table is used so that the same extension graph is not used twice when determining whether or not a single input graph is 2-reconstructible.

As each unique extension graph is found, its 2-vertex deleted deck is determined and each card is canonized. If one of these decks is found to be the same as $D_2(G)$ for input graph G, the graph is not 2-reconstructible. If none of the extension graphs have the same 2-vertex deletion deck as G, then G is 2-reconstructible.

3.8 Run-time Complexity Estimates

When calculating reconstruction numbers for graphs, the majority of time was spent canonizing graphs. The *gprof* profiling tool was used to analyze the *reconstruction* program when calculating the reconstruction numbers for all graphs with eight vertices. While running, the program spent 72.9% of the time in the *makecanon()* function – a function copied directly from the **nauty** package which canonizes a given graph. The function was called a total of 34,519,304 times to calculate reconstruction numbers for the 12,346 graphs with eight vertices. Given a single input graph G of order n, there are three major categories of graphs that must be canonized.

The first of these are all of the cards in $D(G)$, requiring the canonization of n graphs, each of order $n-1$. These are used to set up the relation matrix, so that the decks of relevant extension graphs are related to $D(G)$ according to isomorphic equivalence. This is only a very small fraction of the number of graphs needed to be canonized and does not take up a significant amount of processing time.

The second category of graphs needing canonization are all possible extension graphs of unique cards from $D(G)$. The number of these graphs varies from 2^{n-1} to $2^{n-1} \times n$ depending on the number of unique cards in $D(G)$. Of course, all of these graphs are of order n. These calculation are used to establish the set of unique relevant extension graphs necessary for calculating reconstruction numbers for G. Ruling out repeated extension graphs saves us many canonizations in the next category.

The last category involves taking the extension graphs generated above and canonizing each card from their decks. The maximum number of graphs possible here is $2^{n-1} \times n^2$. However, the actual total is likely to be much less when isomorphism is taken into account.

For 2-reconstructibility of input graph G, we must canonize the $\frac{n(n-1)}{2}$ cards of order $n-2$ that are in its deck. From these, a possible range of 2^{2n-3} to $2^{2n-3} \times \frac{n(n-1)}{2}$ extension graphs of order n are canonized. Consequently, a maximum of $2^{2n-3} \times \frac{n(n-1)}{2} \times \frac{n(n-1)}{2}$ graphs of order $n-2$ are canonized with the actual number depending upon isomorphism within the second set of graphs.

As shown in Table 1, as the order set of input graphs increases the number of total input graphs increases rapidly. Now consider the amount of time it would take to find reconstruction numbers for all graphs with eleven vertices. By looking at the above analysis regarding canonization, we can see that the number of graphs the algorithm canonizes grows an order of magnitude more rapidly than the number of input graphs. Given that there are nearly 100 times as many graphs of order eleven as there are of order ten, and approximately twice as many graphs need to be canonized between the two of them, about 200 times as many calls must be made to *makecanon()* to find reconstruction numbers for all graphs of eleven vertices than all of order ten. However this does not take into consideration that graphs canonized for order eleven calculations have one more vertex than those done for order ten, making each *makecanon()* call more expensive. Table 2 shows the actual amount of time the calculations took

Table 1: Number of graphs of specified order

Order	Graph Set Size
1	1
2	2
3	4
4	11
5	34
6	156
7	1,044
8	12,346
9	274,688
10	12,005,168
11	1,018,997,864

Table 2: Running times with 55 machines working in parallel

Order of Graphs	Time Finding Rec. Numbers	Time Testing 2-Recon
8	2 min.	11 min.
9	24 min.	976 min.
10	1,969 min.	n/a
11	787,600 – 1,440,000 min.[a]	n/a

[a]estimated

for higher order input sets with 55 machines working in parallel. In practice, from finding reconstruction numbers for the first five million graphs with eleven vertices, calculations were completed for approximately one million input graphs every day. These considerations were used to come up with the estimated time for finding reconstruction numbers for all graphs with eleven vertices.

4 Results

4.1 Verifying Calculations

The reconstruction number calculations were verified by comparing them to calculations for smaller graphs performed by hand. In addition, the $\exists rn$ values calculated for disconnected graphs were compared to their expected values as predicted by Theorems 1, 3 and 4. All of these graphs were calculated to be in the range of their predicted values. Finally, the results had to be tested against results from Jennifer Baldwin's project. Even though the two sets of results were extremely close, there were some discrepancies between them.

Where the counts of universal reconstruction numbers disagreed, it was proven that there was at least one graph of order eight with a high $\forall rn$ missing from the Jennifer's data set. In some cases, there were graphs found to have high existential reconstruction numbers in the recent calculations that were missing from the older data. In these instances, it was proven that newly generated graphs had at least the recently calculated reconstruction numbers.

Lastly, there were graphs from the older data set which were found with large existential reconstruction numbers that did not show up in the newer set. Unfortunately, it is much more difficult to disprove that a certain graph has a high reconstruction number. But given that discrepancies were extremely rare, and that in most of these cases they were proven to be in favor of the newer data, it is reasonable to assume that the current calculations are correct. More details on these findings are given in Section 4.4.

Since the algorithm for testing 2-reconstructibility was so similar to that used to find reconstruction numbers, we can safely assume that it is also correct. Just to be sure, graphs found not to be 2-reconstructible were easily verified by hand as the program output the two or more graphs with the same two vertex deleted decks.

4.2 High Existential Reconstruction Numbers

When analyzing the results of calculations, the most emphasis was placed on studying graphs with large existential reconstruction numbers. This was done because they are the part of the project with the most direct connection with the Reconstruction Conjecture. An attempt was made to classify all graphs G where $3 \leq |V(G)| \leq 10$ with $\exists rn(G) > 3$ into categories and explain why graphs from each category would have high reconstruction numbers. Using these categories, we can identify other graphs G where $|V(G)| > 10$ such that $\exists rn(G) > 3$.

4.2.1 Disconnected Graphs Composed of Complete Components

Table 3 shows the count of graphs with high existential reconstruction numbers arranged by order. When Jennifer analyzed her results, she hypothesized that there did not exist a graph G with an odd number of vertices where $\exists rn(G) > 3$ [2]. However, we can see that there are two graphs with nine vertices with high reconstruction numbers.

Table 3: $\exists rn(G)$ counts

$\exists rn$	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
3	4	8	34	150	1044	12,334	274,666	12,005,156
4		3		4		8		6
5				2		2	2	4
6						2		
7								2

This is not surprising considering that Myrvold proved that for any disconnected graph G of the form pK_c, $\exists rn(G) = c + 2$ [15]. This being the case, we can create graph $3K_3$ – a disconnected graph with nine vertices and an existential reconstruction number of five. An example of this was given in Figure 2. This graph and its complement are the two graphs with nine vertices with a high reconstruction number.

Graphs $2K_2$, $2K_4$, $4K_2$, $2K_3$, $3K_2$, $3K_3$, $2K_5$, $5K_2$ and their complements were all identified as having the correct existential reconstruction number in the results. All of these graphs are explained in this category.

As an aside, it is easy to establish that for any composite number n there exists a graph G of order n such that $\exists rn(G) > 3$. This follows as a simple corollary of Theorem 3.

Corollary 3. *Given non-prime integer n, there exists a graph G where $|V(G)| = n$ such that $\exists rn(G) > 3$.*

Proof. Given non-prime integer n and integer $d \geq 2$ such that $d|n$, we can construct graph G of the form pK_d. Applying Theorem 3, $\exists rn(G) = d + 2$. □

However, this still does not answer whether or not there exists a graph G where $|V(G)|$ is prime and $\exists rn(G) > 3$. In the results, no such example was found. It is unfortunate that reconstruction number calculations could not be completed for all graphs with eleven vertices since this would provide data for a much larger set of graphs with a prime number of vertices. It is interesting to note that all categories of graphs identified as having high existential reconstruction numbers require a non-prime number of vertices.

4.2.2 Other Graphs Composed of Isomorphic Components

Unlike the last category, not all disconnected graphs composed entirely of isomorphic components have high existential reconstruction numbers. The reader

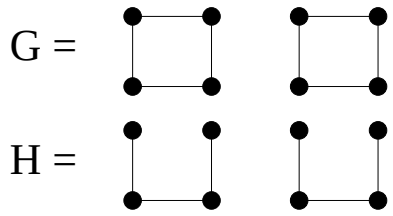


Figure 5: $\exists rn(G) = \exists rn(H) = 4$

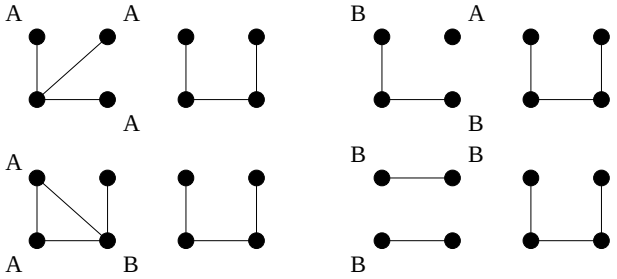


Figure 6: Proof $\exists rn(H) > 3$

will recall that it was recently proved that if G is a disconnected graph of isomorphic components of order c and the components are not isomorphs of K_c, then $\exists rn(G) \leq c$ [1]. There were only two graphs in this category found to have high existential reconstruction numbers, and both had the maximum value possible. They are shown in Figure 5.

Looking at Figure 5, with C_G as a component of G, $\exists rn(C_G) = 4$. The reason for this is $C_G \cong K_{2,2}$. Notice that every card in $D(G)$ is the same. Therefore, there is one unique subset of $D(G)$ for any given subset size. When this holds true for disconnected graphs of isomorphic components, the existential reconstruction number must be at least as large as $\exists rn(C_G)$ – its component graph.

As a side note, consider some general disconnected graph G of the form pC where $D(C)$ is composed entirely of isomorphic subgraphs. Since in this situation $D(G)$ is also composed entirely of isomorphic subgraphs then $\exists rn(G) \geq \exists rn(C)$. (See Proposition 2 for details). Combined with the knowledge that $\exists rn(pC) \leq |V(C)|$ where C is not complete [1] it is proven that $\exists rn(C) \leq |V(C)|$ thus proving graphs, excluding the complete variety, whose deck are composed entirely of several instances of the same card are reconstructible.

Asciak and Lauri gave a proof that all graphs with decks of entirely isomorphic components must be regular or something they label as *quasi-regular*. A graph G of order n is *quasi-regular iff* it contains a vertex u of order $n-1$ such that $G - u$ is regular. However, this is nothing new as the reconstructibility of regular and quasi-regular graphs have already been proven [1].

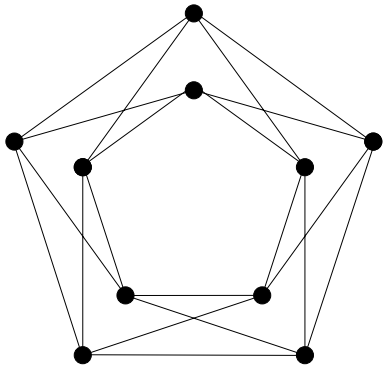


Figure 7: $\exists rn(G) = 4$

Proposition 2. *For any disconnected graph G of the form pC where the elements of $D(C)$ are all isomorphic, $\exists rn(G) \geq \exists rn(C)$.*

Proof. Given graph G of the form pC where the elements of $D(C)$ are all isomorphic, it must also be true that the elements of $D(G)$ are all isomorphic. There is only one unique subset of $D(G)$ for any given subset size. Given graph H such that $D(H)$ has $\exists rn(C)-1$ cards in common with $D(C)$, we can compose disconnected graph $I \cong H \cup (p-1)C$. Graph G shares $\exists rn(C) - 1$ cards with I, forcing $\exists rn(G) \geq \exists rn(C)$. □

With graph H from Figure 5, it is also the case that for its components, C_H, $\exists rn(H) = \exists rn(C_H)$. But unlike graph G, this fact cannot be easily established without analyzing the cards of $D(H)$ and finding other graphs which contain every possible 3-card subdeck of $D(H)$ in their own decks.

Deleting each vertex from the top row of H in Figure 5 creates a set of four isomorphic cards which we label A, and deleting vertices from the bottom row of H leaves us with another set of isomorphs to be labeled B. Figure 6 shows four graphs whose decks contain all possible multisubsets with three elements from $D(H)$. The labeled vertices indicate which can be deleted to recreate cards A or B from $D(H)$.

When considering some disconnected graph G of the form $2C$ where C is not complete, we might wonder whether it must be true that $\exists rn(G) \geq \exists rn(C_G)$. No exception to this rule was found in the results. By using Theorem 4 and considering that we already know that all complete graphs are reconstructible, we may deduce that to prove this would be to prove the Reconstruction Conjecture as well.

4.2.3 Regular Graphs of Redundantly Connected Cycles

For an example of what this report labels as regular graphs of redundantly connected cycles, see Figure 7. This is the only example of this category that

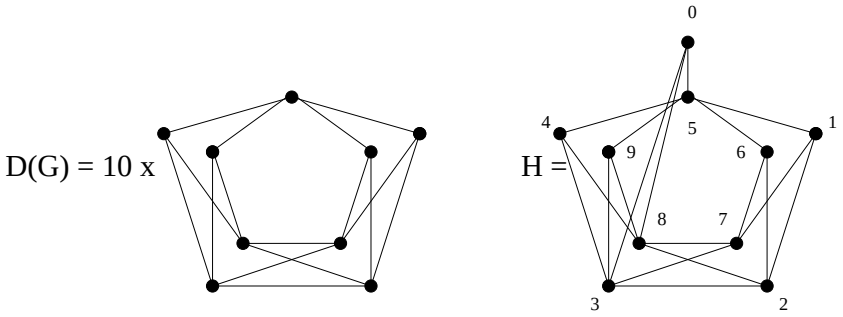


Figure 8: $H - v_0 \simeq H - v_4 \simeq H - v_9 \simeq$ card of $D(G)$

was identified in the results, but we can deduce that there are an infinite number of graphs that share this property.

Graph G is a regular graph where all cards in $D(G)$ are isomorphic. Since all cards are isomorphic, there is one unique subset of $D(G)$ for any given subset size. So to prove that $\exists rn(G) > 3$, we only need to construct one graph that shares three cards in its deck with that of $D(G)$. This is shown in Figure 8.

The graph on the left is isomorphic to every card in $D(G)$. The graph on the right, H, is a graph that is not isomorphic to G that contains three copies of the cards from $D(G)$ in its own deck. These cards are created by deleting vertices v_0, v_4 and v_9.

The reason this works is simple. Notice how vertices v_4 and v_9 are both connected to v_3, v_5 and v_8 and no others. We compose H by adding v_0 and connecting them to these vertices shared by v_4 and v_9. With this established, if we delete either v_4 or v_9 then by moving v_0 to the place of the deleted vertex while preserving its edges, we have reproduced the graph on the left – the two are isomorphic. And of course, deleting v_0 will recreate the card as well. It is also true that connections to same vertices occurs between v_6 and v_1, v_2 and v_7, and v_3 and v_8. In short, it holds for any two vertices from the card of $D(G)$ that appear in the inner and outer polygon and are spatially adjacent to each other. Composing a new graph using any of these two sets of vertices to dictate the connections of v_0 will also create a graph with three cards in its deck in common with $D(G)$.

All of the above holds true for any graph composed of two cycles of the same length with similar connections as shown in graph I of Figure 9. The minimum existential reconstruction number of these graphs can be increased by adding more cycles, as shown by graph J from the figure. J is similar to I in that all of its cards are isomorphs and there are sets of vertices connected to the same destination vertices. However, there are sets of three vertices with the same connections in J, meaning that we can compose an extension to a card of $D(J)$ where a set of four vertices share the same connections. A deletion of each of these four vertices creates an isomorph of a card of $D(J)$. Therefore, $\exists rn(J) > 4$. With this information, we can create a formal definition for regular

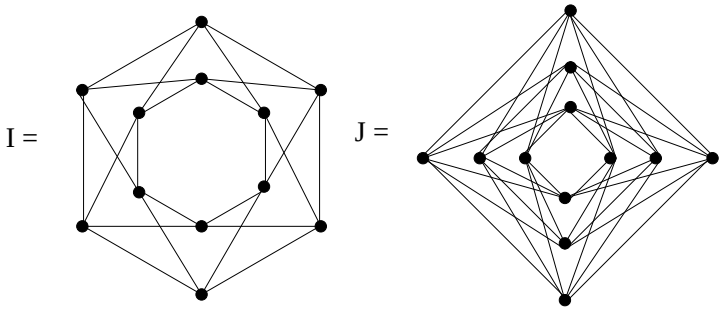


Figure 9: $\exists rn(I) > 3$ and $\exists rn(J) > 4$

graphs of redundantly connected cycles.

First, a few new symbolic conventions are needed. $C(G)$ is the set of cycles of graph G. Notation $v_a \sim v_b$ states that vertex a is connected to vertex b. Finally, for the following definition $v_{c,i}$ identifies vertex i of cycle c.

We define $RCC(n, l)$ to be a regular graph with n redundantly connected cycles, each of length l. So in Figure 9 $I \cong RCC(2, 6)$ and $J \cong RCC(3, 4)$. To compose a graph G of the form $RCC(n, l)$ we begin with graph F where F is the union of n cycles, each of length l. Let the vertices of each $c \in C(F)$ be labeled such that $v_{c,i} \sim v_{c,(i+1) \ mod \ l}$ where $0 \leq i \leq l-1$. Then by adding edges between the cycles of F such that $v_{c,i} \sim v_{d,(i+1) \ mod \ l}$ where $c \neq d$ we create graph G of the form $RCC(n, l)$.

Theorem 5. *Given graph G, of the form $RCC(n, l)$, $\exists rn(G) > n + 1$.*

Although graphs in this category appear among the set of graphs with six, eight and nine vertices, their high $\exists rn$ values were accounted for by other reasons as $RCC(2, 3) \cong K_{2,2,2}$, $RCC(3, 3) \cong K_{3,3,3}$ and $RCC(2, 4) \cong K_{4,4}$. Since the complement of any disconnected graph of the form pK_c is a connected graph whose deck contains isomorphic elements, it is not surprising that there is some overlap between them and the graphs in this section. The following remarks show two classes of graphs that have this overlap as a property.

Remark 2. Given graph G of the form $K_{c,c}$ where $2|c$, $G \cong RCC(\frac{c}{2}, 4)$.

Remark 3. Given graph G of the form $K_{c,c,c}$, $G \cong RCC(c, 3)$.

Given Remark 2, we can see that graph J from Figure 9 is an isomorph of $K_{6,6}$. It may be the case that there are other classes of graphs that also belong to these two categories simultaneously.

Finally, since graphs in this category have decks containing all isomorphic elements, given Proposition 2 we can build other disconnected graphs with large $\exists rn$ values. The result is the following corollary.

Corollary 4. *Given disconnected graph G of the form pC where C is of the form $RCC(n, l)$, $\exists rn(G) > n + 1$.*

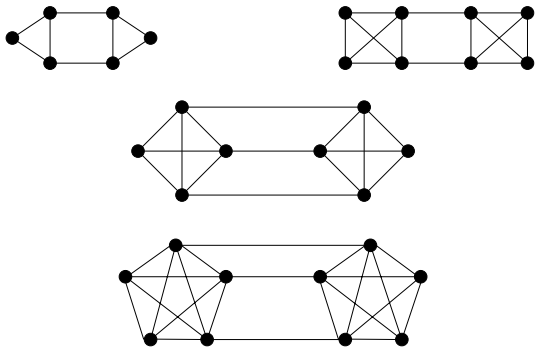


Figure 10: Examples of 1-1 connected K_c pairs

Table 4: $\exists rn$ values for connected K_c pairs

G	$\exists rn(G)$
$K_3 \leftrightarrow^2 K_3$	4
$K_4 \leftrightarrow^2 K_4$	4
$K_4 \leftrightarrow^3 K_4$	5
$K_5 \leftrightarrow^2 K_5$	4
$K_5 \leftrightarrow^3 K_5$	5
$K_5 \leftrightarrow^4 K_5$	5

4.2.4 Pairs of Complete Graphs Connected by 1-1 Edges

For some graph G to be in this category G must be made entirely of two complete K_c subgraphs which are connected to each other by b edges where $2 \leq b < c$. Finally, each vertex must be connected to at most one vertex from the opposite K_c. Examples are given in Figure 10.

Because the two subgraphs being connected are complete, a description of the order of the complete subgraphs and the number of one-to-one edges connecting them specifies a single, isomorphically unique graph. Looking at Figure 10, the given graphs can be labeled as $K_3 \leftrightarrow^2 K_3$, $K_4 \leftrightarrow^2 K_4$, $K_4 \leftrightarrow^3 K_4$ and $K_5 \leftrightarrow^3 K_5$.

To understand why the graphs in this category have large $\exists rn$ values, we must consider the contents of their decks. Every graph of this type has exactly two sets of cards in its deck.

Theorem 6. *For any graph G of the form $K_c \leftrightarrow^b K_c$ where $2 \leq b \leq c-1$, $\exists rn(G) > 3$.*

Proof. Given graph G in the form $K_c \leftrightarrow^b K_c$, $D(G)$ contains copies of $K_c \leftrightarrow^{b-1} K_{c-1}$, subgraph A, and $K_c \leftrightarrow^b K_{c-1}$, subgraph B, exclusively. Therefore, to prove $\exists rn(G) > 3$ we must prove there exists graphs whose decks contain three elements of A, two of A and one of B, two of B and one of A, and finally three

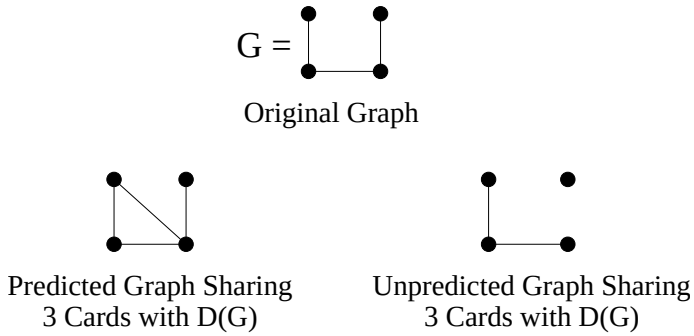


Figure 11: Graph exception with high $\exists rn(G)$

copies of B.

Given graph $K_{c+1} \leftrightarrow^{b-1} K_{c-1}$, which we label graph H, a subgraph isomorphic to A is created for each vertex deleted from K_{c+1} not connected to the labeled K_{c-1}. This means that H contains $c-b+2$ copies of A in its deck. But since we assumed that $c > b$ in graph G, the deck of H contains at least three copies of A.

With graph $K_{c+1} \leftrightarrow^{b} K_{c-1}$, which we label graph I, a subgraph isomorphic to A is created with each vertex deleted from K_{c+1} that is not connected to the opposite K_{c-1}, and we attain a subgraph isomorphic to B for each remaining vertex we delete from the K_{c+1} piece. This gives us b copies of A and $c-b+1$ copies of B in $D(I)$. Since we assume that $b \geq 2$ for graph G, that leaves us with at least two copies of A in $D(I)$ and as $c > b$ there are at least two isomorphs of B in $D(I)$. Therefore, from $D(I)$ we can extract a multisubset with two copies of A and one copy of B and a multisubset with two copies of B and one copy of A.

Finally we consider $K_{c+1} \leftrightarrow^{b+1} K_{c-1}$, or graph J. By deleting each vertex from the K_{c+1} piece that is connected to K_{c-1} by one of the $b+1$ edges, we obtain an isomorph of B. As $b+1 \geq 3$ this leaves us with at least three elements isomorphic to B from $D(J)$.

With these graphs and their noted subdecks we know $\exists rn(G) > 3$. □

In some cases, we can prove that the existential reconstruction number must be higher for graphs in this class. An example is illustrated by the following proposition.

Proposition 3. *Given graph G of the form $K_c \leftrightarrow^{c-1} K_c$, $\exists rn(G) \geq c$.*

Proof. We know that $D(G)$ consists of $2(c-1)$ copies of $K_c \leftrightarrow^{c-2} K_{c-1}$, which we label graph A, and two copies of $K_c \leftrightarrow^{c-1} K_{c-1}$, graph B. From graph $K_{c+1} \leftrightarrow^{c-1} K_{c-1}$, we have a deck containing $c-1$ copies of A and two copies of B. This guarantees that $\exists rn(G) \geq c$. □

Table 5: Predicted graphs G where $\exists rn(G) > 3$ and $|V(G)| = 12$

Minimum $\exists rn$	Predicted Graphs G Where $\|V(G)\| = 12$
4	$6K_2/\overline{6K_2}$ $3K_{2,2}/\overline{3K_{2,2}}$ $RCC(2,6)/\overline{RCC(2,6)}$ $\{K_6 \leftrightarrow^b K_6 \mid 2 \leq b \leq 4\}/\{\overline{K_6 \leftrightarrow^b K_6} \mid 2 \leq b \leq 4\}$
5	$4K_3/K_{3,3,3,3}$
6	$3K_4/K_{4,4,4}$ $K_6 \leftrightarrow^5 K_6/\overline{K_6 \leftrightarrow^5 K_6}$
8	$2K_6/K_{6,6}$

4.2.5 High $\exists rn$ Exception

The only graph identified with a high existential reconstruction number that did not fit neatly in the above categories was $K_2 \leftrightarrow^1 K_2$ having $\exists rn(G) = 4$. Although it is technically a pair of complete graph connected by a one-to-one edge, there are too few one-to-one edges to use Theorem 6 and the complete graphs connected are too small to use Proposition 3 to prove a high existential reconstruction number.

There are two possible multisubsets of size three in its deck. One graph that contains one of these multisubsets in its own deck is $K_3 \leftrightarrow^1 K_1$, which fits one of the graph constructions used in the proof of Theorem 6 ($K_{c+1} \leftrightarrow^b K_{c-1}$). However, the graph sharing the other possible three card subdeck does not match any graph constructions used in proofs from the previous section. This is illustrated in Figure 11. But since $K_2 \leftrightarrow^1 K_2$ has so few vertices, it may just be a degenerate case. It is also notable that this graph is its own complement, explaining the odd number of graphs of order four with $\exists rn(G) = 4$.

4.2.6 Predicting Graphs with High $\exists rn$

Given what we know about categories of graphs with high existential reconstruction numbers, we can predict a number of graphs with some given number of vertices where $\exists rn > 3$. Here we will use the set of graphs G where $|V(G)| = 12$ for an example.

The first graphs we can look for are disconnected graphs of the form pK_c simply by considering multiples of the order in question. For graphs of order twelve these graphs are $2K_6$, $3K_4$, $4K_3$ and $6K_2$. The $\exists rn$ values for all of these graphs are $c + 2$ for themselves and their associated complements.

Next we can consider graphs of the form $K_c \leftrightarrow^b K_c$ where $2 \leq b \leq c - 1$. This means that within the set of all graphs of order twelve, graphs $K_6 \leftrightarrow^b K_6$ where $2 \leq b \leq 5$ and their complements all have $\exists rn$ values greater than three.

Another consideration are disconnected graphs of the form pC. Although

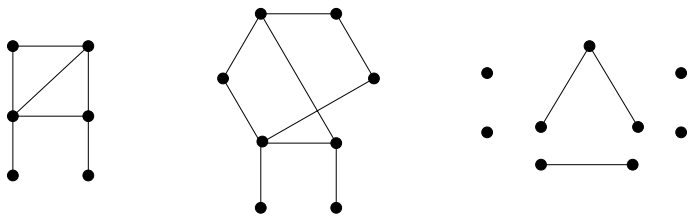


Figure 12: Example graphs with maximal $\forall rn$ values

there is currently no known easy method for determining which of these graphs have high $\exists rn$ values, there are a few specific graphs of this form to consider. First of all, we know that where graph G is $K_{2,2}$, $\exists rn(G) = 4$. Given Proposition 2 and Theorem 4, we can deduce that for any graph H of the form pG, $\exists rn(G) = 4$. So among the set of graphs of order twelve, $3K_{2,2}$ and its complement has an $\exists rn$ value of four. Other graphs to check within the set of graphs of order n would be disconnected graphs of the form pC where $p|n$ and C is of the form $RCC(n, l)$. However, in the case where $n = 12$, all possible regular graphs of this form are already accounted for as complements of disconnected graphs of the form pK_c.

The last category of graphs is the set of possible of RCC graphs of the specified order. In the case of order twelve, these graphs composed of cycles of length three and four have already been accounted for given Remarks 2 and 3. This leaves only $RCC(2, 6)$ (graph I from Figure 9) and its complement. Consequently, $\exists rn(I) \geq 4$ and $\exists rn(\overline{I}) \geq 4$. These predictions are summarized in Table 5.

4.3 Universal Reconstruction Number Statistics

The universal reconstruction number totals arranged according to order are given in Table 6. Although it has been proven that for the majority of graphs G that $\exists rn(G) = 3$, the calculated $\forall rn$ value totals seem to indicate there is no single universal reconstruction number that holds for the majority of graphs.

For this data set, the maximum count of $\forall rn$ values among graphs of a specified order seems to change unpredictably. Surprisingly enough, this maximum count for graphs of order ten was for $\forall rn(G) = 3$. This only occurs again in the results among graphs of order three. For most graphs G in Table 6, $3 \leq \forall rn(G) \leq 5$, but in the absence of additional information this may not remain true as the order of graphs considered increases.

There are simply too many graphs with "high" universal reconstruction numbers to be analyzed as extensively as was done for existential reconstruction numbers. Even after narrowing the analysis to the set of graphs with maximal $\forall rn$ values among graphs of each order, no trends could be identified for a general explanation of their causes.

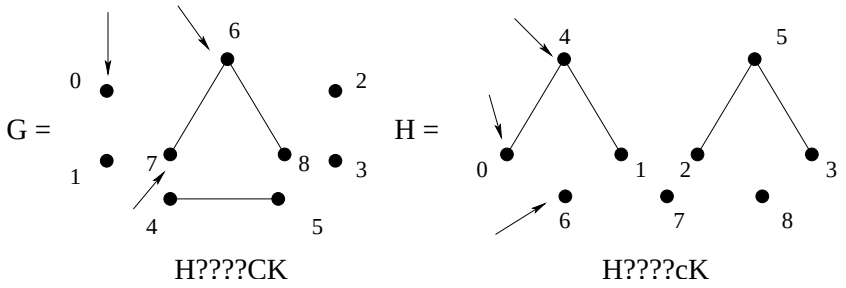


Figure 13: $\forall rn(G) = \forall rn(H) = 7$

Table 6: $\forall rn$ totals

$\forall rn$	n = 3	n = 4	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
3	3	2	7	8	16	266	45,186	6,054,148
4		9	19	56	496	8,308	199,247	5,637,886
5			8	90	520	3,584	28,781	301,530
6				2	12	284	1,434	10,686
7						4	20	914
8								4

Figure 12 shows examples of graphs with maximal $\forall rn$ values among the set of graphs of order six, eight and nine. It is interesting to note that where graph G is H????CK in **graph6** notation, $\exists rn(G) = 3$ according to Theorem 1, and yet $\forall rn(G)$ was calculated to be as high as seven.

Figure 13 explains this property for graph H????CK. Graph G (H????CK) and graph H (H????cK) have six cards from their decks in common. Deleting vertices v_0, v_1, v_2 and v_3 from both graphs create four of these common cards and deleting vertices v_4 and v_5 from both create the other two, proving that $\forall rn(G) \geq 7$ and $\forall rn(H) \geq 7$. By applying the algorithm from [14], the arrows in the figure point to three vertices from G and H whose deletion will create unique subdecks for each. $\exists rn(G) = \exists rn(H) = 3$.

4.4 Discrepancies Between Old and New Data Sets

4.4.1 $\exists rn$ Discrepancies

The $\exists rn(G)$ counts found in old and new data sets matched perfectly where $3 \leq |V(G)| \leq 8$. The $\exists rn(G)$ values for individual graphs also matched where $3 \leq |V(G)| \leq 7$. However, the graphs of order eight found with high $\exists rn$ and their values differed slightly between the two.

The $\exists rn$ values for graphs $4K_2$, $K_{2,2,2,2}$, $2K_4$, $K_{4,4}$, $2[K_2 \leftrightarrow^1 K_2]$, $2\overline{K_2}$ and $\overline{2\overline{K_2}}$ were all in agreement. For $K_4 \leftrightarrow^2 K_4$, $\exists rn$ was calculated as five in the old data, but four in the new results. Graph $\overline{K_4 \leftrightarrow^2 K_4}$ was not found to have a high

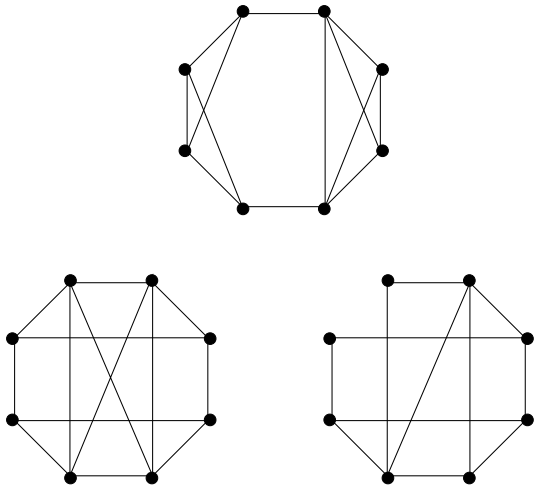


Figure 14: Unmatched graphs from old data set

Table 7: $\forall rn$ total discrepancies where $|V(G)| = 8$

$\forall rn(G)$	Old Totals	Current Totals
4	8,209	8,308
5	3,576	3,584
6	292	284
7	3	4

$\exists rn$ value in the old data, indicating an error. Graph $\overline{2[K_2 \leftrightarrow^1 K_2]}$ was another graph proven to have a high $\exists rn$ that was missing from the old calculations.

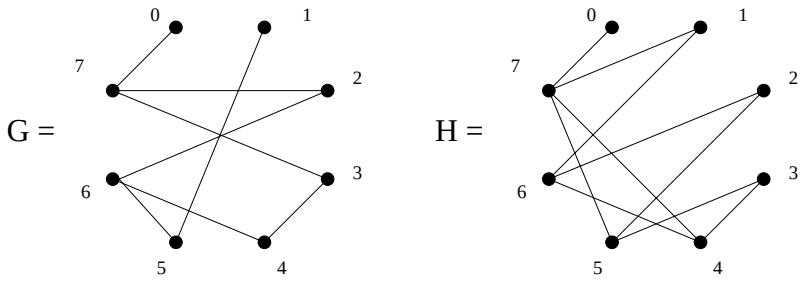
Another graph with a high $\exists rn$ value missing from the old data was $K_4 \leftrightarrow^3 K_4$. Because of Proposition 3, we know that $\exists rn(G) \geq 4$. The $\exists rn$ value for this graph in the new data was five. The graph's complement was also missing from the old data.

Three graphs of order eight which were identified in the old data as having high $\exists rn$ values which were not found in the new data are given in Figure 14. It may be the case that these graphs were simply not drawn correctly in the old writeup. Unfortunately, the **graph6** labels of these graphs were not available from the old data to check whether this was the case.

4.4.2 $\forall rn$ Discrepancies

The totals for $\forall rn$ values for all graphs G where $3 \leq |V(G)| \leq 7$ found in the old and new results agreed. However, the totals for graphs with eight vertices were not the same. These differences are shown in Table 7.

In order to support the newer data set, we will prove that there are at least



G?D@\g G?CrUW

Figure 15: Two graphs with $\forall rn$ values ≥ 7

four graphs G where $|V(G)| = 8$ and $\forall rn(G) \geq 7$. Although this does not prove the correctness of the new data, at least we know that it should not exactly match the old data set.

Figure 15 shows two graphs who share six cards between them from their decks. They are labeled according to their **graph6** notation from **nauty**. The six matching cards are $G - v_0$ and $H - v_1$, $G - v_1$ and $H - v_3$, $G - v_2$ and $H - v_4$, $G - v_3$ and $H - v_5$, $G - v_4$ and $H - v_6$, and finally $G - v_5$ and $H - v_7$. By confirming by hand, we can verify that each pair of subgraphs are isomorphs, and no pair is isomorphic to another. The six shared cards confirms that both $\forall rn(G)$ and $\forall rn(H)$ must be at least seven.

Furthermore, G has eight edges while H has ten. Given that a graph with eight vertices can have a maximum of 28 edges, it is not possible that G and H are complements of each other nor can it be the case that either G or H are complements of themselves. So not only must $\forall rn(G) \geq 7$ and $\forall rn(H) \geq 7$, but $\forall rn(\overline{G}) \geq 7$ and $\forall rn(\overline{H}) \geq 7$ as well. There must be at least four graphs G where $|V(G)| = 8$ and $\forall rn(G) \geq 7$.

4.5 Graphs That Are Not 2-Reconstructible

Calculations testing the 2-reconstructibility of graphs was completed for all graphs G where $|V(G)| \leq 9$. In these calculations, seven of the eleven possible graphs with four vertices were found not to be 2-reconstructible. This is not surprising considering the low number of vertices compared to the number being deleted.

There were four graphs found with five vertices that were not 2-reconstructible. These were the two graphs pictured in Figure 16 and their complements. It is easy to verify by hand that these two graphs have the same 2-vertex deleted decks.

For all graphs G where $6 \leq |V(G)| \leq 9$, G was found to be 2-reconstructible. It may well be the case that all graphs with more than five vertices are 2-reconstructible. As mentioned in Section 1.2, we can generalize the Reconstruction Conjecture to ask, given a number k, what is the minimum order n such

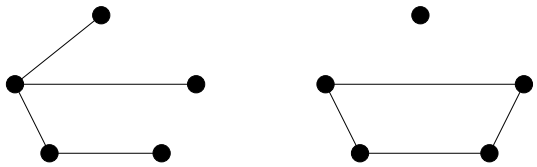


Figure 16: Two graphs with the same 2-vertex deleted decks

that all graphs, G, where $|V(G)| \geq n$ are reconstructible from $D_k(G)$? The Reconstruction Conjecture assumes that when $k = 1$, the minimum order is three.

Some notable properties of the two graphs in Figure 16 are that one is a tree while the other is a disconnected graph. What is notable is that although both trees and disconnected graphs have been proven to be reconstructible when just one vertex is deleted, this does not hold for 2-reconstructibility.

5 Conclusions

The resulting data and analysis from this report point to several topics that are worth further research. Most interesting of these is the question of whether there exists any graphs G with a prime number of vertices such that $\exists rn(G) > 3$. In addition, when considering the categories of graphs identified as having high existential reconstruction numbers, it is worth examining whether there are additional categories to be found containing higher ordered graphs. Also, can we find a more precise range of possible $\exists rn$ values for categories of graphs identified in this report? Specifically, it seems that with a bit more analysis we may easily narrow the range of possible $\exists rn$ values for graphs of the form $K_c \leftrightarrow^b K_c$. Lastly we come to the topic of 2-reconstructibility. Of all graphs analyzed for 2-reconstructibility, graphs G where $3 \leq |V(G)| \leq 9$, two graphs of order five and their complements were identified as the graphs with the most vertices to not be 2-reconstructible. Could it be the case that for all graphs G where $|V(G)| > 5$, G must be 2-reconstructible? These considerations are discussed in more detail below.

First and foremost, do there exist any graphs G of prime order such that $\exists rn(G) > 3$? Since the explanations behind high $\exists rn$ values identified depend on a non-prime number of vertices, the data seems to suggest that they do not. However, this does not disprove their existence. Completing reconstruction number calculations for graphs with eleven may help, but they can only solve the question by identifying a graph G with eleven vertices where $\exists rn(G) > 3$.

The next question to consider is whether or not there are graphs with high $\exists rn$ values that belong to categories other than the four discussed in this report. Recall that graphs in the identified categories contained only one or two unique cards in their decks. Considering that there exist much larger graphs existing with many more combinations of possible subdecks, it is doubtful that these are

the only categories of graphs with high $\exists rn$ values. It may even be the case that no finite set of categories can encompass all graphs G with $\exists rn(G) > 3$.

Notice that in the discussion of categories of graphs identified with high $\exists rn$ values, much of the work was put into establishing lower bounds for existential reconstruction number values. It would be more interesting to establish upper bound $\exists rn$ values instead. Due to prior research, the exact value of $\exists rn(G)$ for disconnected graphs of the form pK_c, and an upper bound $\exists rn(G)$ for disconnected graphs of the form pC are already known. However, little is known regarding upper bound $\exists rn$ values for graphs of the form $K_c \leftrightarrow^b K_c$ and regular graphs of redundantly connected cycles. We know that for any RCC graph G that $\exists rn(G) \leq |V(G)|$ since regular graphs are known to be reconstructible. But the reconstructibility of graphs of the form $K_c \leftrightarrow^b K_c$ is still open, and may be easy to establish.

While on the topic, given a graph G of the form $K_c \leftrightarrow^b K_c$, can we use b and c to calculate the exact value for $\exists rn(G)$? This report only establishes that all graphs of this form must have high $\exists rn$ values and gives a tentative lower bound for graphs of the form $K_c \leftrightarrow^{c-1} K_c$. This leaves much room for improvement.

From the topic of 2-reconstructibility comes the question of whether there exists any graphs G where $|V(G)| > 5$ such that G is not 2-reconstructible. This probably will not be solved any time soon since the seemingly easier question of whether there exists any graphs G where $|V(G)| \geq 3$ that are not 1-reconstructible (the Reconstruction Conjecture) is still an open question.

Finally, we must consider if there are ways to significantly improve the efficiency of the algorithms employed to calculate reconstruction numbers. One portion to consider would be the generation of all possible extension graphs of a given graph. In the current implementation, $2^{|V(G)|}$ extensions of graph G are exhaustively generated and canonized although the resulting set of isomorphically unique extension graphs usually turns out to be much smaller. However the *geng* program from the **nauty** package, which can generate an exhaustive set of isomorphically unique graphs of a given order, does no canonization to generate its results unless the user specifies that she wants the output presented in canonized form. If we can find a method to generate extensions of graphs that significantly cuts back on the number of calls to the expensive *makecanon()* function, we may be able to analyze many more graphs than are currently possible. A detailed description of "a very general technique for generating families of combinatorial objects without isomorphs" is given by Brendan McKay in [12].

A Sample Output Files

A.1 Output Files from *reconstructNums*

Listing of graphs8e600.out:

```
Exceptions
4 G‘?G?C
Complement G]~v~w
Totals
3   569
4     2
```

Listing of graphs8u600.out:

```
Exceptions 6
6 G??tY{
Complement G~~Id?
6 G?B@x{
Complement G~{}E?
6 G??xuK
Complement G~~EHo
6 G?@XvO
Complement G~}eGk
6 G?@Xvo
Complement G~}eGK
6 G?Bapo
Complement G~{\MK
6 G?AZRo
Complement G~|ckK
6 G??~Uo
Complement G~~?hK
6 G?AZvo
Complement G~|cGK
6 G@Ge}w
Complement G}vX@C
6 G?B\ro
Complement G~{aKK
6 G?Azvo
Complement G~|CGK
6 G?@~vo
6 G?O__[
Complement G~n^^_
6 GGC?KK
Complement Gvz~ro
6 G?O_c[
Complement G~n^Z_
```

```
6 GGC?G{
Complement Gvz~v?
Totals
3     6
4   342
5   190
6    33
```

A.2 Output Files from *processOutput*

Listing of graphs8e.out:

```
----------------------
4 G‘?G?C
Complement G]˜v˜w
----------------------
4 G?‘H?c
Complement G˜]u˜W
----------------------
4 GoCOZ?
Complement GNznc{
----------------------
5 GoSsZc
Complement GNjJcW
----------------------
4 G‘iayw
----------------------
6 G˜?GW[
Complement G?˜vf_
----------------------
4 G‘rHpk
Existential Reconstruction # Totals
3 12334
4     8
5     2
6     2
```

Listing of graphs8u.out:

```
Exceptions 7
--------------------
7 G?D@\g
Complement G˜y}aS
--------------------
7 G?CrUW
Complement G˜zKhc
Universal Reconstruction # Totals
3   266
4  8208
5  3584
6   284
7     4
```

References

[1] K. J. Asciak and J. Lauri, On Disconnected Graph with Large Reconstruction Number. *Ars Combinatoria*, 62:173-181, 2002.

[2] J. Baldwin, *Graph Reconstruction Numbers*, Master's Project, Rochester Institute of Technology, 2004.

[3] B. Bollobás. Almost every graph has reconstruction number three. *Journal of Graph Theory*, 14(1):1-4, 1990.

[4] J. A. Bondy. A Graph Reconstructor's Manual. *Surveys in Combinatorics*, 1991 (Guildford, 1991), 221-252, London Math. Soc. Lecture Note Ser., 166, Cambridge Univ. Press, Cambridge, 1991.

[5] J. A. Bondy and R. L. Hemminger. Graph Reconstruction – a survey. *Journal of Graph Theory*, 1:227-268, 1977.

[6] Condor Team, University of Wisconsin-Madison, *Condor Version 6.6.7 Manual*, May 2004.

[7] F. Harary and M. Plantholt. The Graph Reconstruction Number. *Journal of Graph Theory*, 9:451-454, 1985.

[8] E. Hemaspaandra, L. Hemaspaandra, S. Radziszowski and R.Tripathi. Complexity Results in Graph Reconstruction. *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science*. To appear.

[9] P. J. Kelly. A congruence theorem for trees. *Pacific Journal of Mathematics*, 7:961-968, 1957.

[10] P. J. Kelly. *On Isometric Transformations*. PhD thesis, University of Wisconsin, 1942.

[11] J. Lauri and R. Scapellato. Topics in Graph Automorphisms and Reconstruction. London Mathematical Society, Cambridge University Press, 2003.

[12] B. D. McKay. Isomorph-Free Exhaustive Generation. *Journal of Algorithms*, 26:306-324, 1998.

[13] B. D. McKay. nauty *User's Guide (Version 2.2)*. Computer Science Department, Australian National University, bdm@cs.anu.edu.au.

[14] R. Molina. Correction of a proof on the ally-reconstruction number of disconnected graphs, *Ars Combinatoria*, 20:59-64, 1995.

[15] W. J. Myrvold, The ally reconstruction number of a disconnected graph. *Ars Combinatoria*. 28:123-127, 1989.

[16] C. St. J. A. Nash-Williams. The Reconstruction Problem. *Selected Topics in Graphs Theory*. 205-236, 1978.

[17] V. Nýdl. Finite undirected graphs which are not reconstructible from their large cardinality subgraphs. *Topological, Algebraic and Combinatorial Structures*, 1991.

[18] V. Nýdl. Graph reconstruction from subgraphs. *Discrete Mathematics* 235:335-341, 2001.