GRAPHS ADMITTING $(1, \leq 2)$-IDENTIFYING CODES

by

Julie Lang

B.S., Morehead State University, 2012

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Mathematics
Kansas State University

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Approved by:

Major Professor
Sarah Reznikoff

Copyright

Abstract

A $(1, \leq 2)$-identifying code is a subset of the vertex set C of a graph such that each pair of vertices intersects C in a distinct way. This has useful applications in locating errors in multiprocessor networks and threat monitoring. At the time of writing, there is no simply-stated rule that will indicate if a graph is $(1, \leq 2)$-identifiable. As such, we discuss properties that must be satisfied by a valid $(1, \leq 2)$-identifying code, characteristics of a graph which preclude the existence of a $(1, \leq 2)$-identifying code, and relationships between the maximum degree and order of $(1, \leq 2)$-identifiable graphs. Additionally, we show that $(1, \leq 2)$-identifiable graphs have no forbidden induced subgraphs and provide a list of $(1, \leq 2)$-identifiable graphs with minimum $(1, \leq 2)$-identifying codes indicated.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I'd like to thank Dr. R. Duane Skaggs for first introducing me to the ideas of dominating sets and identifying codes, my adviser, Dr. Sarah Reznikoff, for taking the leap to discuss my doodles with me and for her helpful and inspiring discussions, my thesis committee for keeping me honest, and my family for their love, their support, and their smiles as I parade around in an "I'm into domination" t-shirt.

Preface

A fascinating facet of many topics in graph theory is the nature of the problems it poses. Many of the problems can be stated so simply that a person who has never even heard of a graph can have a thorough understanding of the question in a matter of a few minutes. Yet finding solutions for these same problems gets very difficult very quickly and can require some sophisticated mathematics! Identifying codes offer an exquisite example of such a problem. The idea is so accessible that it can be easily explained to young children, but upon delving into the idea ever so slightly, one finds a wealth of nuance and intriguing perplexities. Perhaps this is due to the juxtaposition of the very rudimentary nature of a graph as the bare connections in a set of elements, combined with how incredibly varied and complicated these networks can be. Nevertheless, this paper presents new insights into one such problem: identifying pairs of vertices in graphs.

Chapter 1

Introduction

It can be quite satisfying to work in domination. Dominating sets are diversely applicable and can be used to detect the existence of errors in networks, surveillance, and even in structural analysis of RNA[1]. Taking it one step further, identifying codes are capable of indicating the location of an error in various types of systems and were originally developed with multiprocessor networks in mind. One may think of a vertex in an identifying code as a processor which contains an error-checking code that will return "1" if it and all adjacent machines are functional or "0" if an error occurs in at least one of them. In this way, one can detect an error in the system and (by the specific combination of processors which are returning "0"), determine which node is malfunctioning. In a similar fashion, identifying codes may be utilized to monitor facilities for threats[2]. Yet one of the most wonderful properties of identifying codes is that one may disregard the ample applications to consider identifying codes as purely mathematical entities, and still be left with enticing and interesting puzzles.

To explore the subject more, we begin with some definitions. Let $G = (V(G), E(G))$ be a simple, undirected, connected graph. Given a vertex $v \in V(G)$, we denote the set of vertices adjacent to v by $N(v)$ and refer to this set as the **open neighborhood of v in G**. Related to the open neighborhood, we may define $N[v] = N(v) \cup \{v\}$ and we refer

to this as the **closed neighborhood of v in G**. A **dominating set** of a graph G is a set $S \subseteq V(G)$ such that each vertex of G is either in S or is adjacent to a vertex in S. That is, $\forall v \in V(G), N[v] \cap S \neq \emptyset$. For every graph, one can find a dominating set. A specific type of dominating set, known as an **identifying code** C of G (or more generally, a **$(1, \leq 1)$-identifying code**, as we will see) is a subset of the vertex set such that $N[v] \cap C$ is distinct and nonempty for each $v \in V(G)$. Two vertices u and v are said to be **twins** if and only if $N[u] = N[v]$. A quick result is that that a graph admits an identifying code if and only if it contains no pair of twins. Graphs for which no identifying code exists are called **unidentifiable**.

When initially defining the idea of an identifying code, Karpovsky, Chakrabarty, and Levitin considered varying two parameters: a covering radius and the cardinality of sets to be identified[3]. To account for these, we may generalize our definition using I-sets:

$$I_r(G, C; X) = C \cap \bigcup_{x \in X} B_r(x), \quad \text{where} \quad X \subseteq V(G).$$

The arguments G or C are omitted when understood. The set $B_r(x)$ is the set of all vertices a distance less than or equal to r from x where the distance between two vertices is the length of a shortest path between them. If the sets $I_r(X)$ are distinct, we say that C **separates** the sets X. If the sets $I_r(X)$ are all nonempty for every nonempty set $X \in V(G)$, C is said to **cover** or **dominate** the graph G. If C both separates and dominates all sets X where $|X| \leq l$, then C is said to be an **$(r, \leq l)$-identifying code**. These are useful in detecting up to l many errors in a network, when a single node can find an error in a node a distance $\leq r$ away. If $r = 1$, such a code is occasionally referred to as an **l-set-ID code**, as exemplified in the paper by Laihonen and Moncel[4]. When r or l are not specified, they are assumed to be 1.

Thus far, over 150 papers have been published discussing identifying codes and related ideas (such as locating-dominating sets, watching systems, and discriminating codes). An

up-to-date bibliography of papers pertaining to the subject is kept by Lobstein on his webpage[5]. Those papers which investigate identifying codes generally focus on $(1, \leq 1)$-identifying codes or $(r, \leq 1)$-identifying codes. Only a few dozen have concerned themselves with varying l from 1. In these cases, the papers are usually concerned with constructing graphs with particular properties or proving theorems that only apply to particular graphs (the most common of which are binary Hamming spaces and regular 2-dimensional lattices such as the king grid). So many papers have restricted themselves to these very structured, regular graphs that one has difficulty finding information for a general graph!

Here, we will shed some light on $(1, \leq 2)$-identifying codes in a general graph. Although this itself is a specific case of $(r, \leq l)$-identifying codes, any $(1, \leq l)$-identifying code with $l \neq 1$ must also be a $(1, \leq 2)$-identifying code. So it makes sense to look at this case in particular and to explore its properties. The only other assumption we will consider is that each graph be connected. However, this restriction will not impede any application of theorems, since results can simply be applied to each component of a disconnected graph.

For ease of use in proofs, note that from the definition given by I-sets, we find that G is $(1, \leq 2)$-identifiable if and only if for all distinct sets $\{u, v\}, \{w, x\} \subset V(G)$ (where u and v may be equal, as may w and v) we have $(N[u] \cup N[v]) \cap C \neq (N[w] \cup N[x]) \cap C$. A graph which does not admit a $(1, \leq 2)$-identifying code is called **$(1, \leq 2)$-indistinguishable**, or simply **not $(1, \leq 2)$-identifiable**. Clearly, for any $(1, \leq 2)$-identifiable graph, the entire vertex set $V(G)$ is a viable $(1, \leq 2)$-identifying code. Hence to prove that a particular graph is $(1, \leq 2)$-identifiable, one need only show that $N[u] \cup N[v] \neq N[w] \cup N[x]$ for any $\{u, v\} \neq \{w, x\}$. Observe that this is analogous to demonstrating that a graph is $(1, \leq 1)$-identifiable by showing it contains no set of twins. The more interesting case attempts to minimize the cardinality of such a code. The cardinality of a minimum identifying code has conventionally been denoted γ^{ID} and therefore, we will denote the cardinality of a minimum $(1, \leq 2)$-identifying code as $\gamma^{ID}_{1,\leq 2}$ and refer to it as the **$(1, \leq 2)$-identification number**. However, it will soon be noted that determining whether a graph is even $(1, \leq 2)$-identifiable

in the first place can be challenging, so finding minimal codes will only constitute a small part of our discussion.

Chapter 2

Some known results for $(1, \leq l)$-identifying codes of a general graph

An initial look at $(1, \leq l)$-identifying codes might raise the question of why we have the "$\leq$"? For a code that identifies l many errors at a time, is it obligatory that it must separate sets of any cardinality up to l as well? It turns out that this is not true. A code which separates sets of cardinality exactly l must separate any two sets of cardinality less than l provided that the sets have equal cardinality. Suppose that C separates distinct sets of cardinality l. Now for sets A, B, each of cardinality $l-1$, choose $v \in V(G) - (A \cup B)$. The sets $A \cup \{v\}$ and $B \cup \{v\}$ are each of cardinality l. Therefore $N[A \cup \{v\}] \cap C \neq N[B \cup \{v\}] \cap C$, which implies that $N[A] \cap C \neq N[B] \cap C$. Thus C also separates sets of cardinality $l-1$. Repeating this argument $l-1$ many times will show that A and B can be separated by a code which separates sets of cardinality l if $|A| = |B| \leq l$. Yet if A happens to be a proper subset of B, this proof fails. Indeed, in Figure 2.1 below, there is an example of a graph whose vertex set separates any two sets of cardinality 2, any two sets of cardinality 1, but does not separate a set of 2 vertices from a set of a single vertex. Thus, it makes sense (and is in fact essential)

to use "$\leq$" notation.

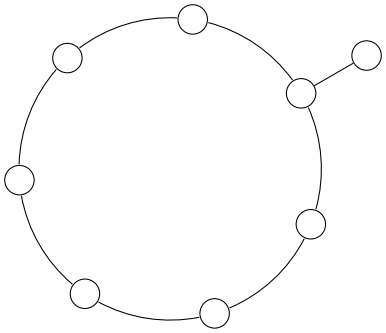


Figure 2.1: A graph for which its vertex set separates all sets of cardinality 2 but does not separate a set of cardinality 1 from a set of cardinality 2 (specifically the vertex of degree 3 and the set consisting of the vertex of degree 3 and the vertex of degree 1).

All $(1, \leq 2)$-identifying codes on a graph must also be $(1, \leq 1)$-identifying codes. In the same way, any $(1, \leq l)$-identifying code with $l \neq 1$ must therefore also be a $(1, \leq 2)$-identifying code and satisfy any mandatory conditions and properties therein. In this sense, one should note that the existence of twins excludes the existence of a $(1, \leq 2)$-identifying code.

In the first paper on the subject, Karpovsky, Chakrabarty, and Levitin proved a tight lower bound for $\gamma^{ID}_{1,\leq l}$ given n, the order of a graph. That is, $\gamma^{ID}_{1,\leq l} \geq \lceil \log_2 \sum_{i=0}^{l} \binom{n}{i} \rceil$. This proof comes from a straightforward counting argument, stemming from the fact that there are 2^n many distinct subsets of a set of n vertices. Additionally presented was a method for constructing graphs which achieve equality in the previous statement[3].

Laihonen and Moncel discuss a similar idea, describing how quickly the order of a $(1, \leq l)$-identifiable graph must grow with respect to l. Let $(G_l)_{l \in \mathbb{N}}$ be a family of graphs such that $\forall l \in \mathbb{N}$, G_l admits a $(1, \leq l)$-identifying code. Let $n_l = |V(G_l)|$. Then $n_l = \Omega(l^2)$ asymptotically[4]. Their paper does explore $(1, \leq l)$-identifying codes for a general graph, focusing mostly on asymptotic bounds. In doing so, several interesting lemmas and theorems are proven, such as the following.

Lemma (Laihonen-Moncel, Lemma 1[4]) If G is $(1, \leq l)$-identifiable, then $\exists S \subseteq V(G)$ an

independent set such that $|S| = l$.

Theorem (Laihonen-Moncel, Theorem 1[4]) If G_1 is $(1, \leq l_1)$-identifiable and G_2 is $(1, \leq l_2)$-identifiable, then the Cartesian product $G_1 \Box G_2$ is $(1, \leq max\{l_1, l_2\})$-identifiable.

It should be noted that the converse of the above theorem is not true, since $G = P_4 \Box P_4$ is $(1, \leq 2)$-identifiable, but P_4 is not.

Laihonen and Ranto briefly discuss general graphs admitting $(1, \leq l)$-identifying codes and prove that if G is $(1, \leq l)$-identifiable and $|V(G)| \geq 3$, then the minimum degree of a vertex in G, $\delta(G)$, satisfies $l \leq \delta(G)$[6]. This result is quite useful in general, but trivial in the case of the $(1, \leq 2)$-identifying code, since it simply prohibits any leaves in a $(1, \leq 2)$-identifiable graph. However, in the case of a $(1, \leq 1)$-identifying code, there are relationships and bounds between the identification number γ^{ID} and degree parameters (as investigated by Foucaud and Perarnau[7]), so we will explore a similar relationship for the more general case of $l = 2$.

Chapter 3

Additional results for graphs admitting $(1, \leq 2)$-identifying codes

Since there is relatively little known about $(1, \leq 2)$-identifying codes, this section will reveal some properties and structure of graphs admitting such codes. First, we will consider a few different classes of graphs, then we will focus on propositions concerning features of a general graph. Additionally, we have included several constructions that result in graphs which are $(1, \leq 2)$-identifiable.

Proposition 1. *If G has distinct vertices v, w with $N[v] \subseteq N[w]$, then G is $(1, \leq 2)$-indistinguishable.*

Proof. Since $N[v] \subseteq N[w]$, then $N[w] = N[\{v, w\}]$. Thus G is $(1, \leq 2)$-indistinguishable. □

A corollary to this is that any graph which contains a universal vertex (i.e., a vertex adjacent to all other vertices) is $(1, \leq 2)$-indistinguishable. Therefore, all complete graphs and star graphs are not $(1, \leq 2)$-identifiable. Star graphs can also be described as $K_{1,n}$, and it turns out that we can generalize to k-partite graphs.

Proposition 2. *All complete k-partite graphs are $(1, \leq 2)$-indistinguishable when $k > 1$.*

Proof. Let G be a k-partite graph. If each part of the graph contains only one vertex, then G is K_k, which is $(1, \leq 2)$-indistinguishable when nontrivial. So suppose there exists a part of G with more than one vertex. Say u and v are distinct vertices in the same part of G. Note that since G is a complete k-partite graph, the open neighborhood of any vertex is $V(G) - \{\text{part of } G \text{ containing } x\}$. Hence $N(u) = N(v)$ and for any vertex w in a part of G which does not contain u or v, it follows that $N[u] \cup N[w] = N[v] \cup N[w] = V(G)$. Since G has at least two parts, this shows that G is $(1, \leq 2)$-indistinguishable. □

Proposition 3. *A cycle C_n is $(1, \leq 2)$-identifiable if and only if $n \geq 7$.*

Proof. It is easy to check that C_3, C_4, C_5, and C_6 are $(1, \leq 2)$-indistinguishable. Now consider C_n where $n > 6$. Enumerate the vertices $a_0, a_1, a_2, ..., a_{n-1}$ such that vertex i is adjacent to vertex $a_{i+1 \mod n}$. Hence for any vertex a_i, $N[a_i] = \{a_{i-1}, a_i, a_{i+1 \mod n}\}$ and thus $N[a_i] \cup N[a_j] = \{a_{i-1}, a_i, a_{i+1}, a_{j-1}, a_j, a_{j+1 \mod n}\}$. We may confirm that this distinct from any other such union of up to two closed neighborhoods by observing that the order of C_n is greater than 6, so there must be at least one vertex a_m which is not in a given $N[X]$, when $|X| \leq 2$. Let k be the smallest number such that $a_{m+k \mod n} \in N[X]$ and let l be the smallest number such that $a_{m-l \mod n} \in N[X]$. Then it follows that $X = \{a_{m+k+1 \mod n}, a_{m-l-1 \mod n}\}$. Hence C_n is $(1, \leq 2)$-identifiable. □

It should be noted that checking whether or not a set C is a $(1, \leq 2)$-identifying code is tedious work; one must compare the I-set for each set of cardinality less than or equal to 2. Thus finding a valid $(1, \leq 2)$-identifying code for a given arbitrary graph is generally quite labor-intensive, let alone finding a minimal such code. It does not help that the union of two disjoint $(1, \leq 1)$-identifying codes does not necessarily form a valid $(1, \leq 2)$-identifying code. Although this may seem counterintuitive, information is lost when the two sets are joined into one. The most obvious example of this is the C_6 (see Figure 3.1), which is not $(1, \leq 2)$-identifiable, but does indeed contain two disjoint $(1, \leq 1)$-identifying codes.

One issue that comes up when studying $(1, \leq 2)$-identifying codes is that it is incredibly tedious to check $n + \frac{n(n-1)}{2}$ many closed neighborhoods intersected with a set C in order to

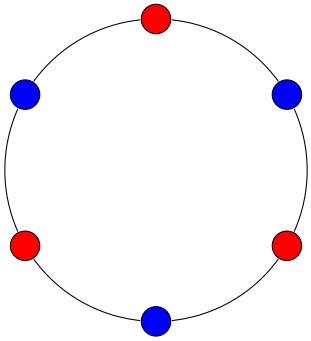


Figure 3.1: There is a set of two disjoint $(1, \leq 1)$-identifying codes for C_6 (the set of red vertices and the set of blue vertices), but the union of these sets does not form a $(1, \leq 2)$-identifying code.

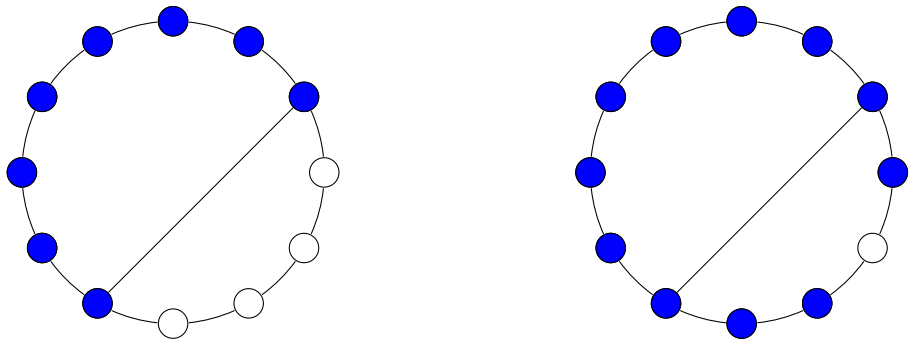
decided if C is a valid $(1, \leq 2)$-identifying code. However, there are some cases in which if C has a particular property, one may quickly rule out the possibility that it could be a $(1, \leq 2)$-identifying code. One of these ways is by looking at the induced subgraph. Given a graph $G = (V(G), E(G))$ and a set $V' \subset V(G)$, an **induced subgraph on V'** is the graph $G' = (V', E')$ where $E' = \{(u, v) | u, v \in V' \text{and} (u, v) \in E(G)\}$. We will write $G[V']$ to denote the subgraph induced by V' in G.

Theorem 4. *If C is a $(1, \leq 2)$-identifying code on some graph G, then C is also a $(1, \leq 2)$-identifying code on $G[C]$.*

Proof. Let C be a $(1, \leq 2)$-identifying code of a graph G. Then for each pair of distinct subsets $\{u, v\} \neq \{w, x\} \subseteq C$ where we permit the possibility that $u = v$ or $w = x$, we have that $(N_G[u] \cup N_G[v]) \cap C \neq (N_G[w] \cup N_G[x]) \cap C$. It follows that $(N_G[u] \cap C) \cup (N_G[v] \cap C) \neq (N_G[w] \cap C) \cup (N_G[x] \cap C)$. Observe that for any vertex $c \in C$, $N_G[c] \cap C = N_{G[C]}[c]$. Thus, $N_{G[C]}[u] \cup N_{G[C]}[v] \neq N_{G[C]}[w] \cup N_{G[C]}[x]$. Therefore, $(N_{G[C]}[u] \cup N_{G[C]}[v]) \cap C \neq (N_{G[C]}[w] \cup N_{G[C]}[x]) \cap C$ and C is a $(1, \leq 2)$-identifying code of $G[C]$. □

The converse of Theorem 4 is not true, which is easy to see because attaching a leaf to any $(1, \leq 2)$-identifiable graph results in a graph which is $(1, \leq 2)$-indistinguishable. From this proposition, it easily follows that $\gamma^{ID}_{1,\leq 2}$ is bounded below by 7, since a cycle on 7 vertices is the smallest $(1, \leq 2)$-identifiable graph. In addition, it should be noted that it's possible

and indeed quite common for there to exist $(1, \leq 2)$-identifying codes $A \subseteq B \subseteq C \subseteq V(G)$ such that the induced graphs $G[A]$ and $G[C]$ are $(1, \leq 2)$-identifiable, but $G[B]$ is not. See Figure 3.2 for an example.



(a) (b)

Figure 3.2: An example of a $(1, \leq 2)$-identifiable graph. Even though the set of blue vertices in 3.2a is a subset of the blue vertices in 3.2b, the induced graph on the former is $(1, \leq 2)$-identifiable whereas the induced graph on the latter is not.

Corollary 5. *Any cycle $C_{n\geq 7}$ has a $(1, \leq 2)$-identification number of n.*

Proof. Removing any vertex from the $(1, \leq 2)$-identifying code C of a graph cycle C_n makes the subgraph induced by C a tree. By Theorem 4, we are done. □

Since Theorem 4 looks at induced subgraphs of $(1, \leq 2)$-identifiable graphs, one might ask what kinds of induced subgraphs exist inside of $(1, \leq 2)$-identifiable graphs. Determining if a particular graph is $(1, \leq 2)$-indistinguishable would be quicker if such graphs had forbidden subgraphs. It turns out, however, that there are none.

Proposition 6. *Every graph is an induced subgraph of a $(1, \leq 2)$-identifiable graph.*

Proof. Given any graph G, for each vertex $v_0 \in V(G)$, create a copy of P_6, denoted P_{6,v_0}, with vertices $v_1, v_2, v_3, v_4, v_5, v_6$. Add edges between v_0 and the leaves of P_{6,v_0}. This essentially makes each vertex of G part of a unique 7-cycle. Now we will show that this graph is $(1, \leq 2)$-identifiable. Observe that for any set $X = \{a, b\} \subseteq V(G)$, there are 3 distinct cases:

First, $a, b \in V(G)$, in which case $N[\{a, b\}]$ contains a, a_1, a_6, b, b_1, b_6 and no other vertices in $P_{6,a} \cup \{a\}$ or $P_{6,b} \cup \{b\}$

Second, $a \in V(G), b \in P_{6,v}$ for some $v \in V(G)$, in which case $N[\{a, b\}] = N[\{a, v_i\}]$ and contains $a, a_1, a_6, v_{i-1}, v_i, v_{i+1}$ (where subscripts are modulo 6) and no other vertices in $P_{6,a} \cup \{a\}$ or $P_{6,b} \cup \{b\}$. The same argument applies when $b \in V(G), a \in P_{6,v}$ for some $v \in V(G)$.

Third, $a = v_i \in P_{6,v}$ and $b = w_j \in P_{6,w}$ for some $v, w \in V(G)$. Then $N[\{a, b\}]$ contains $v_{i-1}, v_i, v_{i+1}, w_{j-1}, w_j$, and w_{j+1} (where subscripts are modulo 6) and no other vertices in $P_{6,a} \cup \{a\}$ or $P_{6,b} \cup \{b\}$. Even if $v = w$, these sets of vertices are separated because the 7-cycle is $(1, \leq 2)$-identifiable.

Since each of these cases have a unique intersection with the $P_{6,v}$'s, this implies that the constructed graph is identifiable. □

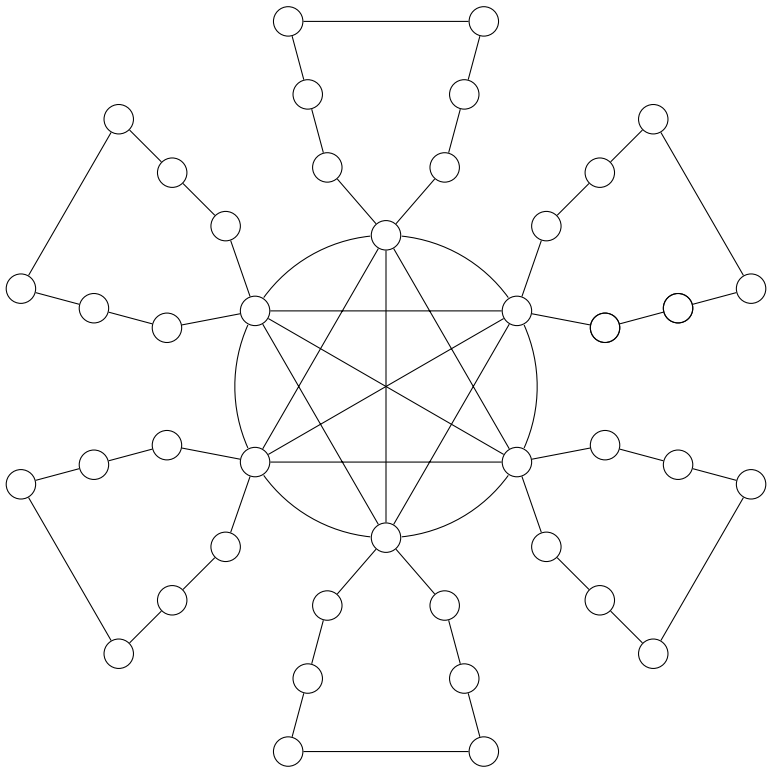


Figure 3.3: A $(1, \leq 2)$-identifiable graph containing K_6 as an induced subgraph.

Since the above result holds for any graph, including K_n, this shows that the maximum degree Δ and the clique number ω are both may be arbitrarily large. However, the maximum degree cannot be arbitarily large when compared to the order n of the graph.

Proposition 7. *If the order of a $(1, \leq 2)$-identifiable graph G is n and the maximum degree of G is Δ, then $\Delta + 1 + \log_2(\Delta) \leq n$,*

Proof. Let $v \in V(G)$ have degree Δ. Then by Proposition 1, for each vertex $u \in N[v]$, $N[u] \nsubseteq N[v]$. That is, each u must be adjacent to at least one vertex w which is not in $N[v]$. Yet in order for G to be $(1, \leq 2)$-identifiable, each u must be adjacent to a distinct subset of $V(G) - N[v]$. Let S be such a subset with $N[u_i] \cap S \neq N[u_j] \cap S$ for $u_i \neq u_j$ where $u_i, u_j \in N(v)$. Since $|N(v)| = \Delta$, we must have $2^{|S|} \geq \Delta$ and thus $|S| \geq \log_2(\Delta)$. Furthermore, since $|N[v]| = \Delta + 1$, it follows that $\Delta + 1 + |S| \leq n$, so $\Delta + 1 + \log_2(\Delta) \leq n$. □

It is likely that the bound in Proposition 7 can be improved by finding a formula for calculating a number c such that $\Delta + 1 + c \leq n$ for all $(1, \leq 2)$-identifiable graphs. From the simple counting argument in the proof of Proposition 7, this c must be greater than or equal to $\log_2(\Delta)$, but there is a limit of how high the lower bound may go; such a c cannot exceed 2Δ.

Proposition 8. *There exist infinitely many $(1, \leq 2)$-identifiable graphs with $3\Delta + 1 \leq n$.*

Proof. Fix $\Delta \geq 3$. Create one vertex v of degree Δ, where no two vertices adjacent to v are adjacent to each other. Denote the vertices adjacent to v by u_i (i from 1 to Δ). Create a path of length Δ consisting of cannonically numbered vertices w_i and connect each w_i to the corresponding u_i. Repeat this by creating another path of length Δ consisting of canonically numbered x_i's and connecting each x_i to the corresponding u_i. See Figure 3.4 for an example with $\Delta = 4$.

The claim is that this graph is $(1, \leq 2)$-identifiable. This is a simple matter of case checking. We will do so in general, with the convention that a subscript for a specific case

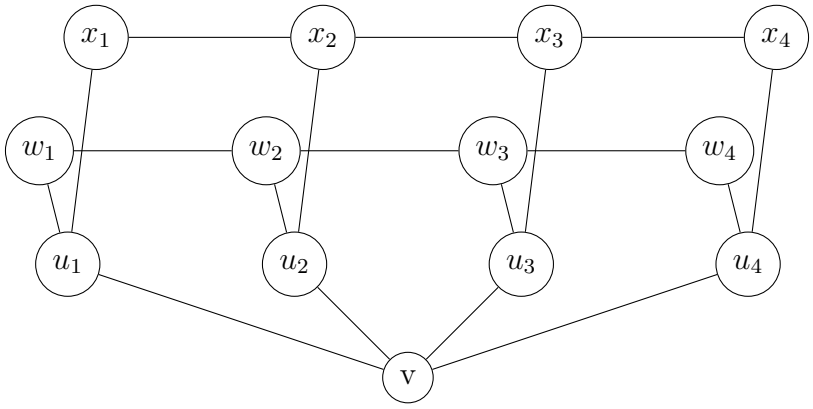


Figure 3.4: An example of a graph satisfying $3\Delta + 1 = n$

is 0 or $\Delta + 1$ implies that the "vertex" in question should be ignored.

$N[v] = \{v\} \cup \{u_k | k = 1, ..., \Delta\}$

$N[u_i] = \{u_i, w_i, x_i, v\}$

$N[w_i] = \{w_{i-1}, w_i, w_{i+1}, u_i\}$

$N[x_i] = \{x_{i-1}, x_i, x_{i+1}, u\}$

$N[v] \cup N[u_i] = \{w_i, x_i, v\} \cup \{u_k | k = 1, ..., \Delta\}$

$N[v] \cup N[w_i] = \{w_{i-1}, w_i, w_{i+1}, v\} \cup \{u_k | k = 1, ..., \Delta\}$

$N[v] \cup N[x_i] = \{x_{i-1}, x_i, x_{i+1}, v\} \cup \{u_k | k = 1, ..., \Delta\}$

$N[u_i] \cup N[u_j] = \{v, u_i, u_j, w_i, w_j, x_i, x_j\}$

$N[u_i] \cup N[w_j] = \{v, u_i, u_j, w_{j-1}, w_j, w_{j+1}, x_i\}$

$N[u_i] \cup N[x_j] = \{v, u_i, u_j, w_i, x_{j-1}, x_j, x_{j+1}\}$

$N[w_i] \cup N[w_j] = \{u_i, u_j, w_{i-1}, w_i, w_{i+1}, w_{j-1}, w_j, w_{j+1}\}$

$N[w_i] \cup N[x_j] = \{u_i, u_j, w_{i-1}, w_i, w_{i+1}, x_{j-1}, x_j, x_{j+1}\}$

$N[x_i] \cup N[x_j] = \{u_i, u_j, x_{i-1}, x_i, x_{i+1}, x_{j-1}, x_j, x_{j+1}\}$

To show that no two of these intersections of neighborhoods would be identical, we observe the following distinctions. Let S be of the form $N[i] \cup N[j]$. Given such an S, we will show that we may uniquely determine the vertices i and j. To do so, please consult the flow chart in Figure 3.5.

Since each closed neighborhood above is distinct from any other, this shows that graphs

constructed in this way have exactly $3\Delta + 1$ many vertices. □

Corollary 9. *Given maximum degree Δ, a lower bound on the order of a $(1, \leq 2)$-identifiable graph lies between $\Delta + 1 + \log_2(\Delta)$ and $3\Delta + 1$.*

Proof. Combining Proposition 7 and Proposition 8 gives this result □

As seen in Proposition 6, there are no forbidden subgraphs of $(1, \leq 2)$-identifiable graphs. This was shown by taking an arbitrary graph and constructing a $(1, \leq 2)$-identifiable graph containing it. However, it should be noted that there is more than one way to construct such a graph. This variety of structures can be useful in computations, since different classes of graphs can be easier to program into a computer. As it turns out, we can observe any connected graph as an induced subgraph of a cycle with chords. In addition to having a structure which is easier to work with, the construction method presented below requires far fewer vertices than that in Proposition 6. The number of computations needed to exhaustively check if a particular set is a valid $(1, \leq 2)$-identifying code grows quickly with the order of the graph, hence keeping graphs as small as possible is quite desirable.

Proposition 10. *For any connected graph G of order $n > 2$, $(1, \leq 2)$-identifiable cycle H with $|E(G)|$ many chords such that G is an induced subgraph of H with $\gamma^{ID}_{1,\leq 2}(H) \geq 4n$.*

Proof. Let G be a connected graph with order n. Choose an arbitrary ordering on the vertices: $v_0, v_1, ..., v_{n-1}$. For each vertex v_i, create four new vertices a_i, b_i, c_i and d_i. Introduce edges (v_{i-1}, a_i), (a_i, b_i), (b_i, c_i), (c_i, d_i) and (d_i, v_i) where indexes are modulo n. The resulting graph will be denoted as H and we will refer to $\{a_i, b_i, c_i, d_i | i = 1, ..., n-1\}$ "additional vertices". Observe that H is a $5n$-cycle with $|E(G)|$ many chords. Note that since G was connected with order greater than two, each v_i has degree ≥ 3 in H; it is adjacent to d_i and a_{i+1}. Also, the additional vertices a_i, b_i, c_i and d_i are all of degree 2, with the b_i's and c_i's being the only vertices adjacent to nothing but vertices of degree 2. Thus even with out labels, it is easy to identify which are the additional vertices. We need only show that H is $(1, \leq 2)$-identifiable. Clearly from construction, the closed neighborhood of any one

vertex is distinct from that of any other since it contains at least 2 but no more than 3 additional vertices. So consider $N[X]$ where $|X| = 2$. There are 15 simple cases to check: If $X = \{v_i, v_j\}$, then $N[X]$ contains $d_i, v_i, a_{i+1}, d_j, v_j, a_{j+1}$, along with at least one other v, but no other additional vertices.

If $X = \{a_i, v_j\}$ then $N[X]$ contains $v_{i-1}, a_i, b_i, d_j, v_j, a_{j+1}$, along with at least one other v, but no other additional vertices.

If $X = \{b_i, v_j\}$ then $N[X]$ contains $a_i, b_i, c_i, d_j, v_j, a_{j+1}$, along with at least one other v, but no other additional vertices.

If $X = \{c_i, v_j\}$ then $N[X]$ contains $b_i, c_i, d_i, d_j, v_j, a_{j+1}$, along with at least one other v, but no other additional vertices.

If $X = \{d_i, v_j\}$ then $N[X]$ contains $c_i, d_i, v_{i+1}, d_j, v_j, a_{j+1}$, along with at least one other v, but no other additional vertices.

If $X = \{a_i, a_j\}$, then $N[X]$ contains $v_{i-1}, a_i, b_i, v_{j-1}, a_j, b_j$ but no other vertices

If $X = \{a_i, b_j\}$ then $N[X]$ contains $v_{i-1}, a_i, b_i, a_j, b_j, c_j$ but no other vertices

If $X = \{a_i, c_j\}$ then $N[X]$ contains $v_{i-1}, a_i, b_i, b_j, c_j, d_j$, but no other vertices

If $X = \{a_i, d_j\}$ then $N[X]$ contains $v_{i-1}, a_i, b_i, c_j, d_j, v_j$, but no other vertices

If $X = \{b_i, b_j\}$, then $N[X]$ contains $a_i, b_i, c_i, a_j, b_j, c_{j+1}$, but no other vertices

If $X = \{b_i, c_j\}$ then $N[X]$ contains $a_i, b_i, c_i, b_j, c_j, d_j$, but no other vertices

If $X = \{b_i, d_j\}$ then $N[X]$ contains $a_i, b_i, c_i, c_j, d_j, v_j$, but no other vertices

If $X = \{c_i, c_j\}$, then $N[X]$ contains $b_i, c_i, d_i, b_j, c_j, d_j$, but no other vertices

If $X = \{c_i, d_j\}$ then $N[X]$ contains $b_i, c_i, d_i, c_j, d_j, v_j$, but no other vertices

If $X = \{d_i, d_j\}$, then $N[X]$ contains $c_i, d_i, v_i, c_j, d_j, v_j$, but no other vertices.

The combination of additional vertices in any $N[X]$ will indicate exactly which vertices are in X. To confirm this statement, let S be of the form $N[i] \cup N[j]$. Given such an S, we may uniquely determine the vertices i and j by consulting the flow chart in Figure 3.6. Therefore, H is $(1, \leq 2)$-identifiable.

Furthermore, any $(1, \leq 2)$-identifying code C on H must contain each additional vertex.

If this were not the case, either $H[C]$ would contain a vertex of degree 1 and contradict Theorem 4 or C would not dominate H, in contradiction with the fact that any $(1, \leq 2)$-identifying code must cover each vertex of H. Since there are $4n$ many additional vertices, each of which must be in any valid $(1, \leq 2)$-identifying code, we have that $\gamma^{ID}_{1,\leq 2}(H) \geq 4n$. $\square$

If the complement of a $(1, \leq 2)$-identifiable graph G has a Hamiltonian cycle, the same proof can be applied with the modification of only introducing three additional vertices per vertex in G as opposed to the four used in the proof of Proposition 10. However, the existence of Hamiltonian cycles in graph complements of $(1, \leq 2)$-identifiable graphs, although plausibly true, is beyond the scope of this research.

One will find an example of the construction from Proposition 10 in Figure 3.7. Observe that the construction presented in Proposition 10 must quadruple the order of a graph. However, it turns out that not all of the additional vertices must be added if the original graph G is already $(1, \leq 2)$-identifiable!

Proposition 11. *If G is a connected $(1, \leq 2)$-identifiable graph, then there exists a $(1, \leq 2)$-identifiable graph H with $|V(H)| = |V(G)| + 3$ which contains G as an induced subgraph.*

Proof. Let G be a connected $(1, \leq 2)$-identifiable graph with distinct nonadjacent vertices u and v. Define $H = (V(G) \cup \{a, b, c\}, E(G) \cup \{(u, a), (a, b), (b, c), (c, v)\})$. Clearly, G is an induced subgraph of H, so we need only show that H is $(1, \leq 2)$-identifiable. Note that for any two $X_1, X_2 \subset V(G) - \{u, v\}$, $N_G[X_1] \neq N_G[X_2]$ since G is $(1, \leq 2)$-identifiable. Thus by construction, $N_H[X_1] \neq N_H[X_2]$. Once again, we are left with case-checking around sets X which are adjacent to the additiona vertices. For this, let $w \in V(G) - \{u, v\}$.

$N_H[w] \cup N_H[u] = N_G[w] \cup N_G[u] \cup \{a\}$

$N_H[w] \cup N_H[a] = N_G[w] \cup \{u, a, b\}$

It should be noted that since G was $(1, \leq 2)$-identifiable, there does not exist a vertex z with closed neighborhood $N_G[z] = N_G[w] \cup \{u\}$ by Proposition 1.

$N_H[w] \cup N_H[b] = N_G[w] \cup \{a, b, c\}$

$N_H[w] \cup N_H[c] = N_G[w] \cup \{b, c, v\}$

$N_H[w] \cup N_H[v] = N_G[w] \cup N_G[v] \cup \{c\}$

$N_H[u] \cup N_H[a] = N_G[u] \cup \{a, b\}$

$N_H[u] \cup N_H[b] = N_G[u] \cup \{a, b, c\}$

$N_H[u] \cup N_H[c] = N_G[u] \cup \{b, c, v\}$

$N_H[u] \cup N_H[v] = N_G[u] \cup N_G[v] \cup \{a, c\}$

$N_H[a] \cup N_H[b] = \{u, a, b\}$

$N_H[a] \cup N_H[c] = \{u, a, b, c, v\}$

$N_H[a] \cup N_H[v] = N_G[v] \cup \{u, a, b, c\}$

$N_H[b] \cup N_H[c] = \{a, b, c, v\}$

$N_H[b] \cup N_H[v] = N_G[v] \cup \{a, b, c\}$

$N_H[c] \cup N_H[v] = N_G[v] \cup \{b, c\}$

To confirm that none of the above are equal, consult the flow chart in Figure 3.8, where S is of the form $N[i] \cup N[j]$ and observe that we may uniquely determing i and j. Therefore, it follows that the graph H is $(1, \leq 2)$-identifiable.

□

Proposition 11 is particularly useful when trying to find graphs which are $(1, \leq 2)$-identifiable as it may be used to create a graph with a Hamiltonian cycle for which a $(1, \leq 2)$-identifiable graph is an induced subgraph. When a graph has a Hamiltonian cycle, it can be considered a cycle with chords, which some may find computationally easier to work with. Indeed, a brute-force program was written in Maple to exhaustively check for $(1, \leq 2)$-identifiable graphs inside of cycles with chords. In the appendix, one will find a list of all cycles on up to 11 vertices with less than or equal to a particular number of chords which are $(1, \leq 2)$-identifiable, along with a copy of the Maple code used to find them.

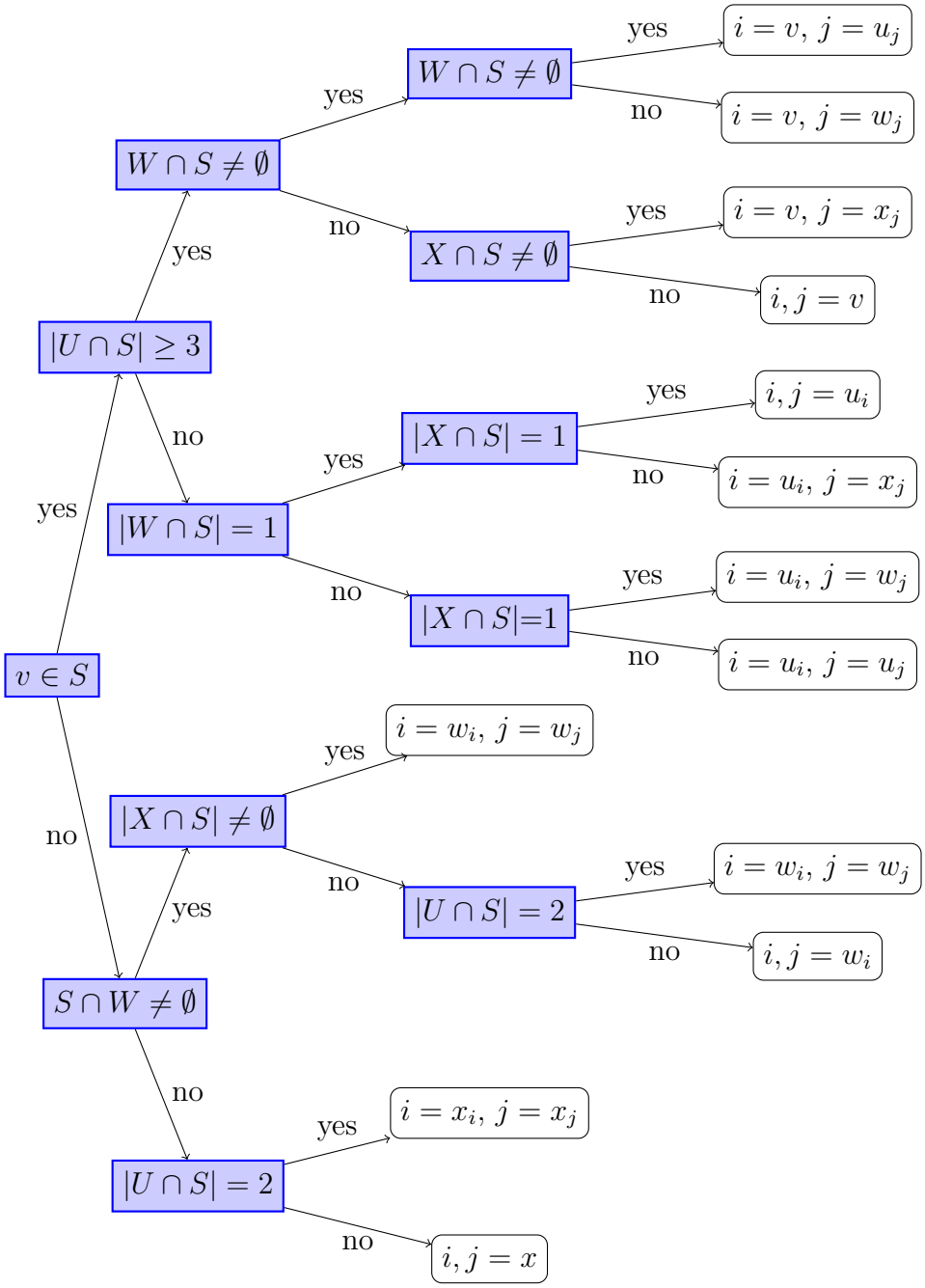


Figure 3.5: A flow chart for determining i and j given $S = N[i] \cup N[j]$ in the construction from the proof of Proposition 8 where $U = \{u_k | k = 1, ..., \Delta\}$, $W = \{w_k | k = 1, ..., \Delta\}$, and $X = \{x_k | k = 1, ..., \Delta\}$.

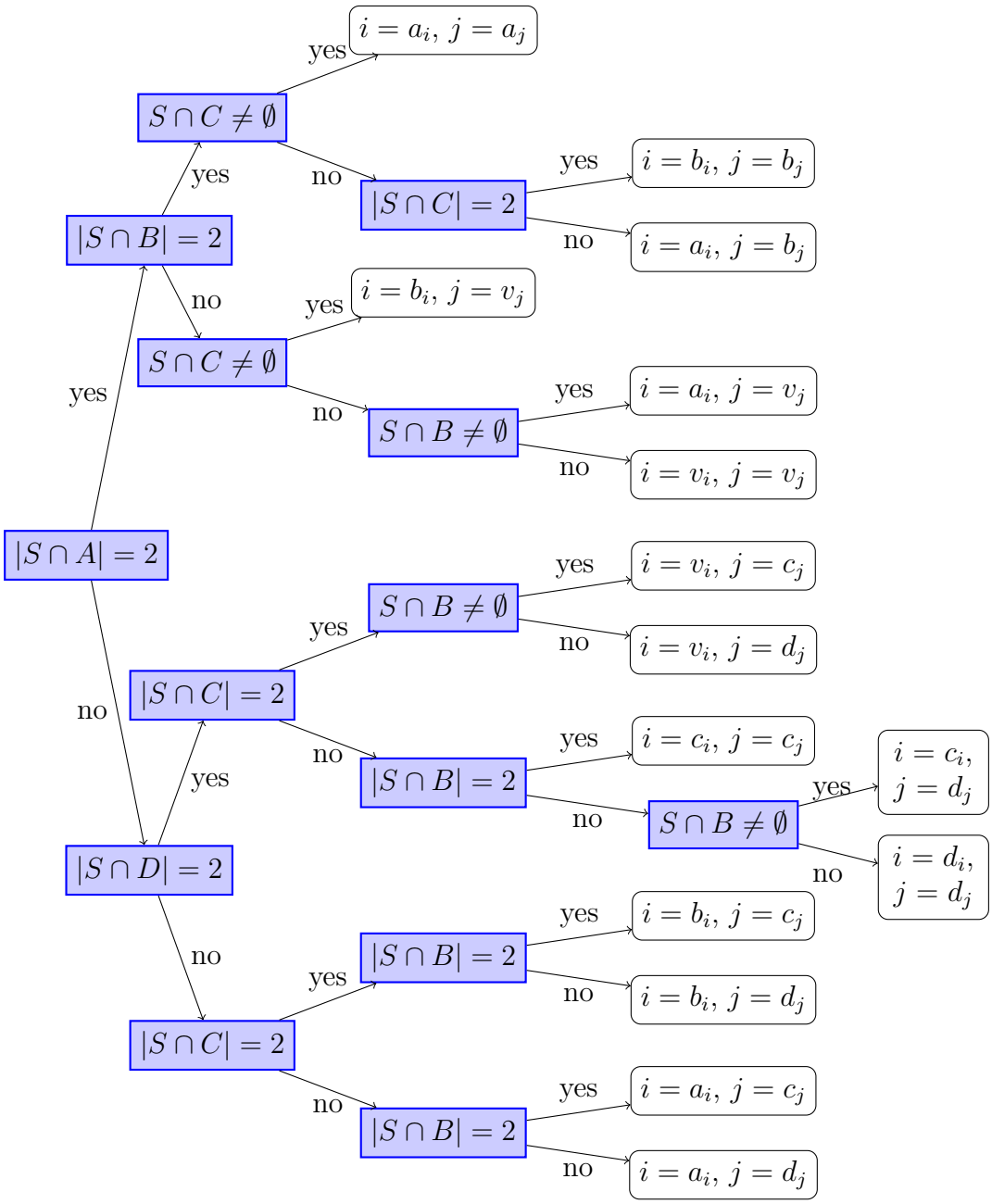
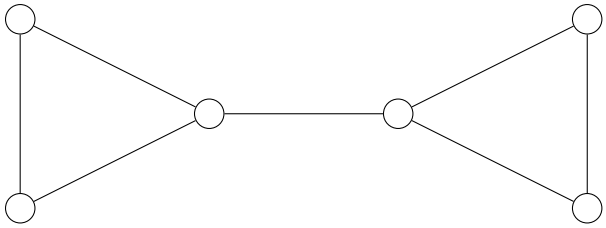
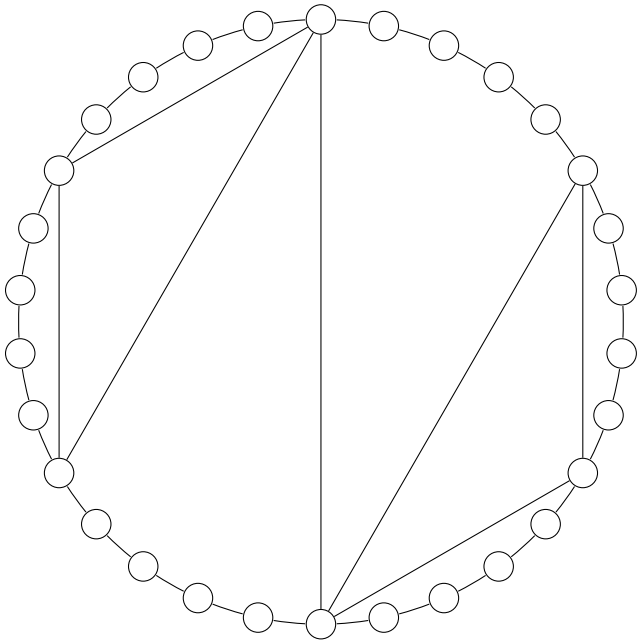


Figure 3.6: A flow chart for how to determine i and j given $S = N[i] \cup N[j]$ in the construction from the proof of Proposition 10



(a) A graph which does not contain a Hamiltonian cycle.



(b) A $(1, \leq 2)$-identifiable graph with a Hamiltonian cycle containing Figure 3.7a as an induced subgraph.

Figure 3.7: An example for Proposition 10

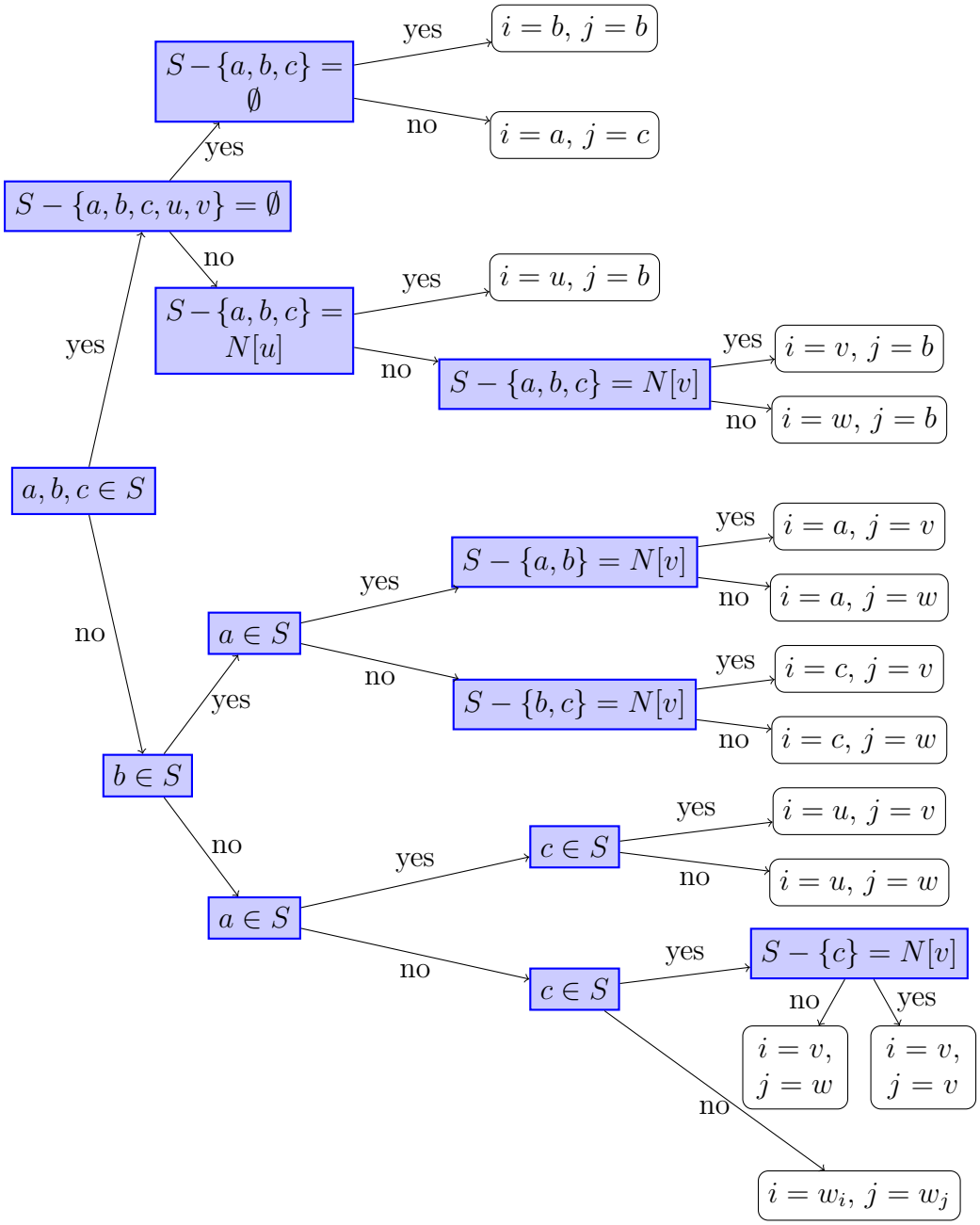


Figure 3.8: A flow chart for how to determine i and j given $S = N[i] \cup N[j]$ in the construction from the proof of Proposition 11

Chapter 4

Conclusion

We have discussed relationships between the existence of $(1, \leq 2)$-identifying codes and the structure of a graph. Many of these are most useful in determining whether such a code exists and if so, determining if a particular set will constitute a valid $(1, \leq 2)$-identifying code. Additionally, a connection between the maximum degree of a graph and the order of a $(1, \leq 2)$-identifiable graph was presented, which can also be helpful in determining whether a particular graph is $(1, \leq 2)$-indistinguishable or not. It should be noted that none of these methods can be used to prove that a $(1, \leq 2)$-identifying code exists, but they are helpful in quickly determining if such a code does not. Further, we explored different methods of constructing $(1, \leq 2)$-identifiable graphs which contain particular induced subgraphs. This is not particularly helpful in understanding $(1, \leq 2)$-identifying codes in and of themselves, but it is incredibly useful in computation.

Clearly, there is much more yet to be discovered in the exciting field of $(1, \leq 2)$-identifying codes. In an effort to help facilitate further investigation into the subject, several $(1, \leq 2)$-identifiable graphs were found and minimum $(1, \leq 2)$-identifying codes of these were indicated. Since verifying that a particular graph is $(1, \leq 2)$-identifiable is so computationally intensive, it is expected that these examples will be helpful in inspiring new theorems in the future.

Bibliography

[1] T. W. Haynes, D. J. Knisley, E. Seier, and Y. Zou. A quantitative analysis of secondary rna structure using domination based parameters on trees. *BMC Bioinformatics*, 7: 108–138, 2006.

[2] S. Ray, R. Ungrangsi, F. De Pellegrini, A. Trachtenberg, and D. Starobinski. Robust location detection in emergency sensor networks. *Proceedings of IEEE INFOCOM 2003*, 1:10441053, 2003.

[3] M. G. Karpovsky, K. Chakrabarty, and L. B. Levitin. On a new class of codes for identifying vertices in graphs. *IEEE Transactions on Information Theory*, IT-44:599611, 1998.

[4] T Laihonen and J. Moncel. On graphs admitting codes identifying sets of vertices. *Australasian Journal of Combinatorics*, 41:81–91, 2008.

[5] A. Lobstein. Watching systems, identifying, locating-dominating and discriminating codes in graphs. http://www.infres.enst.fr/∼lobstein/debutBIBidetlocdom.pdf, 2014. (Accessed July 31, 2014).

[6] T. Laihonen and S. Ranto. Lecture notes in computer science, no. 2227. *Springer-Verlag*, 2227:82–91, 2001.

[7] F. Foucaud and G. Perarnau. Bounds for identifying codes in terms of degree parameters. *Electronic Journal of Combinatorics*, 19:28–32, 2006.

Appendix A

Maple code

Below is the Maple code which was used to find all graphs which are $(1, \leq 2)$-identifiable and are of the form of cycles on 10 vertices up to 4 chords. A similar code was run for the other graphs which are included in the following part of the appendix. The limit on the number of chords was imposed to fit time restrictions. Not only does this return whether a particular graph is $(1, \leq 2)$-identifiable, but it since it checks possible codes in order of increasing cardinality, it will also find a minimum $(1, \leq 2)$-identifying code. Hence we know the $(1, \leq 2)$-identification number $\gamma^{ID}_{1,\leq 2}$ for each graph. To indicate this, each vertex which need not be in a minimum $(1, \leq 2)$-identification code has been colored. Some graphs have more than one minimum $(1, \leq 2)$-identifying code, in which case the removable vertices are different colors for the different codes.

The general strategy was to begin with a cycle and choose all possible combinations of chords among the vertices. Then that graph would be tested for $(1, \leq 2)$-identifying codes. This, however, resulted in several isomorphic graphs appearing in the output. These redundancies of rotation and mirror images were drawn, labeled, and eliminated by hand to make the list more concise.

```
with(GraphTheory):
with(combinat):
```

$ord := 10$:

$mustbethisbig := ceil(\log[2](.5 * (ord * ord + ord) + 1))$:

$bidentifiable := []$:

$Vert := \{\}$:

$E := \{\}$:

for i to $ord - 1$ do

    $Vert := (Vert, \{i\})$:

    $E := E \cup \{\{i, i + 1\}\})$

end do:

$Vert := Vert \cup \{ord\}$:

$E := E \cup \{\{1, ord\}\}$:

$codewannabes := \{\}$:

for i from $mustbethisbig$ to ord do

    $codewannabes := codewannabe \cup choose(Vert, i)$

end do:

$chords := choose(Vert, 2)/E$:

$allcombos := \{\}$:

for i from 0 to 4 do

    $allcombos := allcombos \cup choose(chords, i)$

end do:

$howmanygraphs := 0$:

for $picksome$ in $allcombos$ do

    if $\{\{\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{1, 6\}\} \cap picksome \neq \{\}$ then

        $G := Graph(E \cup picksome)$:

        $howmanygraphs = howmanygraphs + 1$:

        for C in $codewannabes$ do

            $nbhd := \{\}$:

$fail := 0$:

$used := :$

for i in $Vert$ do

$Ian := convert(Neighborhood(G, i, closed), set)$:

for j from i to ord do

$Jon := convert(Neighborhood(G, j, closed), set)$:

$nbhd := (Ian \cup Jon) \cap C$:

if $(nbhd \in used)$ or $(Jon \cap C) = \{\}$ then

$fail := 1$: break: break: break:

else $used := used \cup \{nbhd\}$:

end if:

end do:

end do:

if $fail = 0$ then

$bidentifiable := [op(bidentifiable), picksome, C]$: break: break: break:

end if:

end do:

end if:

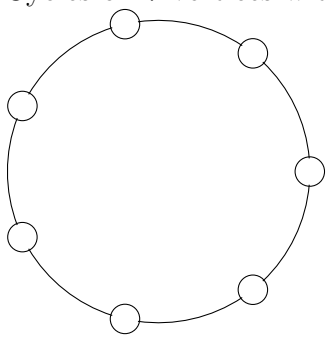
end do:

Appendix B

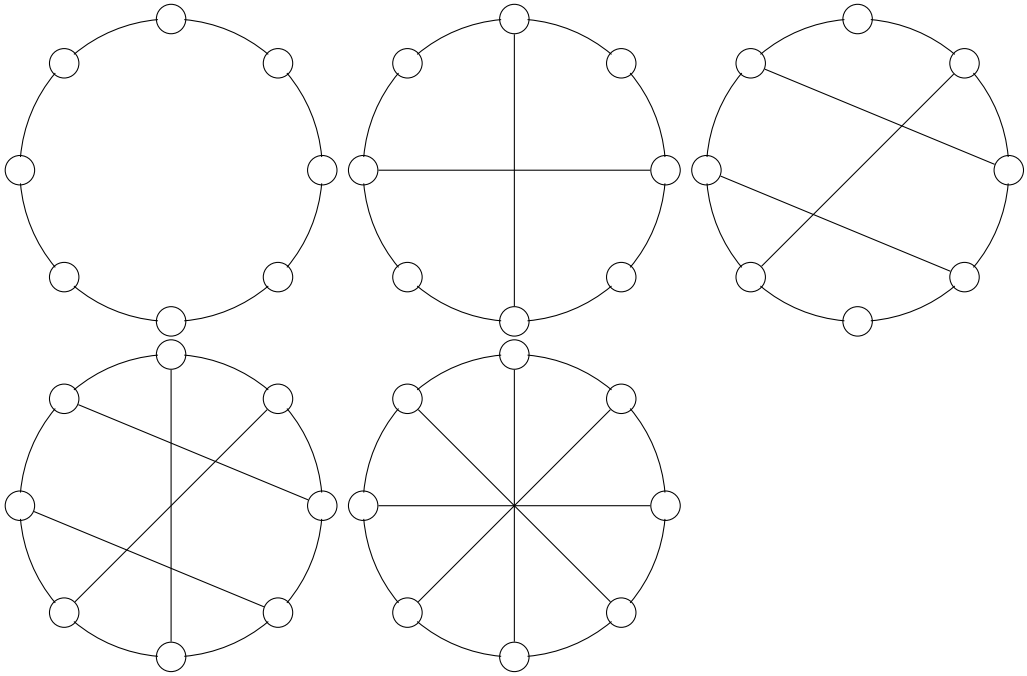
Maple Code Output

Each of the graphs below is $(1, \leq 2)$-identifiable and has a minimum $(1, \leq 2)$-identifying code indicated. If all vertices are white, then each vertex must be in a $(1, \leq 2)$-identifying code. If there is a blue vertex or a red vertex, this indicates that the set of all vertices which are not blue (or red respectively) form a $(1, \leq 2)$-identifying code.

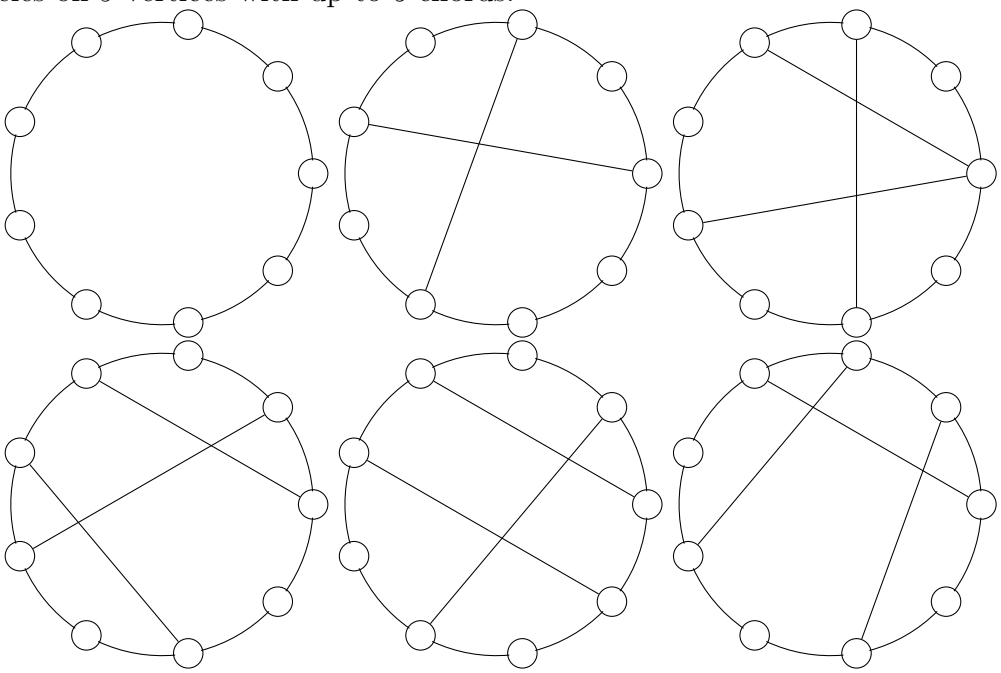
Cycles on 7 vertices with any number of chords:

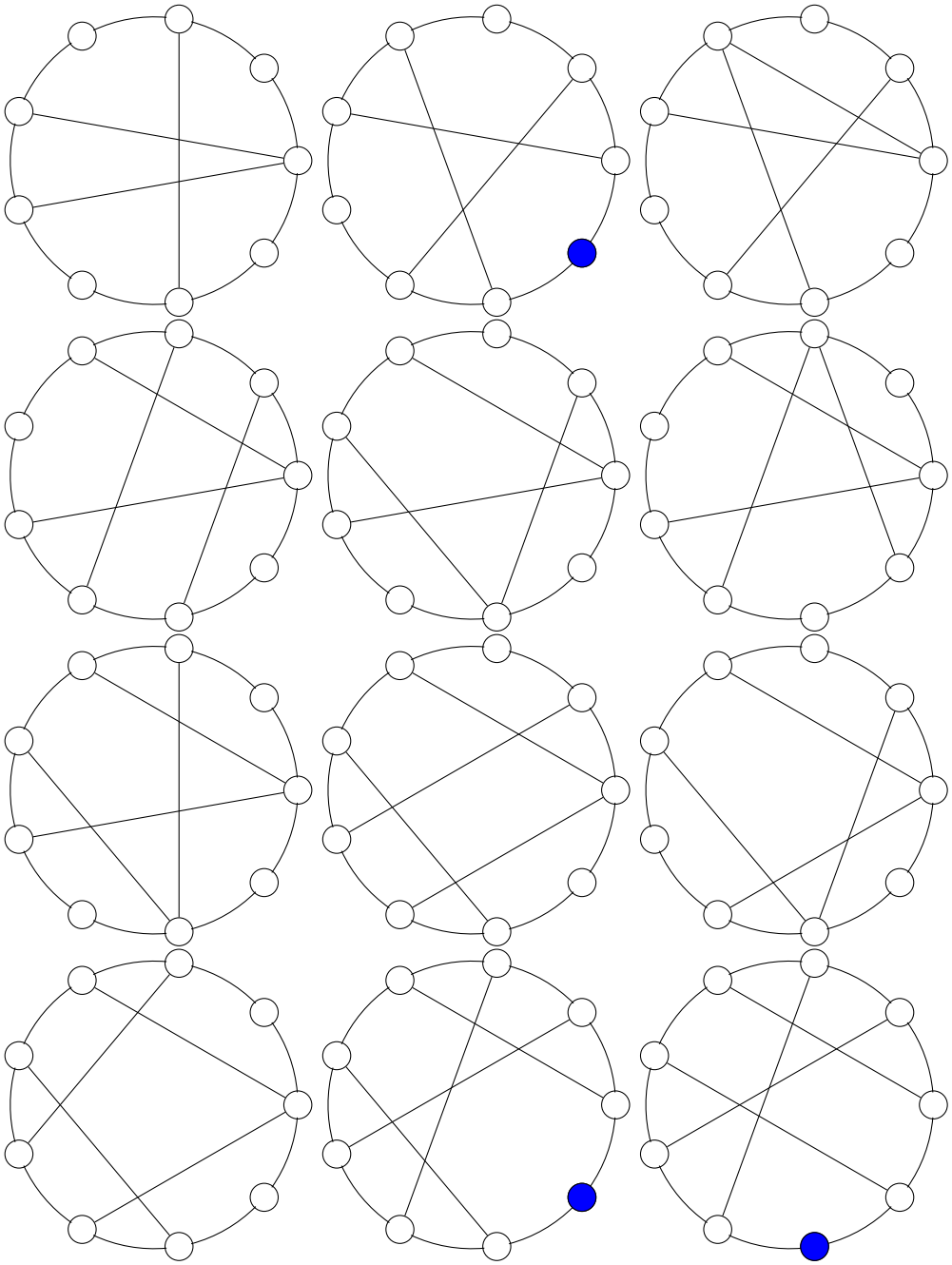


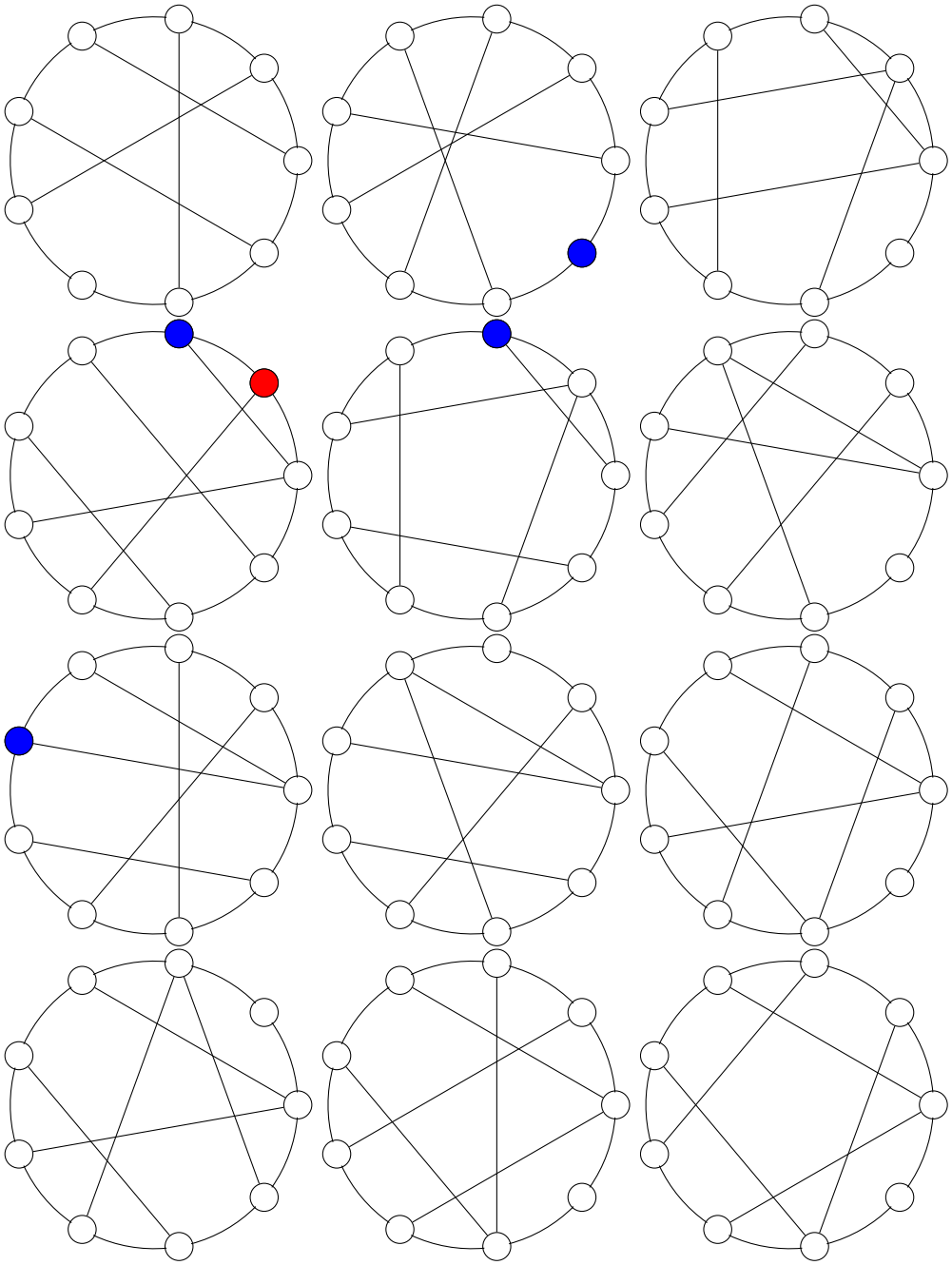
Cycles on 8 vertices with any number of chords:

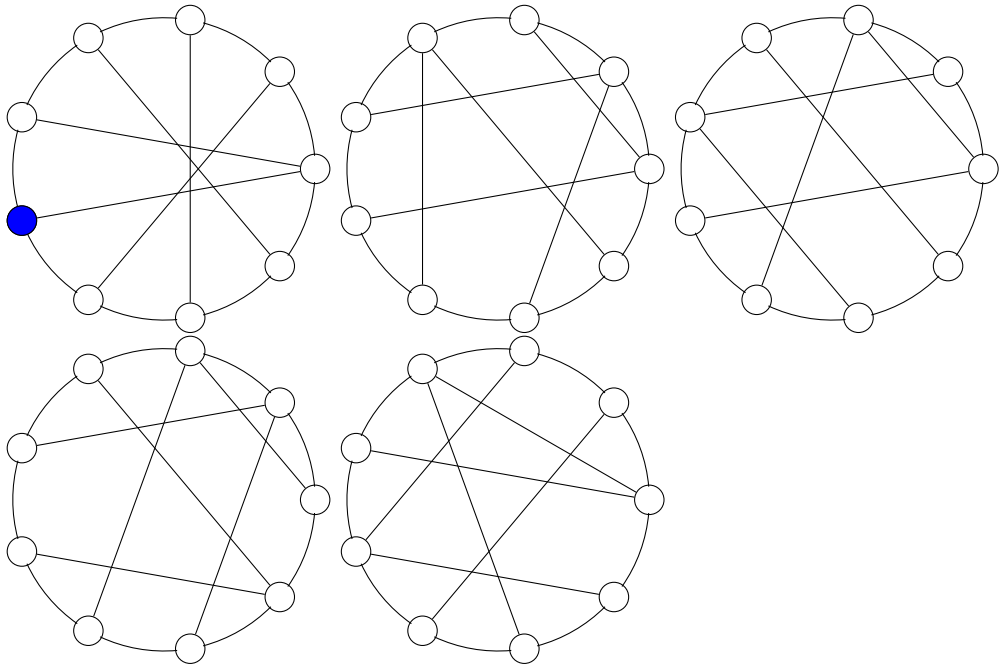


Cycles on 9 vertices with up to 5 chords:

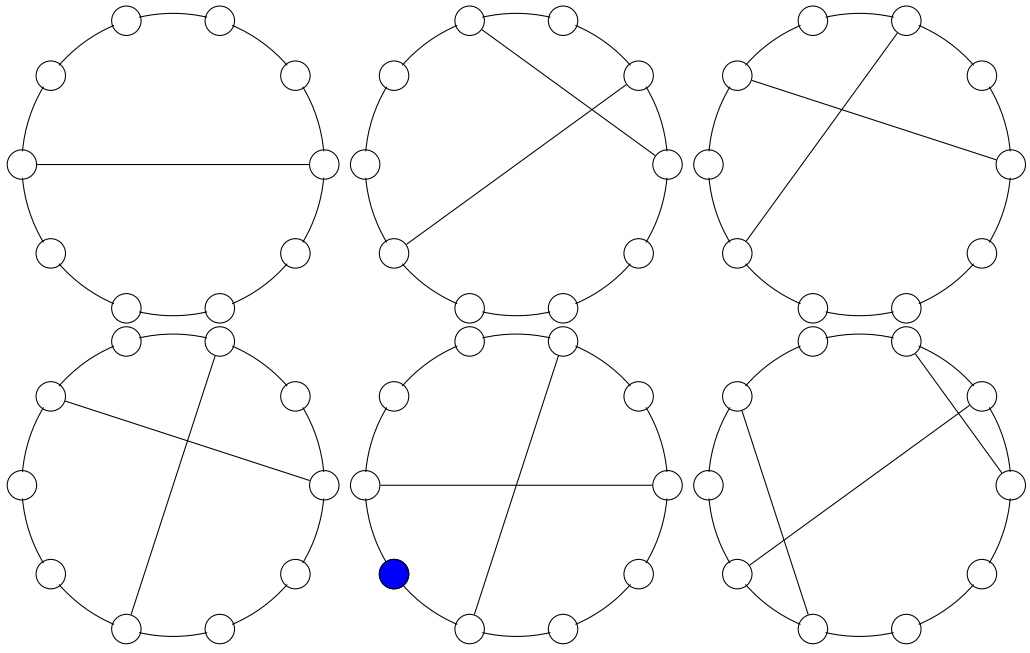


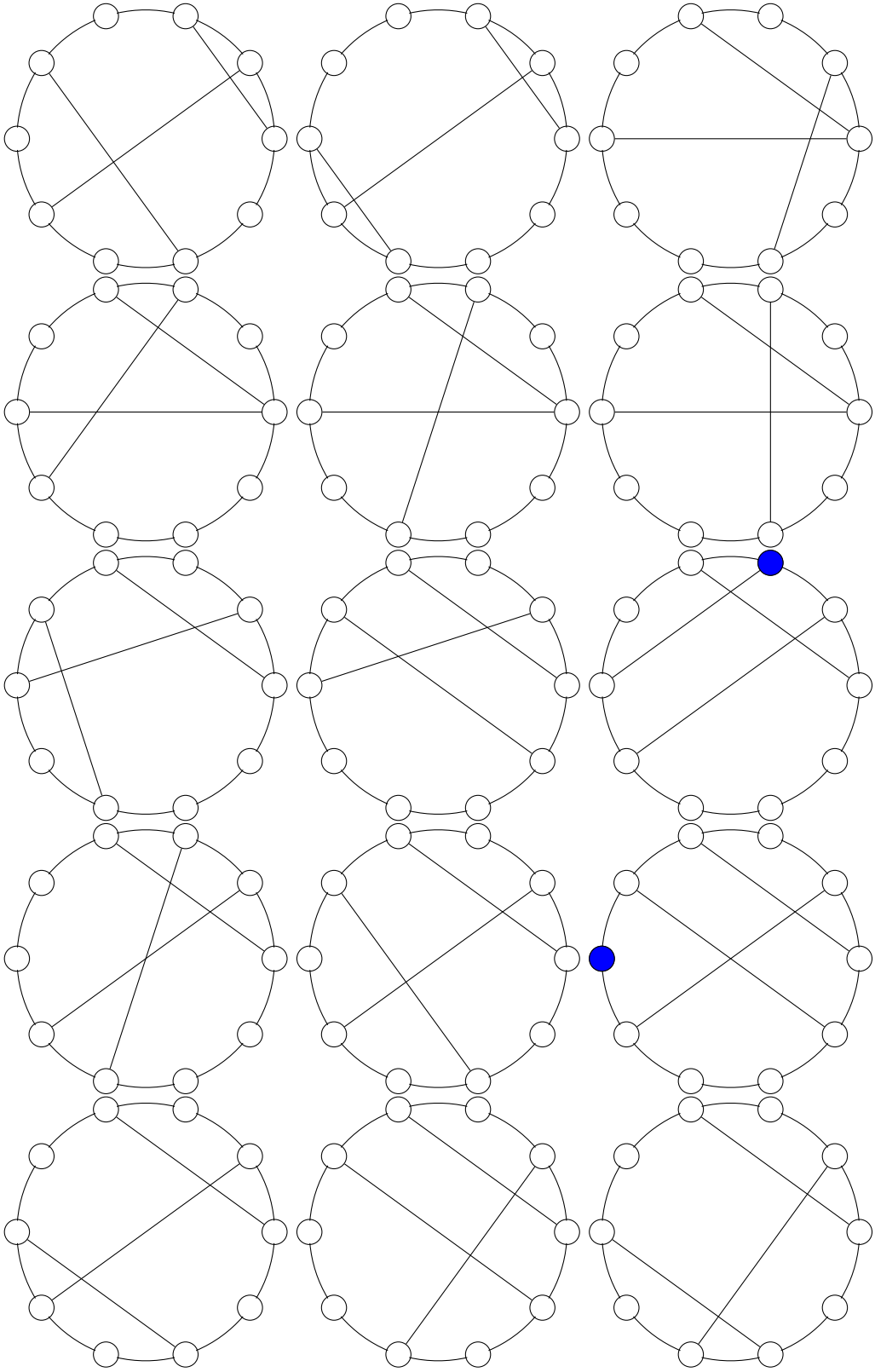


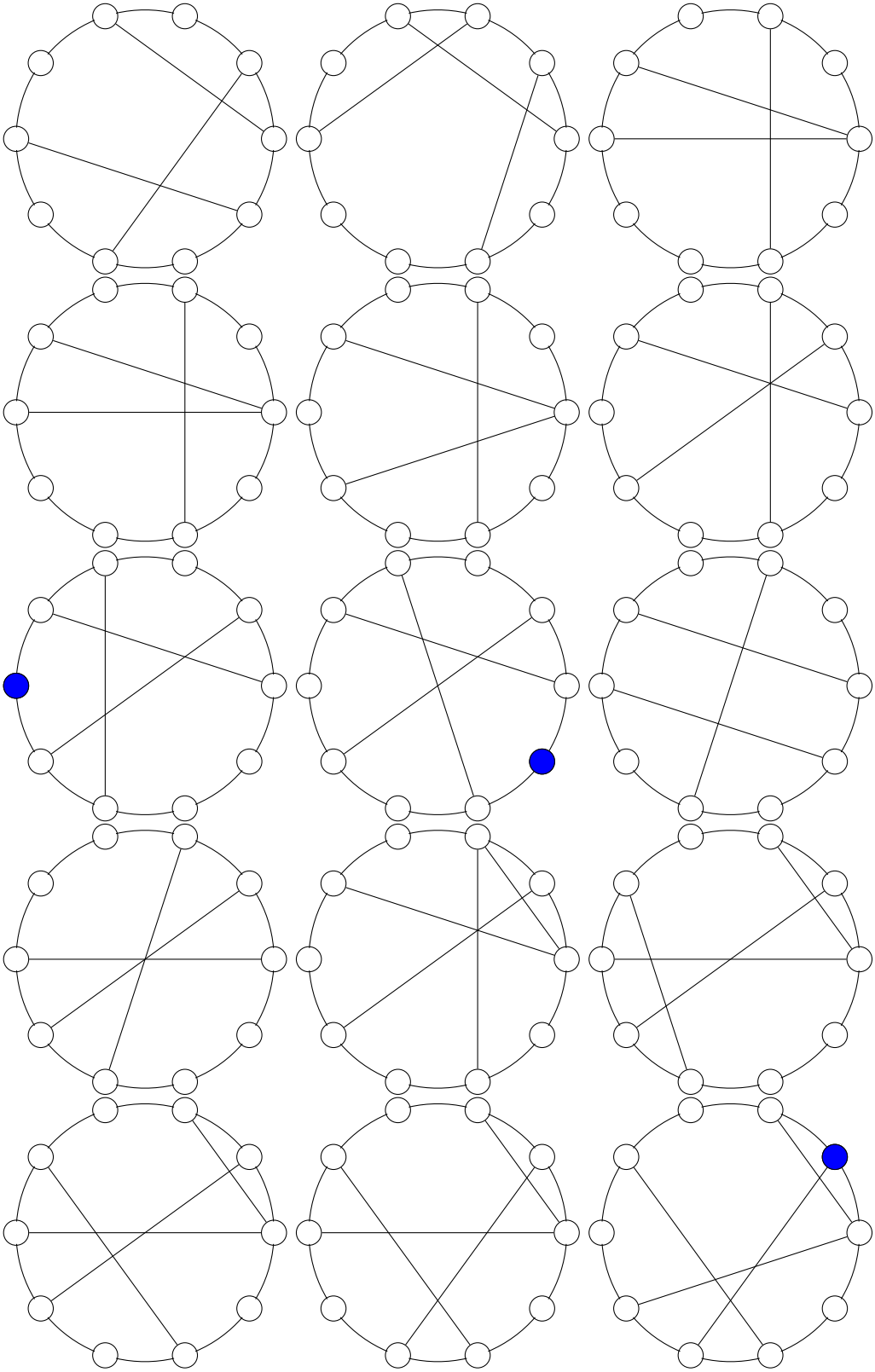


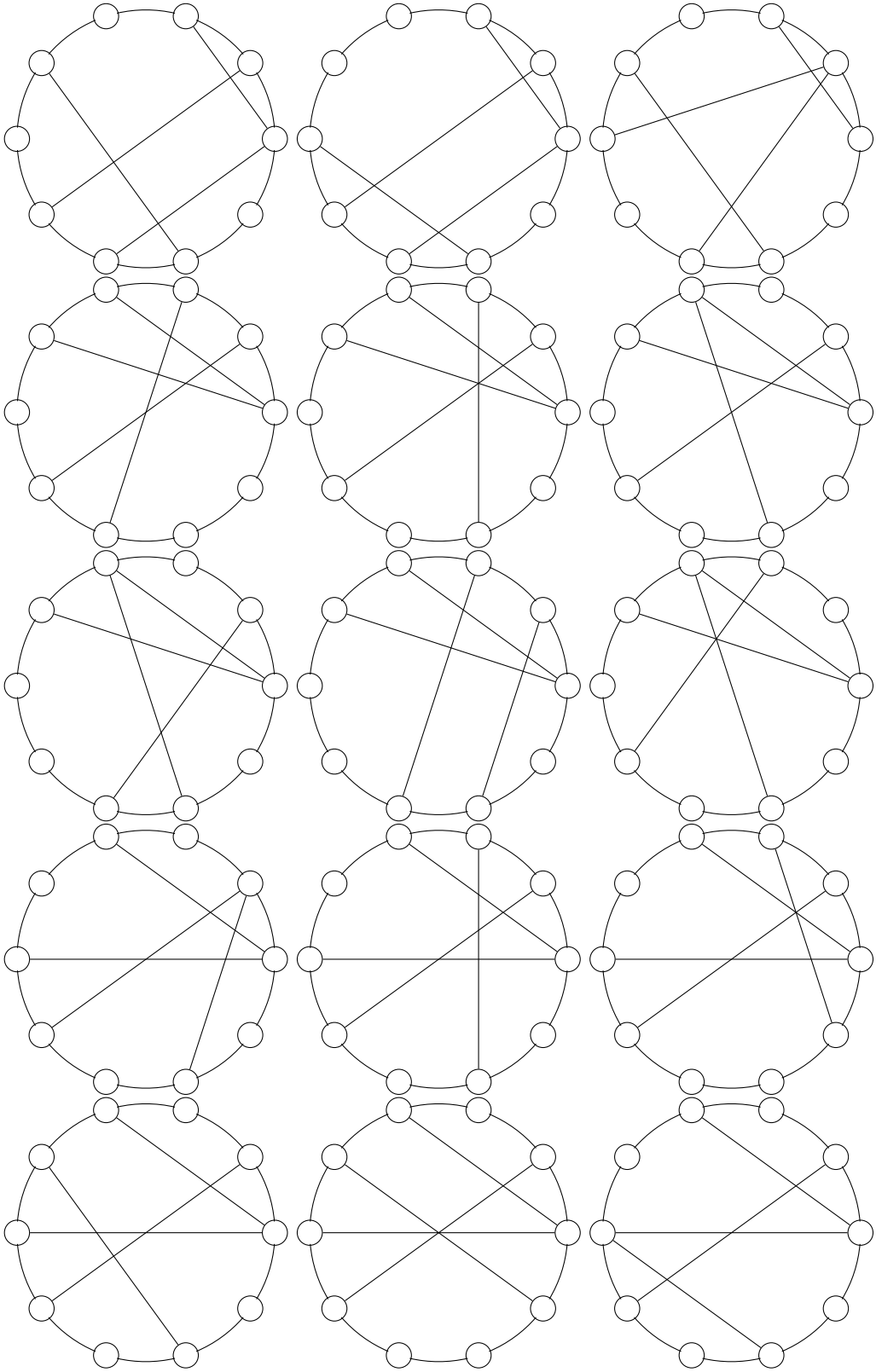


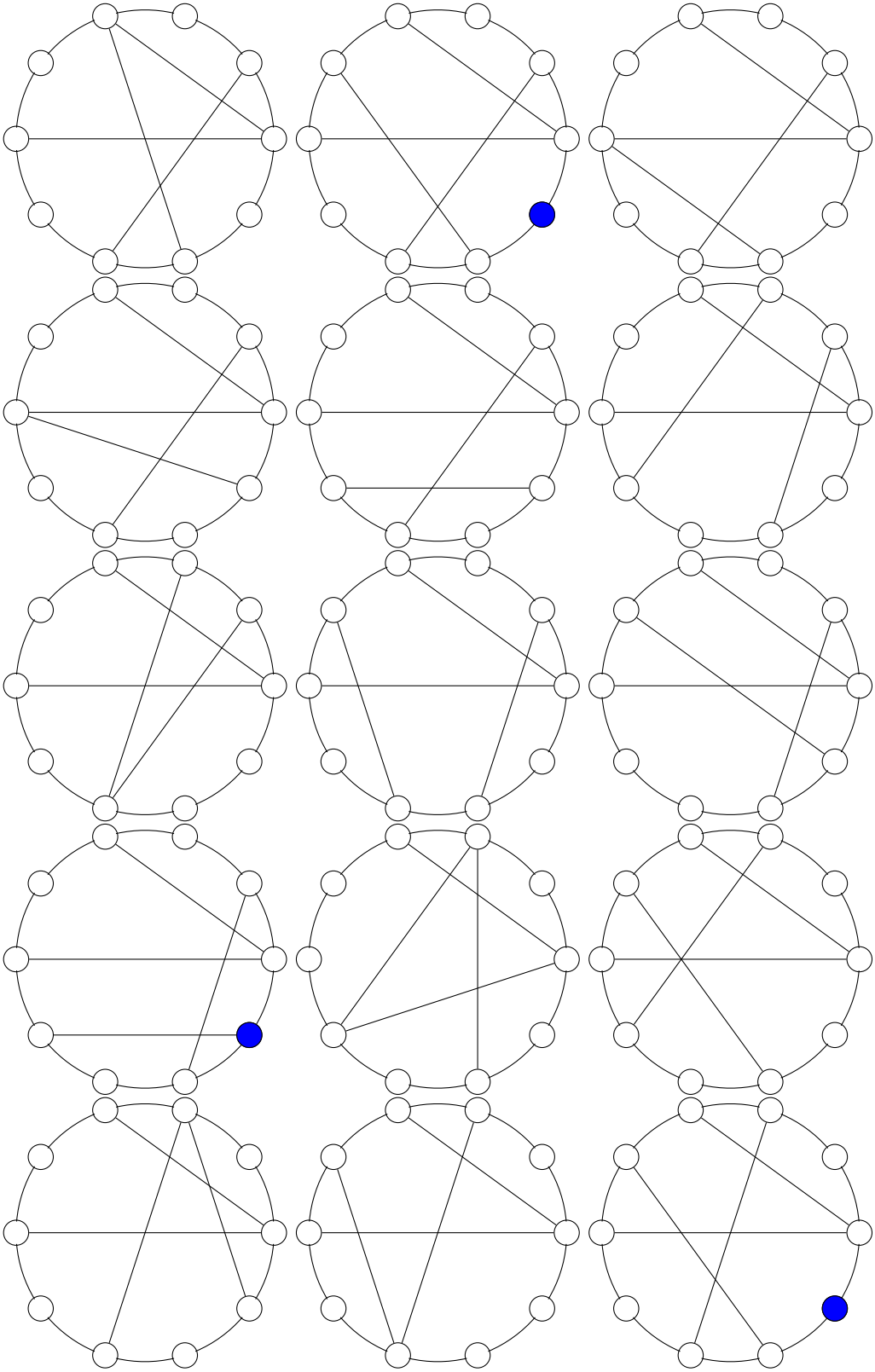
Cycles on 10 vertices with up to 4 chords

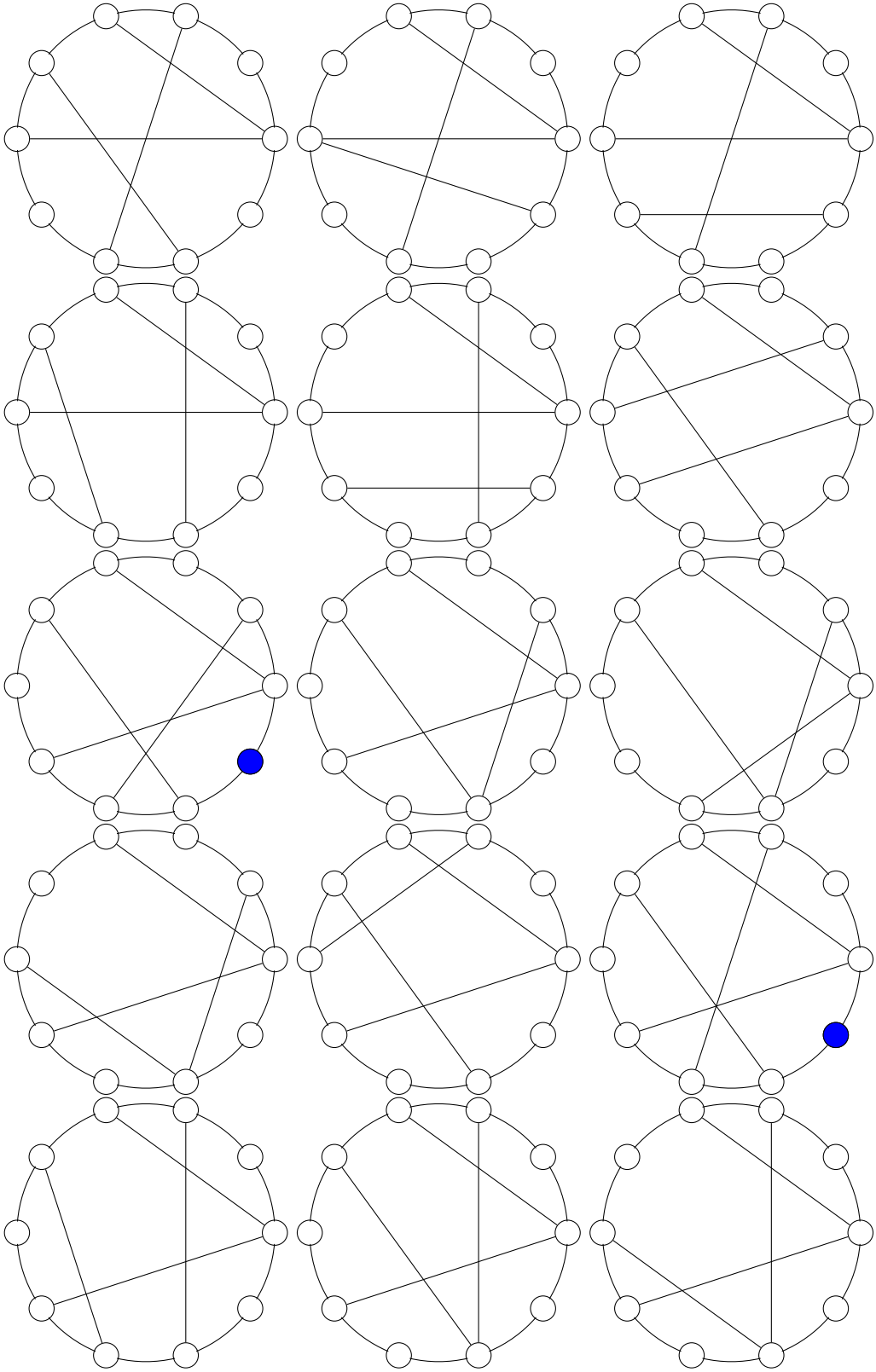


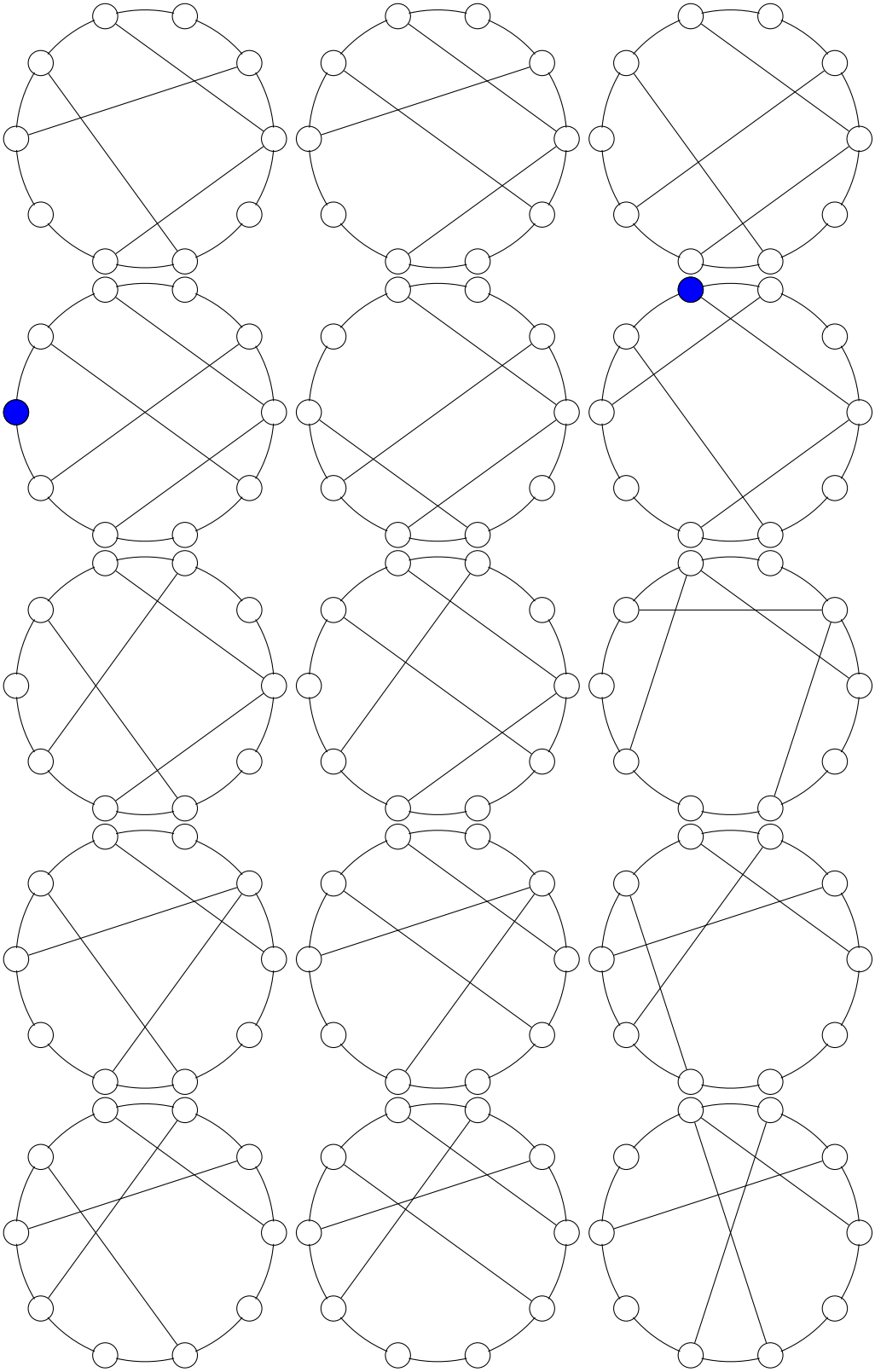


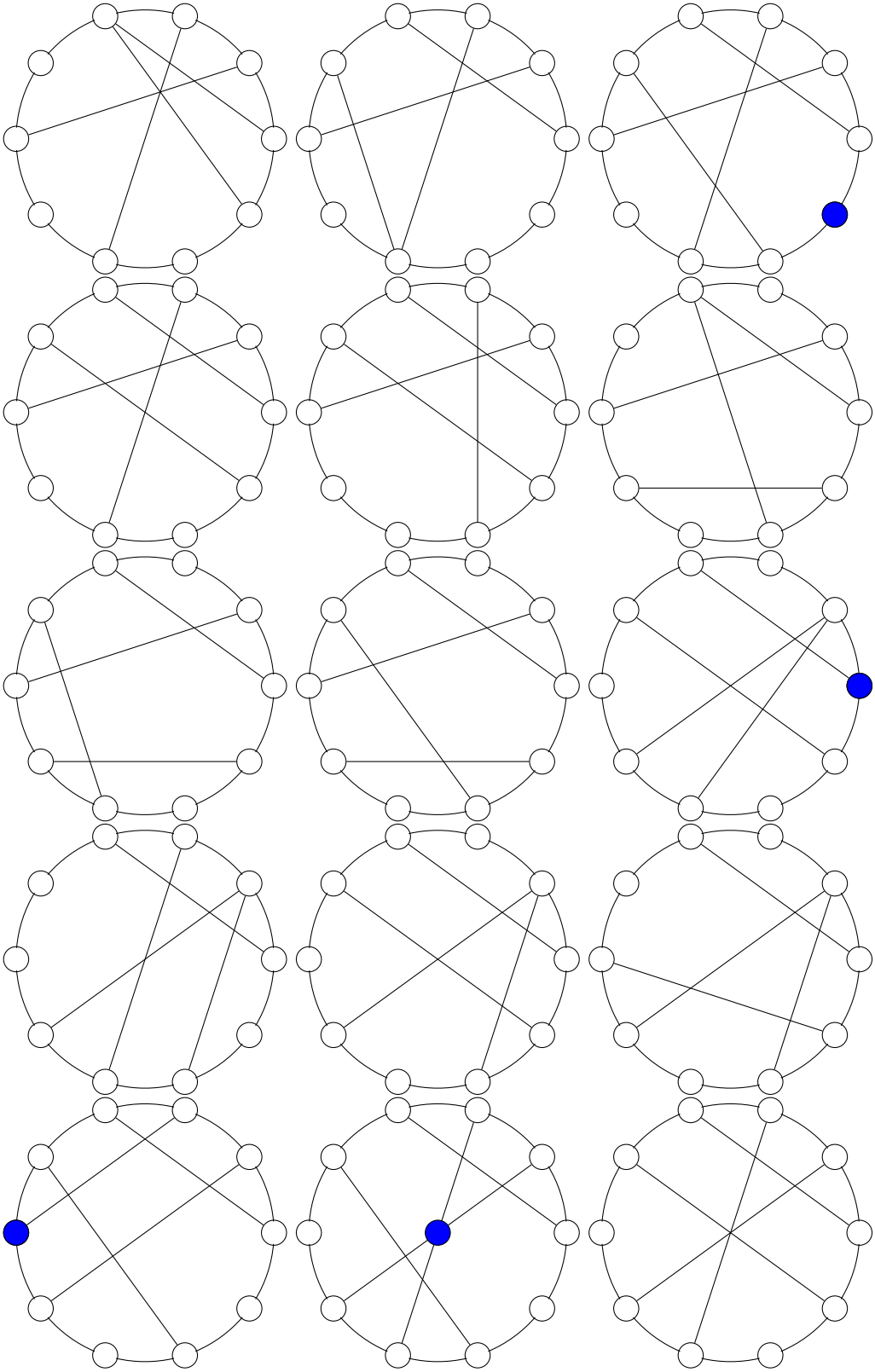


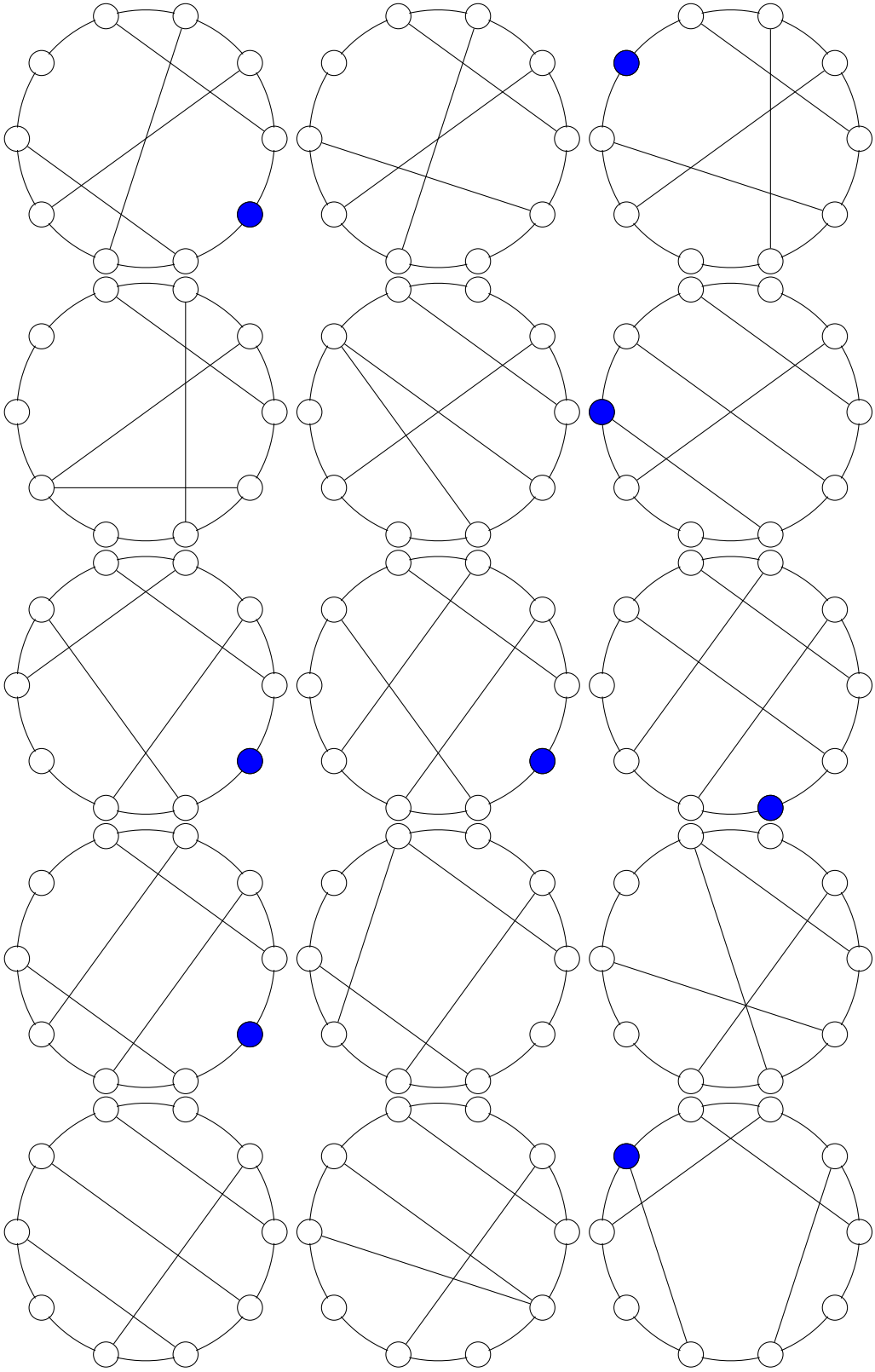


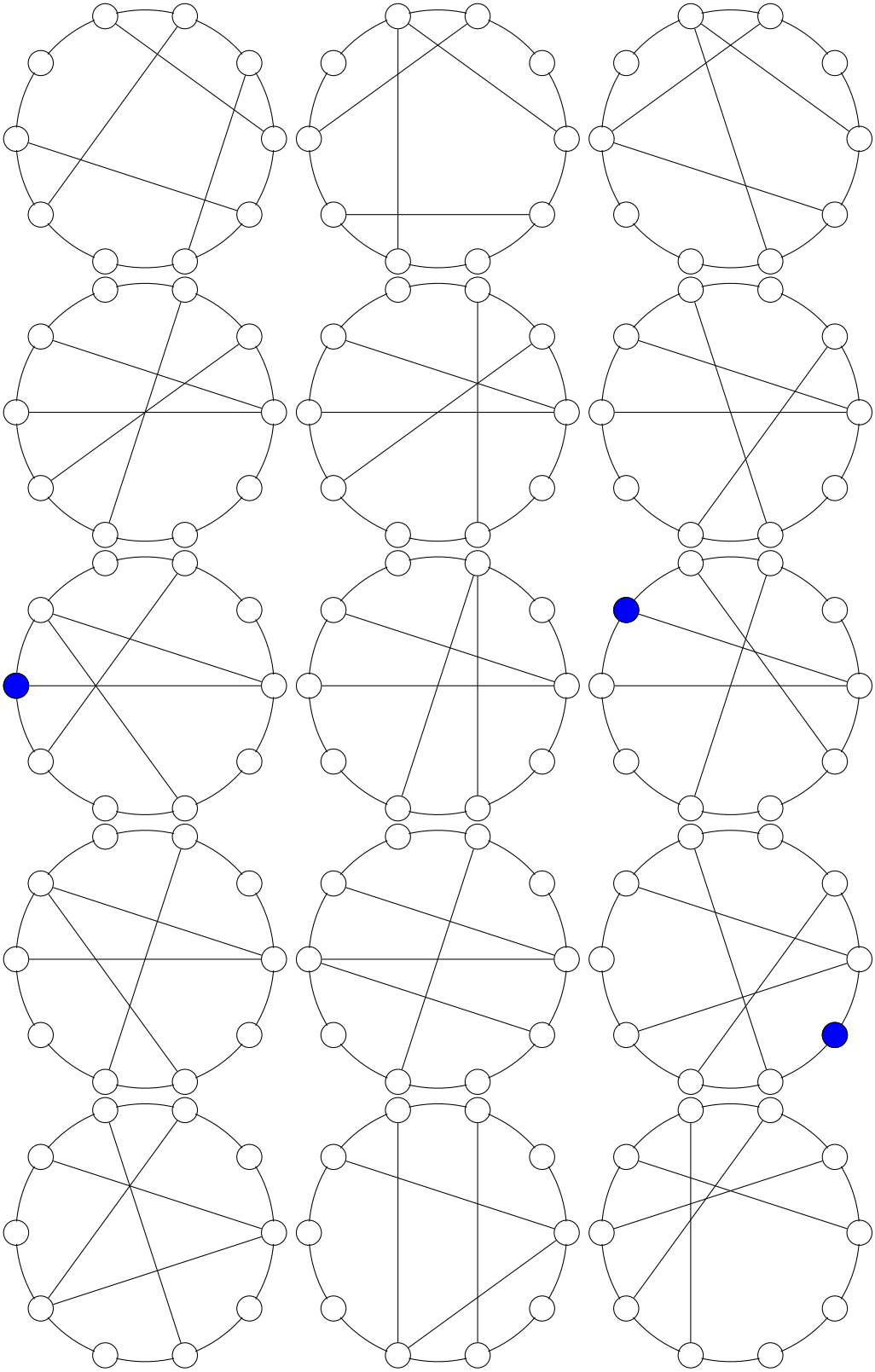


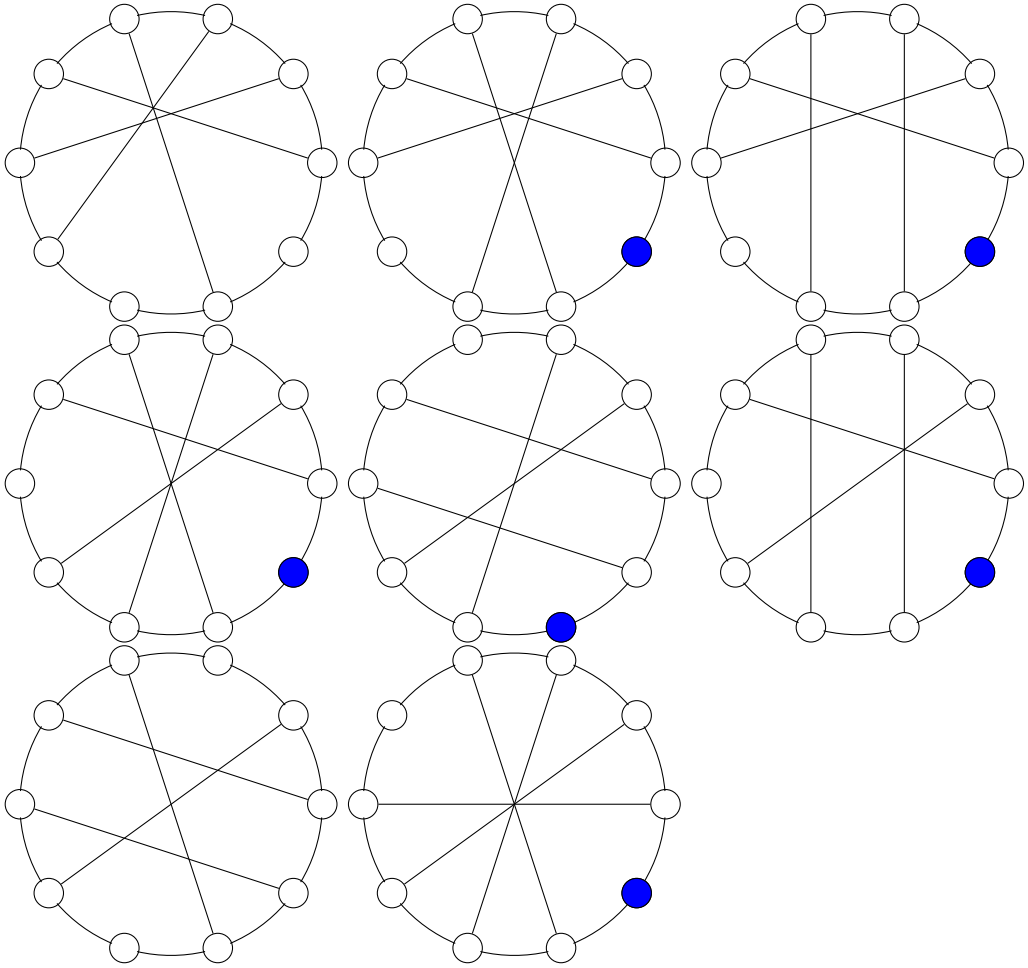












Cycles on 11 vertices with up to 3 chords: